

Emma Sofie Søvik  
Jesper Trøan  
Kristian Tveiten  
Nils Olav Tuv

## Extrusion Planner

Bachelor's project in Engineering - Computer Science

Supervisor: Sony George

Co-supervisor: Hilda Deborah

May 2021



Emma Sofie Søvik  
Jesper Trøan  
Kristian Tveiten  
Nils Olav Tuv

## **Extrusion Planner**

Bachelor's project in Engineering - Computer Science  
Supervisor: Sony George  
Co-supervisor: Hilda Deborah  
May 2021

Norwegian University of Science and Technology  
Faculty of Information Technology and Electrical Engineering  
Department of Computer Science





## Sammendrag av Bacheloroppgaven

Tittel:	<b>Extrusion planner</b>
Dato:	14.05.2021
Deltakere:	Jesper Trøan Kristian Tveiten Nils Olav Tuv Emma Sofie Sjøvik
Veiledere:	Sony George Hilda Deborah
Oppdragsgiver:	Benteler Automotive
Kontaktperson:	Gerda Nubdal
Nøkkelord:	Fullstack, API, React, JavaScript, Java, Spring, MSSQL, Web
Antall sider:	101
Antall vedlegg:	10
Tilgjengelighet:	Åpen

---

Sammendrag:	<p>Teknologi er i konstant endring, og det er viktig for selskapene å holde seg oppdatert på disse endringene for å rykke videre i bransjen. Spesielt moderne programvare forbedres raskt, og det er viktig å oppgradere gammel programvare for å følge nye standarder og trender for å unngå å være utdatert. På grunn av dette bestemte en gruppe fra den globale bedriften Benteler at det var på tide å oppgradere deres excelbaserte planleggingsverktøy, og ønsket at vi skulle være innovative og lage en ny programvare for å erstatte den.</p> <p>Løsningen er en web applikasjon som dekker alle steg i Benteler's planleggingsprosesser. Frontend koden er implementert med det populære JavaScript rammeverket React, og Backend API'et er utviklet i Java med hjelp av Spring Boot biblioteket. Vårt fokus var å lage et brukervennlig verktøy som kunne hjelpe Benteler rasjonalisere planleggingsprosessen deres, og for å gjøre dette fulgte vi dagens programvare trender, i tillegg til å finne innovative løsninger for å løse spesifikke problemer.</p> <p>Utviklingsmodellen Scrum ble brukt til å administrere prosjektet effektivt, og teamet fikk nyttig erfaring med full stack utvikling. Videre fikk teamet erfaring innen programmering, sikkerhet, testing og distribusjon, som til slutt resulterte i en komplett Extrusion Planner web applikasjon.</p>
-------------	--

## Summary of Bachelor Thesis

Title:	<b>Extrusion planner</b>
Date:	14.05.2021
Authors:	Jesper Trøan Kristian Tveiten Nils Olav Tuv Emma Sofie Sjøvik
Supervisor:	Sony George Hilda Deborah
Employer:	Benteler Automotive
Contact Person:	Gerda Nubdal
Keywords:	Fullstack, API, React, JavaScript, Java, Spring, MSSQL, Web
Pages:	101
Attachments:	10
Availability:	Open

---

**Abstract:**

Technology is in constant change and it is important for businesses to keep up with these changes to advance in the industry. Modern software in particular is rapidly improving, and upgrading old software to follow new standards and trends is key to avoid being outdated. On account of this, a group from the global enterprise Benteler decided it was time to upgrade their excel based planning tool and wanted us to be innovative and create a new software to replace it.

The solution is a web application that covers every step in Benteler's planning process. The frontend code is implemented with the popular JavaScript framework React, and the backend API is written in Java with the help of the Spring Boot library. Our focus was to create a user-friendly tool that could help Benteler streamline their planning process, and to do this we followed today's software trends as well as finding innovative solutions to solve certain problems.

The development model Scrum was used to manage the project effectively, and the team got useful experience in full stack development. Furthermore the team acquired experience within programming, security, testing and deployment, which in the end resulted in a complete Extrusion Planner web application.

## **Preface**

This bachelor thesis is written by Jesper Trøan, Kristian Tveiten, Nils Olav Tuv and Emma Sofie Søvik, students at NTNU Gjøvik, Department of Computer Science.

We wish to thank our supervisors Sony George and Hilda Deborah for good teamwork with regular meetings and guidance. We also want to thank Benteler Automotive Raufoss, especially our contact persons Gerda Nubdal and Frode Paulsen for great availability and follow-up. The assignment has provided us with great experience in full stack development and more.

Finally we want to thank each other for good cooperation during this project. This has been an educational and interesting process.

# Contents

<b>Preface</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>Listings</b>	<b>viii</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>Terms and abbreviations</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Background	2
1.1.1 The extrusion planning process	3
1.2 Task description	3
1.2.1 Functionality	4
1.2.2 User group	4
1.2.3 Profile recognition	4
1.2.4 Our contribution to the task	5
1.2.5 Why Benteler wanted a new solution	5
1.3 Limitations	8
1.3.1 Time limitations	8
1.3.2 Scope	8
1.4 Team members	8
1.4.1 Specific project roles	8
1.4.2 Earlier experience	9
1.4.3 What we had to learn	9
1.5 Why we chose this task	9
1.6 Thesis structure	10
<b>2 Development process</b>	<b>11</b>
2.1 System development model	11
2.2 Project characteristics	11
2.3 Choice of system development models	11
2.3.1 Following the Scrum and Kanban practice	12
2.3.2 Time estimation models	12
2.4 Process execution	13



2.4.1	Purpose of each sprint	13
2.4.2	Scrum board	14
2.4.3	Meeting minutes	15
2.4.4	Mindmanager	16
2.4.5	Project time usage	16
<b>3</b>	<b>Requirements</b>	<b>19</b>
3.1	Use Case diagram	19
3.1.1	High level Use Cases	21
3.1.2	Detailed Use Cases	23
3.2	Non functional requirements	26
3.3	Operational requirements	26
3.4	Security requirements	26
<b>4</b>	<b>Technologies</b>	<b>27</b>
4.1	The application type	27
4.2	Frontend	29
4.2.1	Resources used	29
4.2.2	Choice of React UI framework	30
4.3	Backend	30
4.3.1	Resources used	31
4.3.2	Learning material	32
4.4	Testing	32
<b>5</b>	<b>Design and implementation</b>	<b>33</b>
5.1	Application structure	33
5.1.1	JSON	35
5.2	Frontend web-interface	36
5.2.1	Design	39
5.2.2	Setup of the forms	40
5.2.3	Functions in TFC	40
5.2.4	Automatic version and status control	41
5.2.5	Invalid requests	44
5.2.6	Change measurement system	45
5.2.7	Security	46
5.3	Backend API	54
5.3.1	API Layer / Controller classes	56
5.3.2	Model layer	57
5.3.3	Service layer	58
5.3.4	Database	58
5.3.5	CostCalc/Product Controller data export document	60
5.3.6	Security	62
5.4	Deployment of the application	65
5.4.1	Backend deployment	65
5.4.2	Frontend deployment	65
<b>6</b>	<b>User interface</b>	<b>67</b>
6.1	Color pallet	67
6.2	Layout	68

6.2.1	Side menu . . . . .	68
6.2.2	Topbar . . . . .	70
6.2.3	Footer . . . . .	71
6.3	Pages . . . . .	71
6.3.1	Login page . . . . .	71
6.3.2	Search page . . . . .	72
6.3.3	Request, TFC, CostCalc and Order main pages . . . . .	74
6.3.4	The forms . . . . .	76
6.3.5	Edit Database page . . . . .	79
6.3.6	Users page . . . . .	83
6.3.7	Home and FAQ page . . . . .	84
<b>7</b>	<b>Overview of the development environments</b>	<b>86</b>
7.1	Frontend . . . . .	87
7.2	Backend . . . . .	87
7.3	Database . . . . .	88
<b>8</b>	<b>Code quality</b>	<b>89</b>
8.1	Code review . . . . .	89
8.2	Code documentation . . . . .	90
8.3	Frontend . . . . .	91
8.4	Backend . . . . .	92
8.5	Code redundancy . . . . .	92
8.5.1	Large SQL queries in the API . . . . .	92
8.5.2	Large ResultSet to object conversions and prepared statements value bindings . . . . .	92
<b>9</b>	<b>Testing</b>	<b>93</b>
9.1	Cross-browser compatibility . . . . .	93
9.2	Acceptance tests . . . . .	93
9.3	Lighthouse . . . . .	94
9.4	Traffic testing of the server . . . . .	95
9.5	Unit tests . . . . .	96
<b>10</b>	<b>Conclusion</b>	<b>98</b>
10.1	Summary of contributions . . . . .	98
10.2	Feedback from Benteler . . . . .	99
10.3	Future Work . . . . .	100
10.4	Learning Outcomes and Concluding Remarks . . . . .	101
	<b>Bibliography</b>	<b>102</b>
<b>A</b>	<b>Project plan</b>	<b>104</b>
<b>B</b>	<b>Documentation and procedures regarding the code, web-server and code assuring longevity and re-usability of the application</b>	<b>119</b>
<b>C</b>	<b>Feedback from testing phase with solutions and status per case</b>	<b>148</b>
<b>D</b>	<b>CostCalc document code</b>	<b>152</b>

<b>E Mindmanager map</b>	<b>156</b>
<b>F Project agreement</b>	<b>158</b>
<b>G Group rules</b>	<b>162</b>
<b>H Meeting minutes</b>	<b>164</b>
<b>I Full project time usage report</b>	<b>178</b>
<b>J Feedback from Benteler</b>	<b>183</b>

# Listings

5.1	Axios GET	35
5.2	Response from GET request	36
5.3	Example of Routing	37
5.4	ReactDOM render function	37
5.5	Index.html	38
5.6	MakeStyles	39
5.7	CSS progressbar	39
5.8	Structure of a form	40
5.9	React hooks	40
5.10	useEffect hook	41
5.11	Frontend API call for retrieving the latest Request version per a given ID	41
5.12	Backend API function for retrieving the latest Request version per a given ID	42
5.13	Example functions for automatic status control in frontend following a submission of a form	43
5.14	Functions for automatic status control in the backend API following a submission	43
5.15	Conversion example	45
5.16	Authentication request	47
5.17	Authentication function in authContext	48
5.18	isAuthenticated in authContext	48
5.19	The routing authentication	49
5.20	The authorization functions	50
5.21	An example showing the authorization routing	50
5.22	Authorization in the sidemenu	51
5.23	Email validator	52
5.24	Lazy Loading	53
5.25	Strength meter	54
5.26	Example of controller class	56
5.27	Controller class receiving URL parameters	56
5.28	Example of a smaller model class	57
5.29	Example of a smaller service class	58
5.30	Snippet of CostCalc main function see appendix D for full code	60
5.31	JWTService: Creation of a JWT	62
5.32	JWTService: Assert a JSON Web Token	63
5.33	JWT Controller class (API Endpoint)	63
5.34	Prepared statement	64
5.35	Hashing function	64
5.36	web.config IIS module configuration	66
8.1	Example of docstring in Java	90
9.1	Command to run traffic test	95

9.2 Function to retrieve a specific press . . . . . 97  
D.1 Cost Calc main function for producing a bridge file with accessory functions . . . . . 152

# List of Figures

1.1	Process parameters . . . . .	2
1.2	Flowchart explaining properties of the process planning . . . . .	3
1.3	User hierarchy . . . . .	4
1.4	Profile example . . . . .	5
1.5	The main page on Benteler's old system . . . . .	6
1.6	The search page before the TFC form . . . . .	7
1.7	The old TFC form . . . . .	7
2.1	Sprint overview . . . . .	13
2.2	Scrum board in Trello . . . . .	15
2.3	Meeting minutes example . . . . .	16
2.4	Structure of a meeting in mindmanager . . . . .	16
2.5	Working hours spent on different activities . . . . .	17
2.6	Working hours spent per week . . . . .	18
3.1	Use Case diagram . . . . .	20
5.1	MVC pattern of the system . . . . .	33
5.2	Main Application Structure . . . . .	34
5.3	Overview over the code . . . . .	35
5.4	Layout of the application . . . . .	36
5.5	Frontend structure . . . . .	38
5.6	Status field/overview . . . . .	44
5.7	Status overview in popup after selecting an old TFC as base for a new TFC . . . . .	44
5.8	Toggle invalid Request with disabled buttons . . . . .	45
5.9	Backend API data flow when submitting data . . . . .	55
5.10	Backend API data flow when fetching data . . . . .	55
5.11	Database Structure . . . . .	59
5.12	API Service folder . . . . .	65
6.1	Benteler color guideline . . . . .	67
6.2	Closed menu . . . . .	69
6.3	Opened menu . . . . .	69
6.4	Topbar, with the user settings opened . . . . .	70
6.5	The change password popup . . . . .	70
6.6	Footer . . . . .	71
6.7	Login page . . . . .	71
6.8	Search page with an Alloy filter . . . . .	72
6.9	Popup menu with parameter choice . . . . .	74

---

6.10	TFC search page	75
6.11	Request form	76
6.12	Profile category in the Request-form	77
6.13	Input field	77
6.14	Drop-down-menu field	77
6.15	Disabled field	78
6.16	Input-field with a limit warning.	78
6.17	Input-field with conversion to inches.	78
6.18	TFC-form	79
6.19	The Edit Database page	80
6.20	Popup to update a press	81
6.21	Edit Database - Alloy	82
6.22	Edit Database - Others	82
6.23	Edit Database - Others with Customers list opened	83
6.24	Users page	83
6.25	The user popup menu	84
6.26	Home page of the application	84
6.27	The FAQ page displaying questions and answers in expandable UI boxes	85
7.1	Overview of the development environments	86
8.1	A card with QA in trello	90
8.2	Docstring interface example	91
9.1	Excerpt from LOP action items list	94
9.2	Our lighthouse score	94
9.3	Results of traffictesting using autocannon	95
9.4	Postman example: Testing getPressByName functionality	97

# List of Tables

4.1	Overview over pros and cons for desktop vs web application . . . . .	28
10.1	Extract of survey result . . . . .	100



## Terms and abbreviations

<b>API</b>	Application Programming Interface. An interface that defines interactions between multiple software applications.
<b>Backend</b>	Server side. What happens on the server and the database.
<b>Client/Stakeholder</b>	The company supplying this thesis project; Benteler.
<b>CSS</b>	Cascading Style Sheets. Specifies a documents style and layout.
<b>Extrusion</b>	The process when a material is pushed through a die to form a profile.
<b>Form</b>	In this context, a form is the input forms used for Requests, Orders and TFC's.
<b>Framework</b>	Resources and tools used to build and manage web applications.
<b>Frontend</b>	What happens in the browser. What the user see and interacts with.
<b>Full stack project</b>	A project containing all work of database, server, system engineering and client side.
<b>High Level</b>	Non-detailed explanation to get an overview.
<b>IDE</b>	Integrated Development Environment. A software application that provides comprehensive facilities for software development.
<b>JDBC</b>	Java Database Connectivity
<b>JWT</b>	JSON Web Token, a token where session and user data is stored on the users machine and it contains anti-tamper measures.
<b>Order</b>	An order based on Request data.
<b>Request</b>	The Request document is a initial document containing only a certain set of parameters.
<b>Team</b>	The student group working on this project.
<b>TFC</b>	Team Feasibility Commitment. A document result of a meeting, where all parameters necessary for the production of a profile is calculated/extracted into this document.
<b>UI</b>	User Interface.
<b>VPN</b>	Virtual Private Network.

# Chapter 1

## Introduction

### 1.1 Background

Benteler is a big worldwide company that has its headquarters in Salzburg, Austria. The company employs around 30 000 employees from around 100 locations around the globe [37], and Raufoss in Norway is one of them. Extrusion is done to make metal profiles with a specific shape, and Benteler is currently doing extrusion in France, USA and Norway [6]. Extrusion is done by heating up material, which is then pushed through a die of the desired cross-section. This may be reminiscent of the toothpaste effect where material is forced through the die and becomes the desired shape. After the material is put through the die, the material will be cooled down, straightened out and cut to the correct lengths. One of the key parameters through the whole extrusion process is temperature. As you can see in figure 1.1 the keys to achieve good surface quality and at the same time maximum productivity is to always be at the correct temperature and ram speed.

The cost for producing the profiles is given by the size and complexity of the profile as well as the type of aluminum alloy. Benteler in Raufoss, Norway has an existing application for planning the profiles and calculating the costs. This Extrusion planner application lacks wanted functionality and does not cover the needed operational requirements. This led Benteler to propose this task to create a new improved and faster system for planning and calculating profiles.

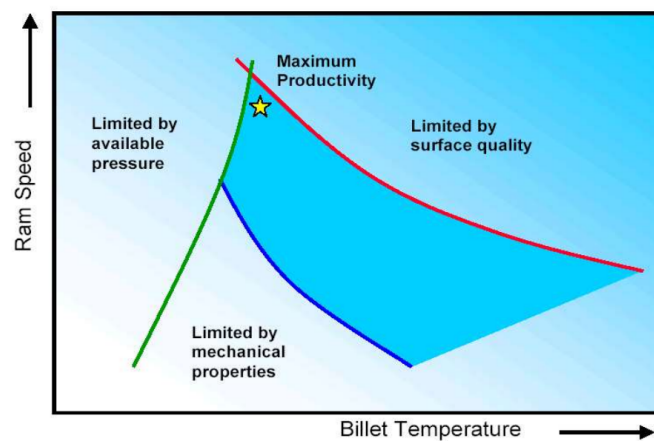


Figure 1.1: Process parameters

### 1.1.1 The extrusion planning process

Benteler has a structured and efficient extrusion process they follow for each new profile (see Figure 1.2). This applies to both the planning phase and the production phase. We will focus only on the planning phase, since that is what our application should contribute to.

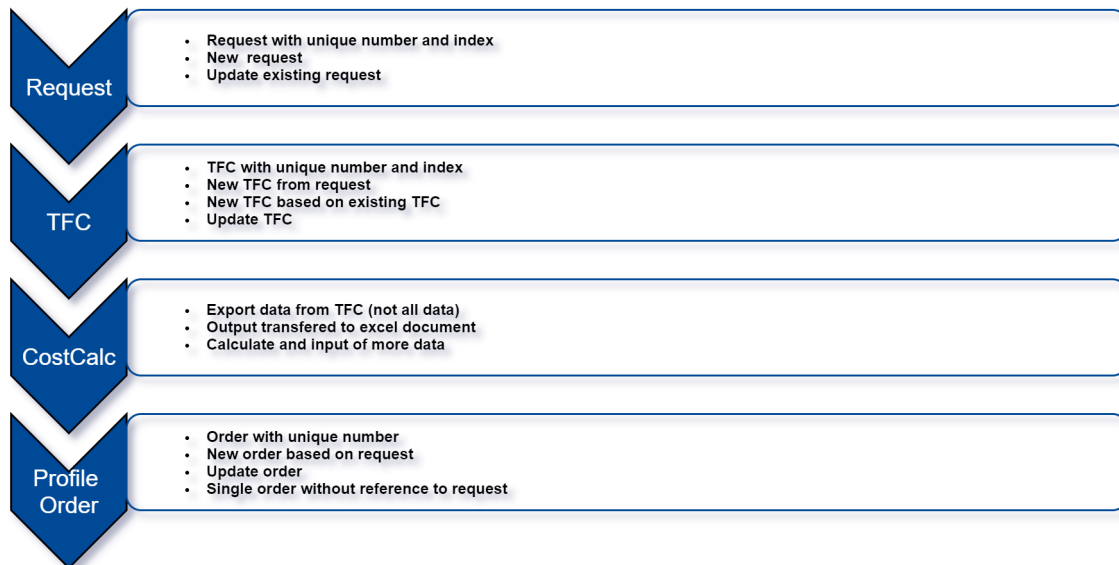


Figure 1.2: Flowchart explaining properties of the process planning

**Request** is the first part of the extrusion planning process. Every profile starts with a request. This request is made by an employee and contains specific values for the desired profile.

**TFC**, after a Request is made a project planner will call in a TFC-meeting (Team Feasibility Commitment meeting). The meeting attendees will review the pending Request, and will discuss and decide relevant planning parameters for the new profile. Sometimes one Request requires multiple TFC-meetings, for instance when alternative extrusion presses are considered for extruding the profile.

**Calculations**, specified parameters from the Request and TFC-report are needed for cost calculations of the profile. Cost calculations are made by the CostCalc team. They fetch values from the Request and TFC and plot them into their own program to do the required calculations.

**Order**, after the TFC-meeting and calculations have been made the next step is to order the new profile. This is done only if the team has decided that the profile is possible and profitable.

## 1.2 Task description

The task in short is to create a system for planning operating parameters (and thus the cost) for new or changed profiles. Benteler did not have any predefined requirements for what technologies and tools the solution should have, but they had an existing database that the old system used and that could be used or modified in the new application.

### 1.2.1 Functionality

Benteler wanted the solution to contribute to every phase in the extrusion planning process. They wanted employees to utilize the application to create new Requests. Also, the solution should make it possible for employees to review and modify existing Requests. The solution should in the same way handle TFC reports and Orders, and it should be easy for the CostCalc team to receive the needed values for their calculations. A more precise description of the desired functionality for the application is written in chapter 3, Requirements, below.

### 1.2.2 User group

There were about 20 employees who had access to Benteler's old system. With the new solution they wish to expand and make it accessible for more employees. Thus they want to divide the users into several user roles. Users were divided into 4 roles; CostCalc-team, normal user, superuser and admin. In which admins have the most rights above superuser who has the second most rights. After that are normal users and the CostCalc-team. This works in the way that admins inherit all the rights superusers have, plus a few extras, and superusers inherit every right a normal user and CostCalc user have (see Figure 1.3). You can read more about the user roles in the requirements description in chapter 3.

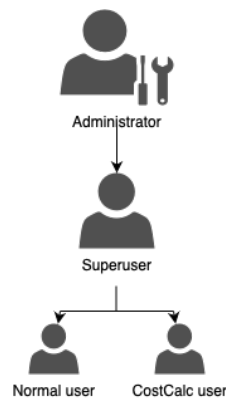


Figure 1.3: User hierarchy

### 1.2.3 Profile recognition

As an extra task, Benteler also had an interest in having a system that could take a profile sketch and compare it with existing profiles to find similar and relevant profiles.

In this context, a profile is the geometric shape of a product Benteler is going to extrude for a client. See Figure 1.4. This task was put aside, as Benteler clearly stated that this was a bonus-task if we had time by the end of the project.

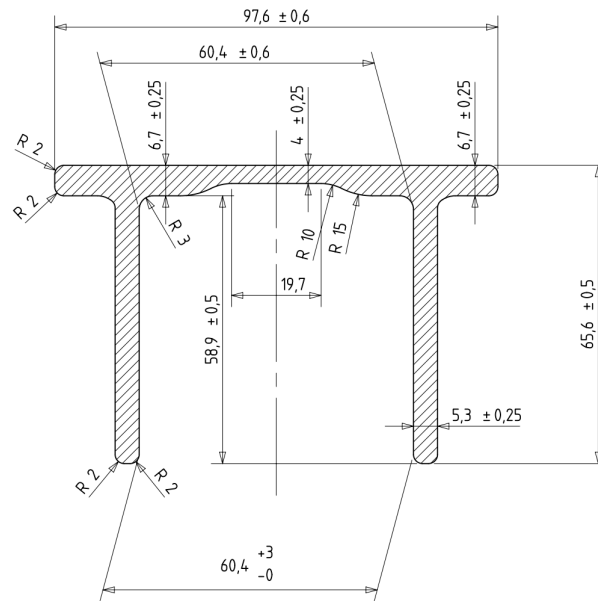


Figure 1.4: Profile example

### 1.2.4 Our contribution to the task

Benteler did not have any predefined requirement specification for what technologies and tools the solution should have. This meant that we had free disposal to evaluate and select the tools and solutions that would fit the task at hand in the best way possible. These solutions were then proposed to the client for approval.

One of the biggest decisions was whether we were to develop a desktop application or a web application. After discussing, the team identified that a web application would fit the task the best, with the main reasoning that a web application would be the most accessible solution and it would make it easier for Benteler to have more employees utilize the application.

Seeing that we lacked experience in web development we knew that a period of the project would go to researching and learning about web development. Consequently, this meant that we would have less time for extra functionality like profile recognition.

The team and Benteler discussed the two options with pros and cons for both of them and concluded that web application would be the best fit even though the research and learning period would be longer.

Another decision that was made was to keep Benteler's existing database. More about the technology choices and reasoning behind it is written in the Technologies chapter 4.

### 1.2.5 Why Benteler wanted a new solution

Considering the fact that Benteler already had an existing application to handle extrusion planning, we could simply describe the task as to improve the system. However, Benteler was looking for a new solution, since their existing one was slow, inefficient and were lacking in key departments and functionality. The old solution relies heavily on passing excel documents around by email which is one of the reasons why Benteler wanted us to come up with new solutions and ideas that can make the extrusion planning process run smoothly.

Therefore, we decided not to use the old application as a base, and instead develop our new application from scratch based on the requirements and wishes Benteler gave us.

Figure 1.5 is the main page of Benteler's old application.

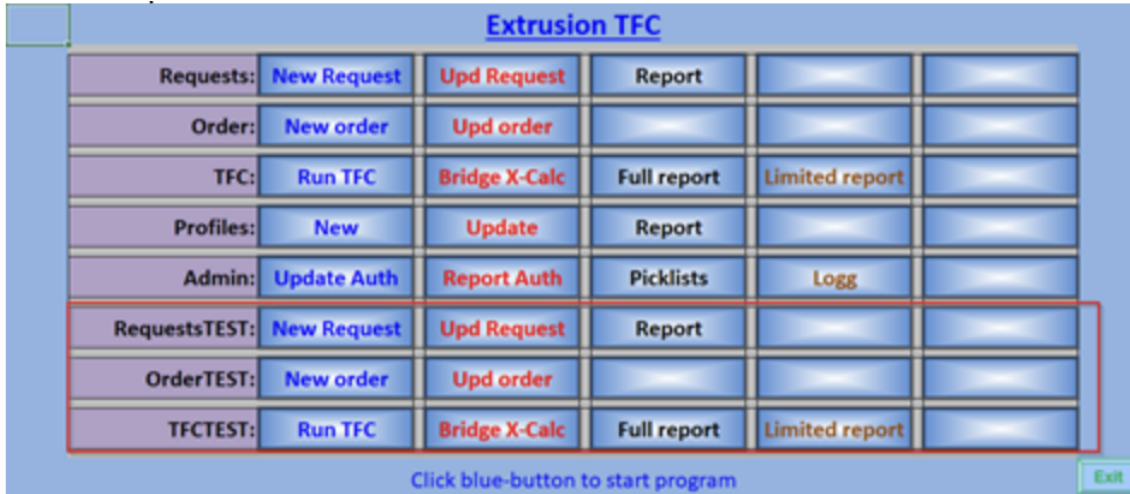


Figure 1.5: The main page on Benteler's old system

As you can see from this figure, the main page of Benteler's old application consisted of a start menu with all the different choices for the application, which meant that the users of the system had to use this page to navigate through the application.

The Request, Order and TFC rows all contain different options within the category. For example TFC's four options: "Run TFC" to create a new TFC document. "Bridge X-Calc" to export specific TFC to the CostCalc program. "Full report" and "Limited report" to fetch specific TFC reports. The "Admin" row contains options for the administrator to edit parts of the application, like certain values used in formulas.

The bottom three rows in the figure were only used by the developers for testing and were not something the new solution needed to have.

To further explain the UI of the old system, we will take a look at how to create a new TFC, with the reasoning that the process for creating a new TFC is fairly similar to creating Request and Orders.

After clicking on the "Run TFC" option on the main page, the user gets sent to a search page containing the most recent TFCs (see Figure 1.6). The search page contains a limited search box to insert search criteria on only a few pre-selected parameters as well as a table of the most recent TFCs. On the search page, the user has the option to select a specific TFC to insert into the forms or to create a form from scratch with only Request values. The user also has an option to remove existing TFCs.

request_no	request_ver_no	sketch	tfc_no	Alloy	press	Customer	Requestor_name	Request_date	Part_description	Annual_tonnage_kg	Info	Profile_je_rgh_mm	Tolerance_mm	Surface	Profile_ty	Profile_w_eight_kg	Extrusion_speed_m_min	Recovery_pct	No_of_cavities	Billet_gh_mm	Len_gth_mm	
7838	1	097-20	2	6062.27	P22	Audi	Orstian Inge Asgerinso	04.11.2020	Rear Sidemember	39 211	Lk 7337-2_127-18E. Behov	5050	±5		O	1,665	18	83,11917	1	750		
7838	1	097-20	1	6062.27	P22	Audi	Orstian Inge Asgerinso	04.11.2020	Rear Sidemember	39 211	Lk 7337-2_127-18E. Behov	4740	±5		O	1,665	18	77,62978	1	750		
7937	1	095-20	1	6063.85	P40	BMW	rdia.Nubdal@benteler.c	02.11.2020	tutl	228 358	7937-114.09.2020. Ref. RV	4450	±5		H	4,237	11	77,78113	1	950		
7936	1	095-20	1	6063.85	P40	BMW	rdia.Nubdal@benteler.c	30.09.2020		1	237 336	7936/14.09.2020. Ref. RA	4900	±5		H	4,366	9	75,80285	1	950	
7935	1	094-20	1	7003.30	P55	Audi	dro.Velasquez@bentel	30.09.2020		1	982 375	7935-114.09.2020. Videne	1128	±2		H	2,903	9	77,73947	1	1050	
7934	1	093-20	1	6062.27	P22	Audi	dro.Velasquez@bentel	30.09.2020		1	151 511	7934-108.09.2020. Kun per	5400	0		O	1,935	13	74,50297	1	750	
7933	1	092-20	1	6060.35	P22	Audi	dro.Velasquez@bentel	17.09.2020	tutlball	154 074	7933-108.09.2020. Mexico	5560	±5		H	1,409	22	79,36431	1	650		
7932	1	091-20	1	6062.27	P40	Audi	dro.Velasquez@bentel	17.09.2020	Frt tutt	692 846	7932-107.09.2020. Er en k	1289	±2		H	4,066	11	71,97884	1	950		
7931	1	090-20	1	7046.92	P22	Audi	dro.Velasquez@bentel	16.09.2020		1	103 723	7931-103.09.2020. Foruste	1200	±2		O	2,401	8	73,60313	1	750	
7930	1	089-20	1	6060.35	P40	Audi	dro.Velasquez@bentel	15.09.2020	s	57 194	7930-131.08.2020. Videne	4530	±5		O	2,837	15	72,59939	1	990		
7929	1	088-20	1	7108.50	P22	Audi	gjus.Livanas@bentele	15.09.2020	Test	131 519		0	1195	0	H	2,317	12	72,5286	1	750		
7928	1	087-20	1	7108.50	P55	1	gerda	01.09.2020		10	531 790	7928-124.08.2020. Likner	1720	±2		H	5,153	7,5	70,5573	1	1250	
7928	2	087-20A	1	7108.50	P55	1	gerda	14.09.2020		10	545 722	7928-124.08.2020. Likner	1720	±2		H	5,288	8	72,58917	1	1250	
7927	1	086-20A	1	6060.35	P22	BMW	rdia.Nubdal@benteler.c	01.09.2020	testb	18 144		0	5140	0	H	0,432	20	76,02855	4	750		
7927	2	086-20A	1	6060.35	P22	BMW	rdia.Nubdal@benteler.c	14.09.2020	testb	1	7927-2/26.11.2020. Geomet	5260	0		H	0,424	20	78,14842	4	750		
7926	1	095-20	1	6060.35	P22	BMW	rdia.Nubdal@benteler.c	01.09.2020	Test profile	89 034		0	4600	0	H	1,349	20	72,37538	2	750		
7926	1	085-20A	1	6060.35	P22	BMW	rdia.Nubdal@benteler.c	01.09.2020	Test profile	85 208		0	5270	0	H	0,988	20	76,17333	2	750		
7926	2	085-20B	1	6060.35	P22	BMW	rdia.Nubdal@benteler.c	14.09.2020	Test profile	63 310	7926/22.10.2020. Geometri	5340	0		H	0,977	20	76,28564	2	750		
7926	3	085-20C	1	6060.35	P22	BMW	rdia.Nubdal@benteler.c	14.09.2020	Test profile	92 470	7926/22.10.2020. Geometri	4410	0		H	1,427	20	72,37332	2	750		
7925	1	084-20	1	7003.30	P40	BAS	rdia.Nubdal@benteler.c	25.08.2020	Testmateriale	75 392		0	4600	0	O	3,523	20	80,33584	1	900		
7924	1	083-20	1	6060.35	P40	JLR	Frøde Paulsen	24.08.2020	Frt Beam	38 355	18.08.2020. Ref. RAB6682	4920	±5		O	2,305	15	74,10906	1	990		

Figure 1.6: The search page before the TFC form

If the user selects a specific TFC and clicks the "Start With old" button, the user gets sent to the TFC form (which is shown in Figure 1.7) with the specific TFC values inserted in the form. The user then has the options to change the desired values and send the new TFC to the database.

Request	Request ver. no	Project	SOP (yyyy-mm)	Average volume (y)	Part	TFC date																																																											
7878	1	Porsche	2022-06	160000	PPE41 ULE Front Beam	24.06.2020																																																											
<table border="1"> <tr> <td>Name requestor</td> <td>Andreas Velasquez</td> <td>Project name</td> <td>x</td> <td>Place of delivery</td> <td>HARA</td> <td>Part category</td> <td>Parts pr vehicle</td> <td>1</td> <td>Participants</td> <td>Adriaens, Gerda, Per Ivar, Bjørn, Jonas, Morten, Erik</td> </tr> <tr> <td>E-mail Cost_calc</td> <td>151149</td> <td>Project no.(PPA)</td> <td></td> <td>Project nr SAP</td> <td></td> <td>Part length</td> <td>SAP/CAD model</td> <td>1074</td> <td></td> <td></td> </tr> </table>							Name requestor	Andreas Velasquez	Project name	x	Place of delivery	HARA	Part category	Parts pr vehicle	1	Participants	Adriaens, Gerda, Per Ivar, Bjørn, Jonas, Morten, Erik	E-mail Cost_calc	151149	Project no.(PPA)		Project nr SAP		Part length	SAP/CAD model	1074																																							
Name requestor	Andreas Velasquez	Project name	x	Place of delivery	HARA	Part category	Parts pr vehicle	1	Participants	Adriaens, Gerda, Per Ivar, Bjørn, Jonas, Morten, Erik																																																							
E-mail Cost_calc	151149	Project no.(PPA)		Project nr SAP		Part length	SAP/CAD model	1074																																																									
<table border="1"> <tr> <td>Alloy</td> <td>6063.85</td> <td>Customer standard</td> <td>Geometric standard</td> <td>Surface quality</td> <td>Other standards</td> <td>Res req.</td> <td>IGC req.</td> <td>Ductility req.</td> </tr> <tr> <td>Temper</td> <td>T1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>							Alloy	6063.85	Customer standard	Geometric standard	Surface quality	Other standards	Res req.	IGC req.	Ductility req.	Temper	T1																																																
Alloy	6063.85	Customer standard	Geometric standard	Surface quality	Other standards	Res req.	IGC req.	Ductility req.																																																									
Temper	T1																																																																
<table border="1"> <tr> <td>Profile</td> <td>037-20</td> <td>A</td> <td>Profile weight/meter</td> <td>1,057 kg/m</td> <td>Min wall th</td> <td>1,9 mm</td> <td>Circ. Circle</td> <td>76,7 mm</td> </tr> <tr> <td>Alloy</td> <td>6063.85</td> <td></td> <td>Hollow/Open</td> <td>H</td> <td>Max wall th</td> <td>1,9 mm</td> <td>Chamber area</td> <td>12,86 cm2</td> </tr> <tr> <td>Temper</td> <td>T1</td> <td></td> <td>Profile length</td> <td>1 074 mm</td> <td>Min corner radius inner</td> <td>2,0 mm</td> <td>No. Of chamb.</td> <td>1</td> </tr> <tr> <td>Profile no.</td> <td></td> <td></td> <td></td> <td></td> <td>Min corner radius outer</td> <td>1,0 mm</td> <td></td> <td></td> </tr> </table>							Profile	037-20	A	Profile weight/meter	1,057 kg/m	Min wall th	1,9 mm	Circ. Circle	76,7 mm	Alloy	6063.85		Hollow/Open	H	Max wall th	1,9 mm	Chamber area	12,86 cm2	Temper	T1		Profile length	1 074 mm	Min corner radius inner	2,0 mm	No. Of chamb.	1	Profile no.					Min corner radius outer	1,0 mm																									
Profile	037-20	A	Profile weight/meter	1,057 kg/m	Min wall th	1,9 mm	Circ. Circle	76,7 mm																																																									
Alloy	6063.85		Hollow/Open	H	Max wall th	1,9 mm	Chamber area	12,86 cm2																																																									
Temper	T1		Profile length	1 074 mm	Min corner radius inner	2,0 mm	No. Of chamb.	1																																																									
Profile no.					Min corner radius outer	1,0 mm																																																											
<table border="1"> <tr> <td>Press</td> <td>P22</td> <td>Reduction ratio:</td> <td>4,4</td> <td>Cavities</td> <td>2</td> <td>Billet length (mm)</td> <td>750 mm</td> <td>Billet Ø (mm)</td> <td>203</td> <td>Butt end (mm)</td> <td>25</td> <td>Butt end weight (kg)</td> <td>2,34 kg</td> </tr> </table>							Press	P22	Reduction ratio:	4,4	Cavities	2	Billet length (mm)	750 mm	Billet Ø (mm)	203	Butt end (mm)	25	Butt end weight (kg)	2,34 kg																																													
Press	P22	Reduction ratio:	4,4	Cavities	2	Billet length (mm)	750 mm	Billet Ø (mm)	203	Butt end (mm)	25	Butt end weight (kg)	2,34 kg																																																				
<table border="1"> <tr> <td>Process</td> <td>Extrusion speed</td> <td>20,0 m/min</td> <td>Ram speed</td> <td>7,54 mm/s</td> <td>Batch size</td> <td>44 billets</td> <td>Net productivity</td> <td>1 545 kg/h</td> <td>Recovery</td> <td>78,0 %</td> <td>Annual tonnage</td> <td>181 635 kg</td> <td>Optimal lengths</td> <td>1 102 mm</td> </tr> <tr> <td>Scrap</td> <td>Front</td> <td>2,0 m</td> <td>Rear</td> <td>1,0 m</td> <td>Gross extr length</td> <td>29,90 m</td> <td>Gross extr length</td> <td>1 981 kg/h</td> <td>Recovery 2</td> <td>83,0 %</td> <td>Weekly tonnage</td> <td>3 949 kg</td> <td>1 058 mm</td> </tr> <tr> <td>Steady state</td> <td>Front first</td> <td>3,0 m</td> <td>Rear last</td> <td>1,1 m</td> <td>Net extr length</td> <td>25,45 m</td> <td>Cycle time per billet</td> <td>119,1 s</td> <td>Recovery 1</td> <td>83,1 %</td> <td>Calc hours per week</td> <td>2,6</td> <td>Profile weight</td> <td>1,135 kg</td> </tr> <tr> <td>Additional</td> <td>Non perf. Scrap</td> <td>5 %</td> <td>Safety margin</td> <td>1,5 %</td> <td>Profiles per billet</td> <td>48</td> <td>Profile left over /out rest</td> <td>0,67 m</td> <td></td> <td></td> <td>Calc billets per week</td> <td>77,2</td> <td>Cooling</td> <td></td> </tr> </table>							Process	Extrusion speed	20,0 m/min	Ram speed	7,54 mm/s	Batch size	44 billets	Net productivity	1 545 kg/h	Recovery	78,0 %	Annual tonnage	181 635 kg	Optimal lengths	1 102 mm	Scrap	Front	2,0 m	Rear	1,0 m	Gross extr length	29,90 m	Gross extr length	1 981 kg/h	Recovery 2	83,0 %	Weekly tonnage	3 949 kg	1 058 mm	Steady state	Front first	3,0 m	Rear last	1,1 m	Net extr length	25,45 m	Cycle time per billet	119,1 s	Recovery 1	83,1 %	Calc hours per week	2,6	Profile weight	1,135 kg	Additional	Non perf. Scrap	5 %	Safety margin	1,5 %	Profiles per billet	48	Profile left over /out rest	0,67 m			Calc billets per week	77,2	Cooling	
Process	Extrusion speed	20,0 m/min	Ram speed	7,54 mm/s	Batch size	44 billets	Net productivity	1 545 kg/h	Recovery	78,0 %	Annual tonnage	181 635 kg	Optimal lengths	1 102 mm																																																			
Scrap	Front	2,0 m	Rear	1,0 m	Gross extr length	29,90 m	Gross extr length	1 981 kg/h	Recovery 2	83,0 %	Weekly tonnage	3 949 kg	1 058 mm																																																				
Steady state	Front first	3,0 m	Rear last	1,1 m	Net extr length	25,45 m	Cycle time per billet	119,1 s	Recovery 1	83,1 %	Calc hours per week	2,6	Profile weight	1,135 kg																																																			
Additional	Non perf. Scrap	5 %	Safety margin	1,5 %	Profiles per billet	48	Profile left over /out rest	0,67 m			Calc billets per week	77,2	Cooling																																																				
<table border="1"> <tr> <td>Die &amp; bolster</td> <td>Die type &amp; size</td> <td>PH285x160</td> <td>Die cost</td> <td>20 000 NOK</td> <td>Run limit</td> <td>44 billets</td> <td>Net length [m]</td> <td>1 160 m</td> <td>Saw:</td> <td>Prof. width on saw table</td> <td>51,0 mm</td> </tr> <tr> <td>Currency</td> <td>Botster cost</td> <td>B-V-2-26</td> <td>Botster cost</td> <td>14 000 NOK</td> <td>Die life time</td> <td>450 billets</td> <td>Die cost per kg</td> <td>0,37 NOK/kg</td> <td>Profile height</td> <td>66 mm</td> </tr> <tr> <td>NOKEUR,USD</td> <td>Insert</td> <td></td> <td>Manchrel set</td> <td>1</td> <td>Billet temp</td> <td>470 °C</td> <td></td> <td></td> <td>Cavities stacked</td> <td>No</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>Saw time (saw table)</td> <td>20,6 min</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>Extr. time (saw table)</td> <td>21,8 min</td> </tr> </table>							Die & bolster	Die type & size	PH285x160	Die cost	20 000 NOK	Run limit	44 billets	Net length [m]	1 160 m	Saw:	Prof. width on saw table	51,0 mm	Currency	Botster cost	B-V-2-26	Botster cost	14 000 NOK	Die life time	450 billets	Die cost per kg	0,37 NOK/kg	Profile height	66 mm	NOKEUR,USD	Insert		Manchrel set	1	Billet temp	470 °C			Cavities stacked	No										Saw time (saw table)	20,6 min										Extr. time (saw table)	21,8 min			
Die & bolster	Die type & size	PH285x160	Die cost	20 000 NOK	Run limit	44 billets	Net length [m]	1 160 m	Saw:	Prof. width on saw table	51,0 mm																																																						
Currency	Botster cost	B-V-2-26	Botster cost	14 000 NOK	Die life time	450 billets	Die cost per kg	0,37 NOK/kg	Profile height	66 mm																																																							
NOKEUR,USD	Insert		Manchrel set	1	Billet temp	470 °C			Cavities stacked	No																																																							
									Saw time (saw table)	20,6 min																																																							
									Extr. time (saw table)	21,8 min																																																							
<table border="1"> <tr> <td>Packing</td> <td>Pallet / rack type</td> <td>Europallet</td> <td>Profiles per layer</td> <td>30</td> <td>Number of layers</td> <td>10</td> <td>Profiles per pallet/rack</td> <td>300</td> <td>Spacer type</td> <td>Plastic</td> <td>No of spacers per layer</td> <td>Gross weight pallet/rack</td> <td>341 kg</td> </tr> <tr> <td></td> <td>Additional description</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>							Packing	Pallet / rack type	Europallet	Profiles per layer	30	Number of layers	10	Profiles per pallet/rack	300	Spacer type	Plastic	No of spacers per layer	Gross weight pallet/rack	341 kg		Additional description																																											
Packing	Pallet / rack type	Europallet	Profiles per layer	30	Number of layers	10	Profiles per pallet/rack	300	Spacer type	Plastic	No of spacers per layer	Gross weight pallet/rack	341 kg																																																				
	Additional description																																																																
<table border="1"> <tr> <td>Comments</td> <td colspan="6">ref: RAB6687. Bra nok kjøling? RAB6687 er i 6005.63. 23.06.2020. Geometriending.RL</td> <td>Investments required / assumed:</td> <td></td> </tr> </table>							Comments	ref: RAB6687. Bra nok kjøling? RAB6687 er i 6005.63. 23.06.2020. Geometriending.RL						Investments required / assumed:																																																			
Comments	ref: RAB6687. Bra nok kjøling? RAB6687 er i 6005.63. 23.06.2020. Geometriending.RL						Investments required / assumed:																																																										

Figure 1.7: The old TFC form

As explained earlier, the process for creating a Request and Order is similar to the TFC process, and the different search pages and forms all have a very similar design.

Even though we decided to not use the old application as a base for our new one, we still spent time reviewing their old system. We made an effort to understand what our application needed to do better, since our main goal was to make an application that could replace it.

## 1.3 Limitations

The project had a couple of limitations, especially related to time and scope. There were no financial limitations, since Benteler was responsible for the servers and there were no other expenditures connected to this project.

### 1.3.1 Time limitations

The due date for the bachelor thesis report was set the 20th of May. Even though the report was written over the whole period, we decided that we would finish the project development and deploy during the last week of April. This gave us time to finalize the report before the deadline.

### 1.3.2 Scope

The scope was limited to a web application optimized to be used on a computer, since Benteler does not use smartphones or tablets to complete these tasks. Benteler is an international company and the software is going to be used worldwide. Because of this, we chose to limit the language of the application, code and documentation to English.

## 1.4 Team members

The team members are Emma Sofie Søvik, Jesper Trøan, Nils Olav Tuv and Kristian Tveiten. We are all in the same study program, computer science. Consequently, the group members share the same professional experience.

### 1.4.1 Specific project roles

Working on the project, every team member had a general role of being developers, where we all worked on the entirety of the application. After developing the different modules of the application, we each took responsibility of a larger section of the application of which we had worked the most with.

**Jesper Trøan** was selected as the project leader, which had the main responsibility of making sure the meeting agenda was followed, keeping the MindManager map up to date and write meeting minutes. In addition to this he took the main responsibility for the layout and infrastructure of the forms with the progress bar.

**Kristian Tveiten** was selected as the document manager. He was responsible for documenting the meeting minutes with Jesper, documenting the time usage with data from Toggl, keeping backups of both the repository and the report. He also took the responsibility of setting up the main API infrastructure, deployment of the application and the database.

**Nils Olav Tuv** was responsible for conducting tests and improving the application based on this, he also took the responsibility of the CostCalc infrastructure and parameter management within the application. This included the database page where the admin could view, modify and delete parameters stored in the database and used in the whole application.

**Emma Sofie Søvik** had the main responsibility of being the contact person with Benteler for information flow besides meetings, she also took the main responsibility of implementing the formulas and the infrastructure behind the TFC Forms.



### 1.4.2 Earlier experience

Through our studies, we have acquired knowledge about the whole development process. In particular the class Software Engineering (IMT2243)[24], where we got knowledge on the software development process and on tools to use in the different phases of the process. We also got experience in project planning. This is all something we know will be useful and relevant for this project.

In addition to that, we have also gained relevant experience and knowledge on databases, application development, and software security through various other classes.

Besides this, the group also had previous experience in working as a team together during a diversity of projects throughout the study. We have learned a lot of tools necessary for working with such a project and was both confident and excited to exercise our knowledge with this real-life task.

### 1.4.3 What we had to learn

Although we all have earlier relevant experience through our studies, there was still a lot to be learned before, and during, the project. Specifically, the group lacked important knowledge on web development and the deployment phase. Our learning experience and our technology decisions are written about throughout the whole report.

## 1.5 Why we chose this task

The group wanted a challenging and exciting task, that would give us developing experience that could later give us an advantage when entering the job market.

The group agreed that Benteler's Extrusion Planner task fulfilled these wishes and it was immediately put into a list of relevant tasks.

After reviewing the task description and discussing the task with Benteler, we acquired a big interest in this particular task and we thought that it would be a good task to have as our bachelor thesis.

The Extrusion Planner project is a full-stack development project, which is something that we found very exciting and something we thought we would obtain a lot of learning experience from. All four group members wanted more experience in both frontend development and backend development, and this task would give us that. In addition to that, we also liked the freedom in this task and that we would be able to decide what technologies would fit the task the best. We thought this would be a good opportunity to use popular languages and technologies that are relevant for our future as developers, but of course also make decisions based on what would fit the task and Benteler's wishes.

Benteler had many functionality requirements and operational requirements that we found exciting and challenging. We will not only have the opportunity to build and design an application from scratch, but we also will have the responsibility of securing the application. By choosing this project we would not only become better code developers, but would gain experience in testing and the deployment phase as well.

## 1.6 Thesis structure

**Introduction** The reader is introduced to the task and the team, as well as the limitations.

**Development Process** How and when the different parts of the application is developed and what development model is followed to reach the goal of the application.

**Requirements** This chapter contains all the requirements the team gathered from Benteler in the planning phase.

**Technologies** Here the reader can read what tools that were used by the group to solve the various tasks and problems that were faced. Including descriptions on specifically which parts of the used technologies that were of great use in the case of this development process.

**Design and implementation** The reasoning for certain design choices, what, why and how things were done the way they were done to enhance the user experience and make the application as easy as possible to use.

**User Interface** Descriptions and pictures of the different parts of the application and user interface, here you can see the result of the application.

**Overview of the development environments** An overview of the development environments and different tools the team used in the development, divided into frontend, backend and database.

**Code Quality** In this chapter, the different measures taken to ensure a certain quality of the code are presented to the reader.

**Testing** How we decided to test the application and description of the various results.

**Conclusion** Here the reader can read more details about the team's experience and our thoughts at the end of the project. The result is discussed by the team and also includes feedback from the client.

## Chapter 2

# Development process

### 2.1 System development model

Using system development models during development aids in succeeding with development projects that deliver on certain premises and requirements. It helps keep the work organized and structured as well as easily keeping track of what tasks are done, when and how the development process is executed. With this task, there were not a lot of restrictions and desires for how the development process should play out from the client, but still we had criteria we needed to follow.

In the project plan (appendix A), the choice and reasoning behind the selection of the development model is described in more detail.

### 2.2 Project characteristics

Below are the characteristics of the project that were taken into consideration when choosing the development model.

1. Set deadline for delivery
2. Smaller development team of four
3. Regular meetings with the client to receive frequent feedback on the work and new ideas or room for improvement.
4. Functionality and requirements of the system is mostly figured out but there are still room for improvements and ideas from the team/client.

### 2.3 Choice of system development models

The client had presented a couple of core requirements/functionality initially, but it was anticipated that more requirements to the software would come underway as the initial requirements did not go into complete detail.

This meant that choosing an incremental development model was not a viable solution. Meanwhile, an agile development model would allow us to continuously receive suggestions and feedback, and make core changes to the functionality underway.

The team strongly converged towards Scrum, which is a known practice used on earlier occasions. We also evaluated Extreme Programming (XP), for its fast-paced working environment that values teamwork and communication. However, as the client needed documentation of the application, which is lacking in the XP practice, it was ruled out.

In combination with Scrum, we chose to follow the Kanban practice of keeping track of backlog elements and their status in a Kanban board.

### 2.3.1 Following the Scrum and Kanban practice

Combining Kanban with Scrum was an excellent choice for this project, as the Kanban board could be revised/balanced during sprint meetings/scrums, to catch up to tasks that have stayed in a certain phase for too long. This combats congestion of tasks in a certain phase, so those backlog elements are not abandoned.

The team decided that the sprint intervals should be two weeks, as we found out that having shorter intervals than two weeks would be counterproductive. The sprints were set to start on Wednesdays and would end on Tuesdays when we had a meeting with the product owner. Trello<sup>1</sup> was used as a tool to follow the Kanban practice.

#### Scrum meetings with Product Owner

The team had scrum meetings with the product owner after every sprint, where finished software modules were presented and reviewed. From this meeting, we received feedback on the developed software and suggestions for further development which would be added to the backlog. We also conducted meetings with the product owner in the middle of each sprint to discuss the status and provide a way for the client to follow the development process closer during the sprint.

#### Internal scrum meetings

The group also conducted scrum meetings internally 3 days of the week, to review progress and re-prioritize important tasks, change or break down certain backlog elements. On Fridays, the group met internally to discuss the current sprint status and share what each member had been working on with code explanation. This session was also used to share knowledge of new ways of coding and techniques that were discovered during development, this greatly improved the learning process and startup time.

#### Documentation with scrum

Scrum combined with Kanban also allowed us to conduct the documentation routines we needed to complete the project with the report. We just simply added the documentation task in the backlog, so that they could easily be added as a task to complete during a Sprint.

#### Scrum roles

We decided that the role of scrum master should be delegated to the project leader Jesper Trøan. Frode, Gerda and Kristian-Inge from Benteler were the product owners.

### 2.3.2 Time estimation models

Planning poker<sup>2</sup> was utilized as a way for us to estimate the time usage per backlog item. Planning poker is a technique for estimating the effort necessary to complete a software development task. The team members give an estimate each and the estimates are discussed until they are at agreement. At the start of every sprint,

---

<sup>1</sup><https://trello.com/>

<sup>2</sup>[https://en.wikipedia.org/wiki/Planning\\_poker](https://en.wikipedia.org/wiki/Planning_poker)

the team estimated the time usage per backlog item verbally in order to estimate how many backlog items could be assigned to each team member.

## 2.4 Process execution

The project was divided into 6 sprints, and as explained earlier each sprint lasted 2 weeks. To keep track of the process and each sprint's task the team made use of a variety of helpful tools and programs, which is all documented below.

### 2.4.1 Purpose of each sprint

The 6 sprints were distributed evenly over the period February 1. - April 30 as seen in Figure 2.1. During the planning phase the team set up the sprints and started discussing each sprint. We decided to give each sprint a purpose and set a rough plan for each one, to help us keep the process going and to be able to finish the project on time.

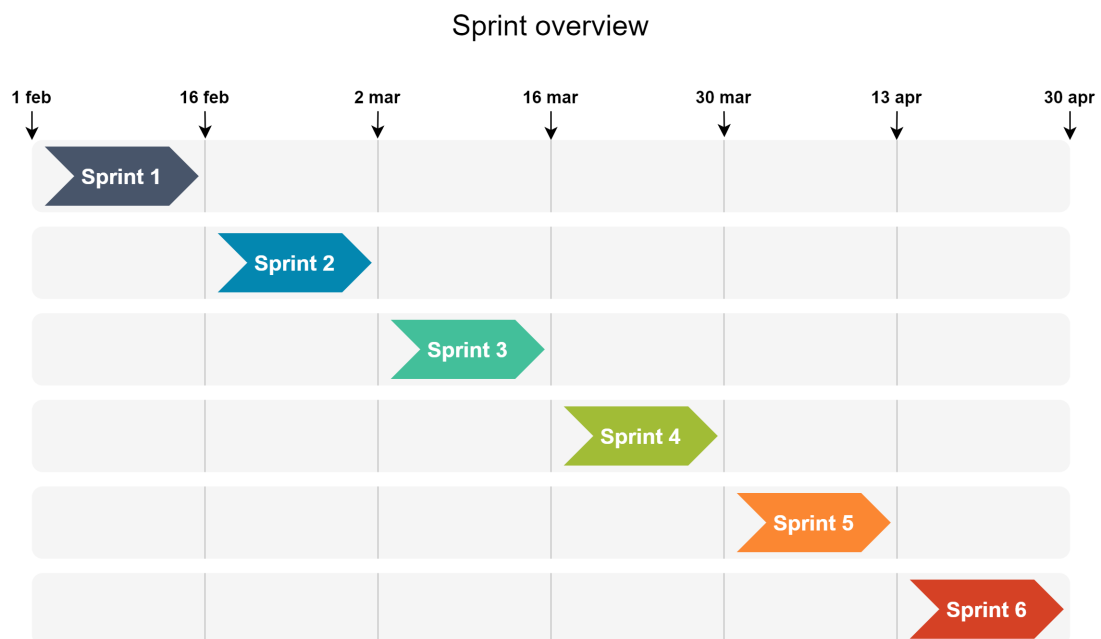


Figure 2.1: Sprint overview

#### **Sprint 1:** Login and main interface components

We decided to dedicate this sprint to creating a solid foundation in both the frontend and backend. Our focus was on the login functionality and to work on authentication/authorization. The foundation for the UI was also a focus, and our goal was to implement a couple of the main components of the application like side-menu, footer and a navigation-bar.

We also wanted to start design and implement the Request form and search functionality, which would give us a good start on sprint 2.

**Sprint 2:** TFC, Request and Order functionality

The focus on this sprint was the core functionality of the application, the TFC, Request and Order functionality. Our goal was to have a good foundation of the three forms, as well as three main pages with different options for the three categories. We knew that this was a lot to implement in both frontend and backend and that it would be time-consuming and challenging. Consequently, our backlog for this sprint was big and we had expectations that a part of the task would be pushed to the next sprint.

**Sprint 3:** Finishing touches on core functionality and cost calculations.

The goal of this sprint was to complete the rest of the important functionality of the application. We knew we had to keep working on the core functionality, the TFC, Request and Order functionality. We also had to implement the cost calculation functionality and other functionalities that might be missing.

**Sprint 4:** Design and Security

In this sprint, the main focus would be the design and to secure the application. We decided not to spend too much time on design early in the project, since the functionality of the application was a priority. The design was therefore pushed aside until sprint number 4 where we had set off time to make finishing touches on the design. Security was also something we thought would be a good thing to focus on later in the project. We planned to implement good security practices from the start, but to have a sprint to go over it and prioritize it towards the end.

**Sprint 5:** Testing

This sprint was set aside to test the application. Our main task was to deploy the first version of the application to let Benteler test it themselves. In the meanwhile, the team would also be running different tests to ensure that the application behaved as expected.

**Sprint 6:** Deployment

In this sprint, we finished all the finishing touches, and are ready to deploy the final version. The application is deployed on Benteler's servers and the database connection is changed from their test database to their main one.

## 2.4.2 Scrum board

The Kanban board (Figure 2.2) was an important tool during the execution of the project, and as mentioned before the Kanban-style list-making application Trello was used for this. Our Trello board consisted of 10 lists: Backlog, Next-up, Current Sprint, In Progress, Quality Assurance, On Hold, Bugs, Done, Ideas and Abandoned Ideas. These lists gave us a constant overview of the project and which part of the process each task was in. Every task was divided into separate cards, and each card consisted of what the task was about, which team member was working on this task, and comments if we needed to add more information.

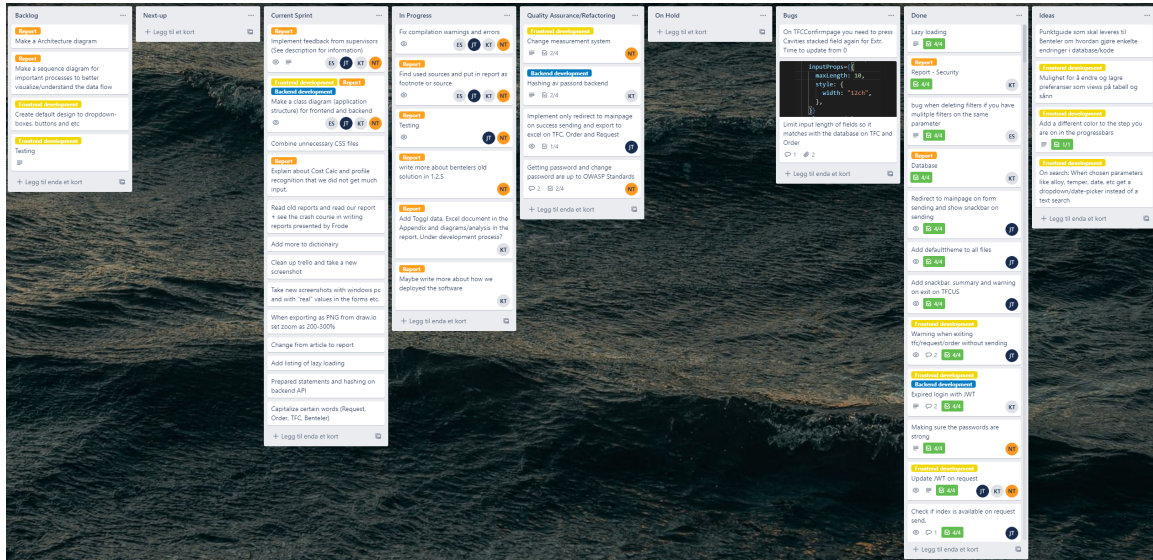


Figure 2.2: Scrum board in Trello

Most of the board follows standard Scrum setup with lists like backlog, in progress and done. In addition, the team decided to add a few extra lists to have an even more detailed process. An example of this is the "Quality Assurance/Refactoring" list, which every task had to go through before moving it further to "Done". This list is written more about in section 8.1. Another example is the "Idea" list, where the member could add new ideas and solutions for the application or the report and it would be discussed at the next sprint meeting.

### 2.4.3 Meeting minutes

At the end of each sprint, we had sprint review meetings with Benteler to show them what had been done during the sprint. In addition, the team had internal sprint planning meetings to discuss what should be prioritized at the beginning of each sprint. The team also had frequent status meetings inside the sprints to discuss how well the progress is developing (as described in 2.3.1). After each meeting it was written a meeting minutes for it in Google docs, and in appendix H you can see the meetings minutes for all the meetings, but in Figure 2.3 we have added an example of a written meeting minutes from a meeting with Benteler.

## 24.03.2021 - Meeting with Benteler

**Agenda:** Discussing US/Metric conversion, show and discuss the overview over previous filled in parameters in TFC.

**Place:** Teams

**Duration:** 40 minutes

**Participants:** Nils, Kristian, Jesper, Emma, Frode, Gerda, Kristian inge.

**Outcome:** Got feedback that we don't have to prioritize implementing saving forms locally because internet shutdown is very rarely to happen. Showed also the latest process. We got feedback that they had envisioned a bit different solution of the summary in TFC, but they were willing to test it out first. Benteler also asked what would happen if two people were working with the same request/index and we agreed to implement a solution where the index number would change and notify the user on send, if someone already has sent in the same request/index combo.

Figure 2.3: Meeting minutes example

### 2.4.4 Mindmanager

The visualization tool Mindmanager [21] was used during the process execution to visualize and bring structure to all the meetings that were held during the project. This was visualized in a mind map with meetings with Benteler, internal meetings, and supervisor meetings as main nodes in the map. The whole mind map can be found in appendix E, but in Figure 2.4 we have added a clip of the structure of a meeting in the mind map. The attendees and agenda are saved in sub-nodes, and in the notes section it is saved a PDF of the meeting minutes of the specific meeting.

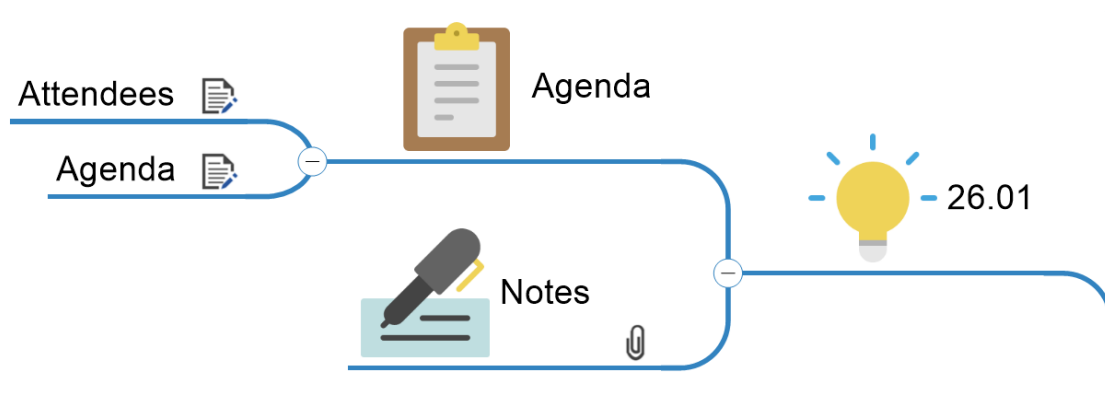


Figure 2.4: Structure of a meeting in mindmanager

### 2.4.5 Project time usage

Executing a bachelor thesis project successfully requires spending a good amount of work hours. To reach our goals within the project, we as a team decided to set ourselves a target of 30 hours per week each, or 6 hours per day working with the project (appendix G). This amount was estimated as the necessary amount of



work required for a subject worth 20 study points. The team utilized Toggl<sup>3</sup> for time registration and tagging with different activity types.

The document manager (Kristian) also coordinated an Excel sheet with the results from Toggl for a better overview of exactly how much time was spent per activity, member, week etc. The full Excel sheet can be seen in appendix I.

Time usage per activity can be seen in Figure 2.5. As seen in Figure 2.6, the team has managed to hover around the 120 hours mark, which is the combined work target for all the group members each week. In the 13th week, the team decided that we could take Easter off, with only a slight amount of work being done during this holiday.

The reason for the low amount of registered testing activity in Figure 2.5, is that most testing has been done underway of developing and separate testing sessions were rarely done. This led to time spent testing being registered as development as this was an activity that went hand in hand.

Testing by the client is also not taken into account, which is a considerable amount of time. Take note that Figure 2.6 only contains data from week 2-19, meaning that all the report work done after this time is not taken into account.

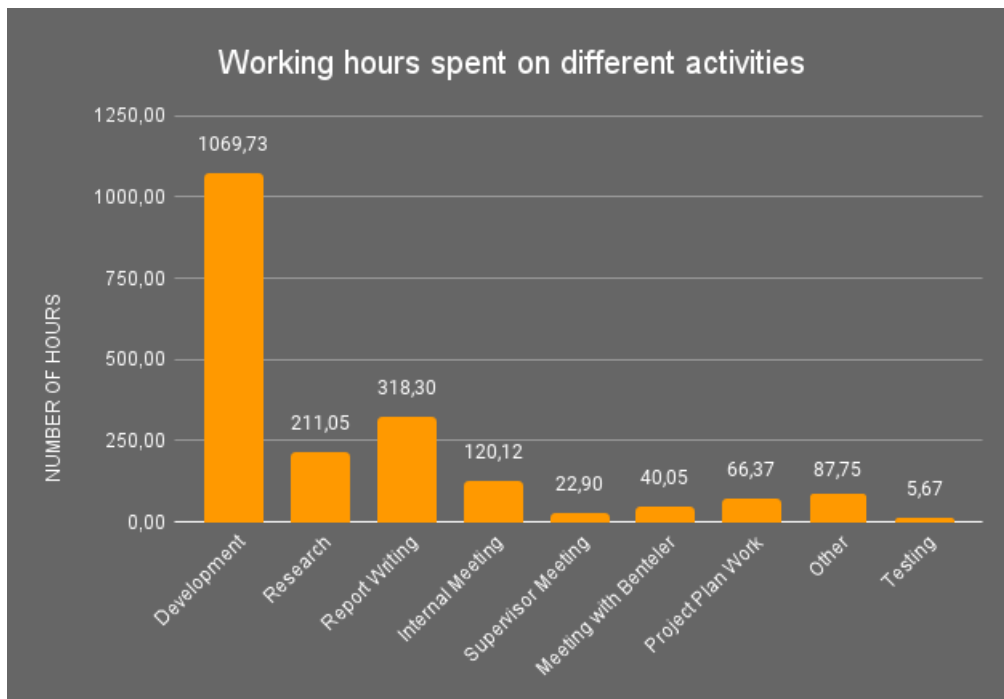


Figure 2.5: Working hours spent on different activities

<sup>3</sup><https://toggl.com/>

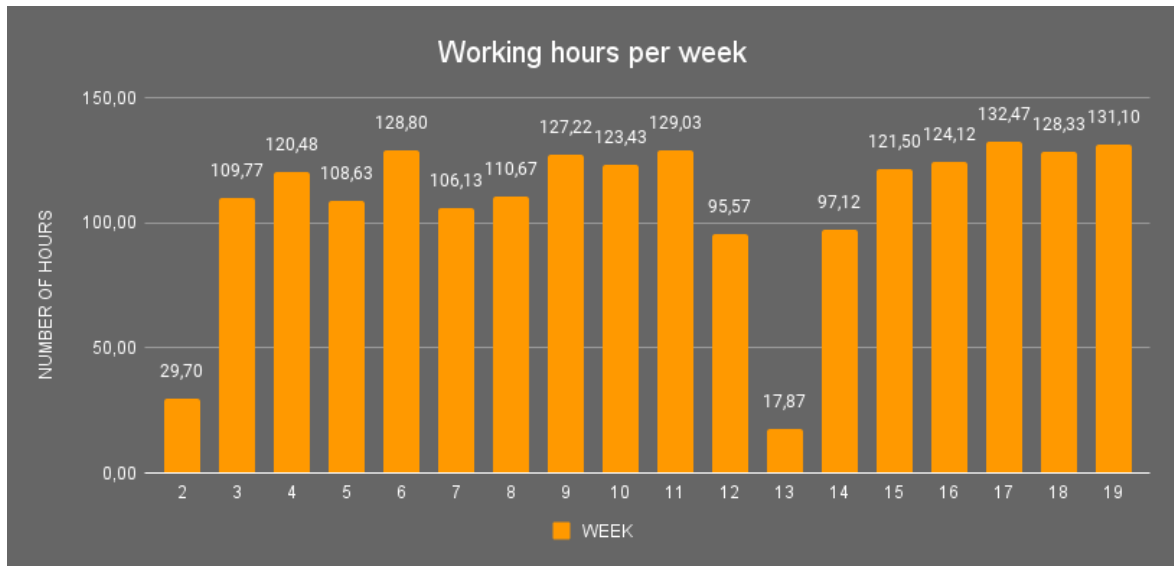


Figure 2.6: Working hours spent per week

The total amount of hours spent on this project was just below 2000 (1941), this results in an average of 114 hours a week in total or 28.5 hours a week per group member if the Easter week is not taken into account. This was really close to our goal of 30 hours a week and we are really satisfied with the effort put into the project by the team as a whole.

## Chapter 3

# Requirements

By using Scrum as our development model we opened up to receive requirements during the project, but it was still important for us to have a rough overview of the most important functionalities and non functional requirements at the start. Because of this, the team had multiple meetings with the product owners in the planning phase to discuss the requirements of the application.

### 3.1 Use Case diagram

To easily distribute the functional requirements per the different actors of the system, the team decided to create a Use-case diagram (see Figure 3.1) to describe the requirements. The users of the system are divided into four different roles with each their own permissions of what can be accessed. Admin has the most access, followed by the only slightly restricted superuser, then CostCalc users and normal users. We chose to explain more complex use cases in detail with alternative scenarios and failed scenarios, where as the simpler ones were only described in a high level (non-detailed) way.

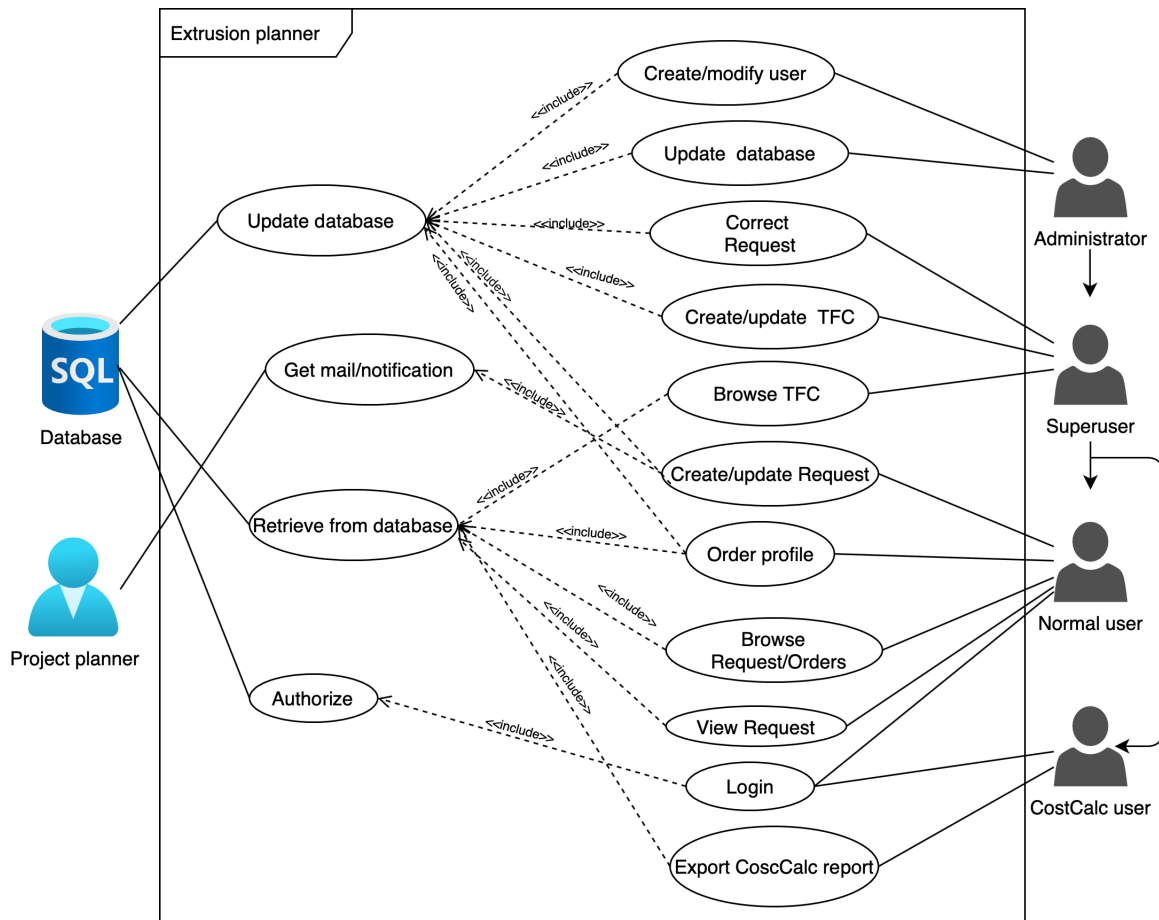


Figure 3.1: Use Case diagram

The arrows between the actors indicate inherited use cases. Where the actors with a higher rank inherit the use cases belonging to lower-ranked actors which is illustrated by the arrows on the diagram.

The actor “Project Planner” is used as an illustration only, in reality, the Project Planner is included in the Superuser group.

### 3.1.1 High level Use Cases

Use Case	Create/modify user
User	Admin
Goal	Add a new user to the system or modify an existing user.
Description	Admin has the opportunity to create new users with a specific role (Normal user, CostCalc-team, superuser etc). A new user will then be added to the system, and a username and password will be added to the database. In addition to that the administrator also has the opportunity to modify the existing user, this means updating the user information or deleting a user from the system.

Use Case	Update Database
User	Admin
Goal	Modify, add or delete properties in the database.
Description	Admin should be able to modify, add or delete properties and parameters in the database.

Use Case	Correct Request
User	Superuser, Admin
Goal	Correct existing Requests
Description	Superusers and admin have the opportunity to change a Request without creating a new Request number or version number, this is used to correct simple mistakes made by the requestor. The superuser and admin should also have a way of deleting a Request or making it invalid.

Use Case	Order Profile
User	Normal user, Superuser, Admin
Goal	Create an order.
Description	The user should be able to create a new Order by inputting the required information in an Order form. The user should be able to open an empty form or choose the specific Request and TFC data to fill parts of the form. The user should also be able to update an existing Order by choosing an Order to insert into the form.

Use Case	View Request
User	Normal user, Superuser, Admin
Goal	View a specific Request
Description	Users should be able to view a specific Request with all its parameters.

Use Case	Log in
User	CostCalc-team, Normal user, Superuser, Admin
Goal	Log into the system.
Description	Users can gain access to the system by logging in with a correct user-name and password which has been registered in the system.

Use Case	Export CostCalc report
User	CostCalc-team, Superuser, Admin
Goal	Export the relevant data to the CostCalc report
Description	The CostCalc team should be able to search for TFC and export the chosen data to a CostCalc report.

### 3.1.2 Detailed Use Cases

Use Case	Create/update TFC
User	Superuser, Admin
Goal	Create a new TFC or change an existing one.
Description	<p>The user should be able to create a new TFC by inputting the required information in a TFC form. The user should be able to open an empty form with only the selected Request data or to open a form with data from a chosen existing TFC. The new TFC should receive a new TFC number (unique with the Request number) and should receive 1 as the version number.</p> <p>The user should also have the option to update/change the values of an existing TFC. A new version with the same TFC number and new version number is then created.</p>
Pre-Conditions	User is logged in with enough permission and a Request for the profile has been made
Post-Conditions	A complete TFC is made with legal values which makes the product producible. The TFC is then sent to the API and saved in the database
<p><b>Main Success Scenario</b></p> <ol style="list-style-type: none"> <li>1. The user selects TFC in main menu, then clicks the TFC search tab</li> <li>2. The user selects a desired TFC to base a new on using the search interface</li> <li>3. The user clicks on “Create new TFC based on a earlier TFC”</li> <li>4. The application displays a selection window to choose what Request should be accompanied with the TFC.</li> <li>5. User clicks a desired Request.</li> <li>6. All appropriate values are pulled from the database for the chosen TFC and Request and inserted into the TFC form.</li> <li>7. Meeting commences, and the rest of the values are finalized or modified.</li> <li>8. All values are within legal thresholds, and the user clicks “Send TFC to database”</li> <li>9. TFC is successfully sent to the database.</li> </ol> <p><b>Alternative Scenario</b></p> <ol style="list-style-type: none"> <li>1a) The user selects the Request search tab             <ol style="list-style-type: none"> <li>1. The user selects the desired Request</li> <li>2. The user selects "Create a new TFC from Request"</li> <li>3. The user is presented with the form pre-filled with Request data</li> <li>4. The TFC meeting commences and the rest of the data is filled in</li> <li>5. The user submits the TFC to the database</li> </ol> </li> </ol> <p><b>Failed Scenario</b></p> <ol style="list-style-type: none"> <li>9a) The database/backend is not reachable             <ol style="list-style-type: none"> <li>1) User gets a message saying the database is not available, and an option to save the TFC as a draft locally.</li> </ol> </li> </ol>	

Use Case	Create/update Request
User	Normal user, Superuser, Admin
Goal	Create a new Request or change an existing one.
Description	The user should be able to create a new Request by inputting the required information in a Request input form/page. The user should be able to open an empty form or choose existing Request data which should be inserted in the form. The new Request should receive a new unique Request number and start with version number 1. In addition to creating a Request the users should also be able to change one. The user should then be able to choose a Request and change specific values on that Request in the form. A new version of that Request is then created.
Pre-Conditions	User is logged in with enough permissions
Post-Conditions	A complete Request is made, which is then sent to the database.
<p><b>Main Success Scenario</b></p> <ol style="list-style-type: none"> <li>1. Use Case "Log in" is executed successfully</li> <li>2. The user navigates to the Request page</li> <li>3. The program shows an interface for searching for existing Requests.</li> <li>4. User selects a Request</li> <li>5. User clicks on "New with old as base"</li> <li>6. A Request form is displayed with the parameters from the corresponding base-Request already filled in from the database.</li> <li>7. The Request gets a new Request number and the user changes necessary parameters.</li> <li>8. User clicks send.</li> </ol> <p><b>Alternative Scenario</b></p> <ol style="list-style-type: none"> <li>5a) The user clicks on "New Request from scratch" <ol style="list-style-type: none"> <li>1) An empty Request form is displayed.</li> <li>2) The user fills in all desired and needed parameters</li> <li>3) User clicks send.</li> </ol> </li> <li>5b) The user clicks on "Change Request" <ol style="list-style-type: none"> <li>2) Request form is displayed with the parameters from the corresponding base-Request already filled in from the database.</li> <li>3) The Request gets a new version number and the user changes the desired and allowed values.</li> <li>4) User clicks send</li> </ol> </li> <li>5c) The user clicks on "Recalc" <ol style="list-style-type: none"> <li>1) Request form is displayed with the parameters from the corresponding base-Request already filled in from the database.</li> <li>2) The Request gets a new version number and the user changes the desired and allowed values.</li> <li>3) User clicks send</li> </ol> </li> </ol>	



Use Case	Browse TFC/Request/Order
User	User, CostCalc, Superuser, Admin
Goal	Search for a desired TFC/Request/Order document
Description	The user should be able to search for the TFC/Request/Order categories they have access to. The user should be able to search for specific data by choosing which category and which parameter to search for. Having multiple search terms at once as filters should be possible. With the search function it should also be possible to export the desired data.
Pre-Conditions	The user is logged in and there is TFC/Requests/Order available in the database.
Post-Conditions	The user views the desired TFC/Requests/Orders data.
<b>Main Success Scenario</b> <ol style="list-style-type: none"> <li>1. User navigates to the search page.</li> <li>2. An interface for searching is presented</li> <li>3. User chooses between: <ol style="list-style-type: none"> <li>a) "Search for TFC"</li> <li>b) "Search for Request"</li> <li>c) "Search for Orders"</li> </ol> </li> <li>4. The table of the chosen category is presented.</li> <li>5. User types in the search criteria per a selected parameter to search by</li> <li>6. All Requests/TFC/Orders matching with the searching criteria are displayed,</li> <li>7. User wants to download data: <ol style="list-style-type: none"> <li>a) User selects a specific row to download.</li> <li>b) User downloads a file with the whole search results.</li> </ol> </li> <li>8. A file is downloaded to the users computer</li> </ol>	

### 3.2 Non functional requirements

After discussing the initial design proposed by the team, the client wanted us to follow a certain Benteler design guideline. This was a guideline provided by Benteler which would make the design comply with their existing web-sites or services. This enhances the user experience as they navigate something that seems familiar to the target group of the system.

### 3.3 Operational requirements

After questioning the client, the team landed on a requirement that the application should handle at least 20 simultaneous users as this was the initial number of users of the application.

### 3.4 Security requirements

When it came to the security requirements of the system, the team had to evaluate what would be necessary for the system. This was due to the lack of requirements obtained from the client, most probably due to the lack of security experience. Even though we did not receive any specific security requirements from Benteler, we still wanted to make sure that the application was safe and followed the most important security standards. Because of this, the team decided to contact the software security lecturer Shao-Fang Weng, to discuss our security needs in detail and to receive tips on what we should prioritize.

The team decided to follow the most important security principles to prevent malicious activity. The biggest focus was on authentication and authorization since this could lead to malicious activity on the application if not done right. The deployment server that was supplied by the IT team at Benteler was restricted in the form of only allowing access from within the Benteler internal network or by using a Benteler VPN. This meant that outsiders could not access the application without being employed or having access to the VPN which in itself enforced strong authorization requiring a dynamic pin code/password every time. Having this level of security relieved some of the importance of having certain security measures in place. The in-depth security measures and implementations can be read about in chapter [5.2.7](#) and [5.3.6](#).

## Chapter 4

# Technologies

The client did not propose any predefined requirements for which technologies and tools to solve the task with. Therefore, the team had to evaluate the different technologies available and find out what would be the best fit to solve the task. In this chapter we will discuss our reasoning behind the choices of technologies we made and why we chose to abandon others.

### 4.1 The application type

The first thing we had to consider was the type of application the solution should be. We knew that their older solution was a desktop type application, that was spreadsheet-based and created excel documents. We found this solution not only to be slow and inefficient, but also not user-friendly and poorly designed. We then had to consider what type of application we wanted to create and what type we found the most fitting for the task. We ended up having two choices, a desktop application and a web application and we thoroughly considered them both. In [Table 4.1](#) you can see an overview of the pros and cons for the two choices, and below the table there is a more detailed description of the two options.

Table 4.1: Overview over pros and cons for desktop vs web application

	<b>Desktop application</b>	<b>Web application</b>
<b>Pros</b>	<ul style="list-style-type: none"> <li>• Earlier experience</li> <li>• Not so dependent on internet speed</li> <li>• Performance</li> </ul>	<ul style="list-style-type: none"> <li>• Popular and high demand</li> <li>• No need to download software</li> <li>• Easily accessible</li> <li>• Not hardware dependent</li> <li>• More easily expand the software</li> <li>• Easier to manage and update</li> <li>• Great learning opportunity</li> </ul>
<b>Cons</b>	<ul style="list-style-type: none"> <li>• Must be downloaded and installed</li> <li>• Hardware dependent</li> <li>• More challenging to maintain and update</li> </ul>	<ul style="list-style-type: none"> <li>• More dependent on internet speed</li> <li>• Lack of experience</li> <li>• Sacrifice a bit of development time to research</li> </ul>

### **Desktop application**

The team already had earlier experience within desktop application development. The members all had the elective subject Application Development (IMT3281)[25] where we completed a couple of desktop application projects. Considering this the group found a desktop type application to be fitting and at the start it was our primary choice.

One big advantage of developing a desktop application would be the teams earlier experience. In addition to that desktop applications has several advantages compared to web applications. A desktop application is very easily accessible after download, as it is on the users computer and the icon can be visible at all times. Another advantage is the performance, a desktop application does not have to be dependent on internet speed.

However, after doing more research and discussing the requirements with the client we started considering a web application more. We found that the pros of a desktop application were not as relevant for this particular project, especially because the solution would be dependent on the internet either way because of the database connection. Besides that a big disadvantage is that the application would have to be downloaded on every users computer, this makes it harder for Benteler to extend the use of the application and allow more employees to download it. Another disadvantage of desktop applications is that it is hardware dependent. Managing and updating a desktop application is also a bit more challenging than doing it for a web application, and that is also something we had to consider.

### **Web application**

Web-based application has grown in popularity and has the highest demand at the moment. With a web application the user does not need to download anything on their computer, and they can easily access the application from a web browser wherever and whenever they want to. A web application is not hardware-dependent like a desktop application, and the team does not have to consider the software of the client when developing it.

However a web application is dependent on an internet connection, but as mentioned earlier the solution would have to use an internet connection anyway. Another thing to consider with a web solution is the teams lack of experience within web development. We knew that if a web solution was chosen we would have to spend a period of time learning about web development and researching tools. As a consequence of this we would have to sacrifice some of the development time just to learn and research, this meant that extra tasks like profile recognition would be pushed aside and only considered if we had extra time in the end.

A web application would make it easier for the client to give more employees access to the application. We also figured that a web application would be easier to manage and update if necessary. It is also a great learning opportunity for the team, as web development is dominating the software market at this moment. Consequently it is important to gain experience within this type of programming.

The team and the client discussed the pros and cons of web and desktop application and concluded that a web application would be the most fitting solution for the project. The team made it clear for the client that they lacked the experience, but also that we were confident that we would be able to create a good solution if we reserved extra time for learning and researching at the start.

## 4.2 Frontend

The Extrusion planner project is a Full-Stack project, which means that it consists of both frontend- and backend-development. Considering the lack of web-development the group had, frontend was something the group had to learn from scratch. In the research period, the group started learning and researching HTML, CSS, and JavaScript. We read multiple articles and tutorials on frontend development to obtain a basic understanding of frontend-development. W3School<sup>1</sup> was a resource the group used frequently to quickly gain basic knowledge on the languages in frontend development.

### 4.2.1 Resources used

Early in the research period, we figured out that we had to decide on what frontend framework to use. A frontend framework would help us create a consistent and intuitive UI, while also helping us create maintainable code and prevent spaghetti code[26].

We quickly realized that the decision was between three JavaScript frameworks, Angular<sup>2</sup>, Vue<sup>3</sup> and React, because these are the three most widely used framework and we wanted to choose something with high demand. After research on the different frameworks/languages and consultation from a developer who has worked with them all, the choice fell quickly on React. The decision was made due to Reacts versatility and flexibility with components, state handling and way of rendering the web page dynamically. React also uses JavaScriptX instead of Angular's TypeScript, which means we would be developing with a more widely used script language, which presumably will lead to better support online for problems and making a concept happen [10].

---

<sup>1</sup><https://www.w3schools.com>

<sup>2</sup><https://angular.io>

<sup>3</sup><https://vuejs.org>

## 4.2.2 Choice of React UI framework

As for components that are repeated often in the application, we wanted to follow a user interface library, so we got a common thread through the whole application.

Early in the project, we discovered a library called Material UI [34]. Material UI is an open-source project that features React components that uses Googles material design. One of the things that makes material UI fit to this project is that it offers great customization to for example text-fields that we use in a big part of the application. Another thing that makes the use of Material UI great in this application is the we can create default themes that are used on all the components. This makes it easier for us to implement the primary and secondary colors of Benteler over the whole application and make the application better looking. Another advantage is that Google maintains Material design and keeps documentation for how to use and implement it. This makes it easier for us to apply it to our project. After comparing Material UI to other user interface libraries and a discussion in the group we found out that Material UI would fit our project better than other libraries like bootstrap because of the way the application would look and its practicality in use.

As the team wanted the best possible learning outcome it was decided to design certain components ourselves so that we learned to use a popular UI framework, but also learned to design components from scratch using HTML, JavaScript and CSS.

## 4.3 Backend

### API

For the backend, we needed something versatile that delivers a lot of functionality in pre-made libraries to make the API solution easy to create, maintain and develop. Due to us choosing a new framework/language for the frontend which requires learning, we decided to look into Java first.

The team has six months of experience in Java coding, as well as having courses about object-oriented programming in earlier semesters, which easily translates into Java. We quickly found that Java was a very popular backend/API framework, due to a library called "Spring Boot" <sup>4</sup>.

### Database + Management Software

When researching which database server/management system to go for, the criteria was to find something which is easy to use and delivers powerful software for managing the database.

HeidiSQL<sup>5</sup>, phpMyAdmin<sup>6</sup>, MariaDB<sup>7</sup> and others were explored until we stumbled upon the tried and tested Microsoft SQL Server 2019<sup>8</sup> in combination with Microsoft SQL Server Management Studio. This turned out to be the best package in terms of usability, familiarity and easy integration with the API as the online support is massive, with corresponding drivers readily available.

---

<sup>4</sup><https://spring.io/projects/spring-boot>

<sup>5</sup><https://www.heidisql.com>

<sup>6</sup><https://www.phpmyadmin.net>

<sup>7</sup><https://mariadb.org>

<sup>8</sup><https://www.microsoft.com/en-us/sql-server/sql-server-2019>

### **Testing Server**

To run the backend, we had to obtain a server that would allow us to set up the software/system like it would be deployed early on. This would make it much easier to develop the software as the database and backend functions would be unified.

The team requested a development server from Benteler early on, but as the development started we decided to consult NTNU to obtain a server through the SkyHiGH service until we got access to the client's server. The team requested a server with two CPU cores and 4 GB of RAM, which were estimated to be sufficient for the purpose of testing. On this server, we installed MS SQL Server 2019, SSMS, IntelliJ and Git. Git was used to pull new revisions from GitLab, and IntelliJ was used to quickly run the API server without having to build a new JAR.

### **Deployment Server**

As of the 11th of March, we obtained access to the client's own web server, where they pre-installed a couple of the required software like SQL Server and SSMS. They also requested a walkthrough/recipe on how to set up the server from scratch, with all our settings, dependencies, data, software and connections used. Due to earlier experience where older software would stop working due to the developers using a dependency that stopped working and deemed the software unusable.

The team also proposed to make documentation on how to do changes to the database content and update the SQL queries in the backend, in case they would like to add or remove certain columns in existing tables after we were done with the project. This was appreciated and a document explaining how to edit the application code and re-deploy the application was supplied by the team, see Appendix B. The backend will need to be rebuilt in that case, which required thorough documentation as it is not always a straightforward process.

## **4.3.1 Resources used**

### **Spring Boot**

Spring Boot is an incredibly powerful platform for getting an API up and running in no time. It transforms the project into a stand-alone server and delivers easy ways to define API endpoints, and HTTP request methods.

Spring Boot also lays the foundation for a stable server, as every HTTP request is treated independently and does not affect the well-being of the server if something goes wrong with one request.

### **Microsoft SQL Server 2019 + SSMS**

As stated earlier, this combination turned out to be the best package in terms of usability, familiarity and easy integration with the API as the online support is massive, with corresponding drivers readily available.

The decision was later reinforced when we got information about the client's current database which used Microsoft SQL Server 2012. This made importing tables straightforward when we got sent their test database 11. February.

### 4.3.2 Learning material

Lecture style YouTube tutorials were widely used as it was very helpful in learning new language concepts and ways of doing certain things. This helped a lot with learning both React and the Spring Boot API from scratch.

A book about Java programming was also obtained for reading, but we used it in a limited fashion due to the fact that the learning required was mostly about Spring Boot[9]. To understand the Spring Boot API, we used the API documentation<sup>9</sup> as well as YouTube tutorials.

We used the Software Engineering course book[24] to re-visit and refresh our knowledge about the system development process in order to have a professional approach to the project.

Naturally, StackOverflow<sup>10</sup> was also used to search for different ways of solving issues or ideas we had during the project.

To learn about how to deal with security in React, an React security course[7] was taken by all the team members to better understand what to implement.

## 4.4 Testing

This section contains an overview of the testing tools we used to test the application, to read about the tests and the results check chapter 9 Testing.

### Node/npm

To run the frontend locally we used Node [12] and npm [23], in order to launch a development test server, which automatically opens the website and refreshes every time the code in the IDE is saved. This saved a lot of time, due to being able to see results immediately.

This solution provided us with an automatic code-linter as well, which checked the code for errors and warnings every time it was saved. It then displayed errors when the build failed or existing warnings when the build compiled, for example if there were unused dependencies or certain code practices/standards that were not followed correctly.

### Postman

In order to test the backend early on, the team decided to utilize Postman[29], which is an API testing service, where the user can create and save certain HTTP requests using different methods, and see the status result of the request, as well as the result, returned from the API. It allowed us to test changes in the API quickly without having to update the frontend accordingly to see the result.

### Lighthouse (Google Chrome extension)

Lighthouse[35] is an extension in the web browser Chrome made by Google. It delivers the ability to run automated performance tests on a given web page, and then generate a performance result that pinpoints what elements or pages are slowing the page down. This made it easier to improve the overall performance of the application.

---

<sup>9</sup><https://spring.io/projects/spring-boot>

<sup>10</sup><https://stackoverflow.com/>



## Chapter 5

# Design and implementation

In this chapter we will be presenting the ways we implemented certain functionality and go in-depth in both the code and structure. The reader should obtain a clear understanding of how all the parts of the system work together, as well as how the functions responsible for a functionality works to deliver the expected result.

### 5.1 Application structure

As we chose to structure our application after the MVC architecture (see Figure 5.1), we could split our application into different parts.

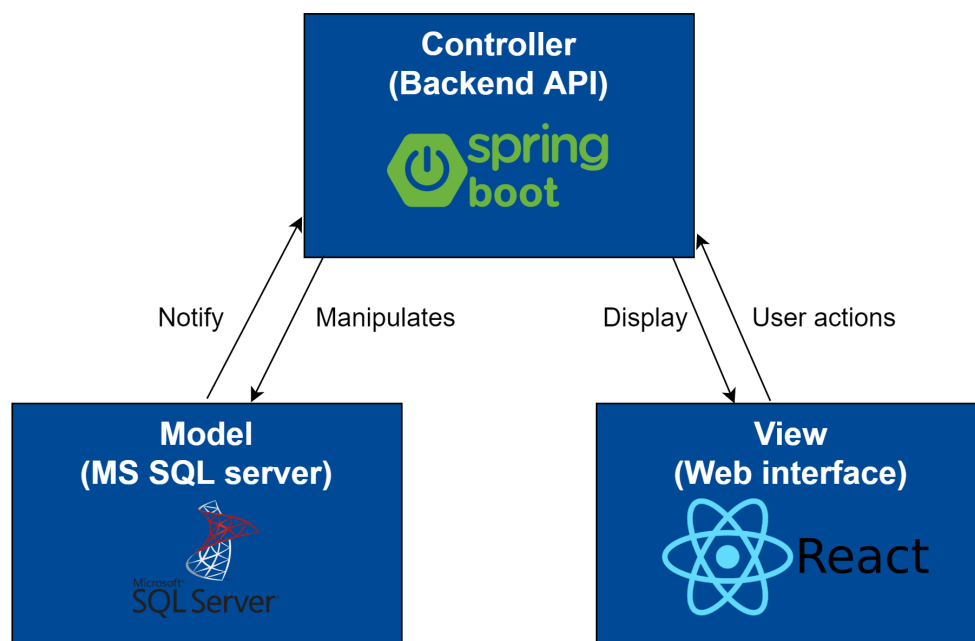


Figure 5.1: MVC pattern of the system

The application structure is illustrated in Figure 5.2. First, we have our frontend web application coded in React. This is what the user sees and where the interaction with the application happens, it is considered a thin client as all information is fetched server-side and nothing besides cookies is stored client-side. The interface is delivered with Microsoft Internet Information Services on the web server. When IIS receives a request on port 3000, the interface is returned to the client using Lazy Loading. [31]

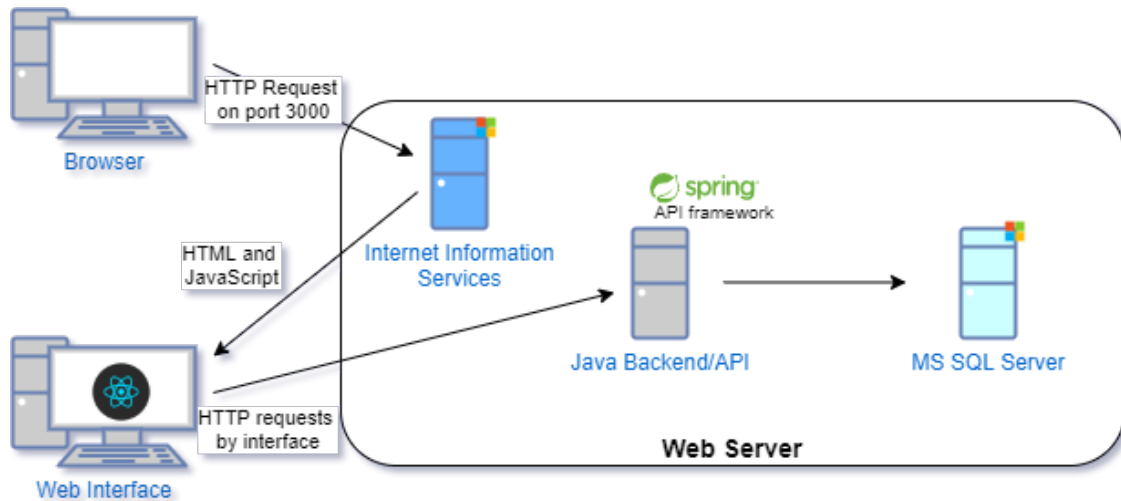


Figure 5.2: Main Application Structure

The frontend interface then communicates with the backend Java application, where we utilized the Spring Boot framework. This powerful framework provided an easy way to make API endpoints in order for the backend to host a range of API endpoints reachable by HTTPS methods: GET, POST, PUT and DEL.

The backend does the heavy work in the application by tying the frontend and database together by logic consisting mainly of SQL Queries for fetching, manipulating and deleting data in the database. In addition, the backend handles security and authentication using JSON Web Tokens.

For storage of the application's data we utilized a Microsoft SQL Server, chosen for its familiarity and easily available JDBC drivers.

Figure 5.3 shows an overview of the code the web application is written in. Most of the code is written in JavaScript. This includes most of the web application, and is involved in making the web application interactive. 18,99% of the code is written in Java. This includes the whole backend API with the Spring boot framework. The rest small percentage of the code is written in CSS and HTML. We have also added the total hours of development work and total lines of code in the figure, to give an impression of the scope of the coding work.

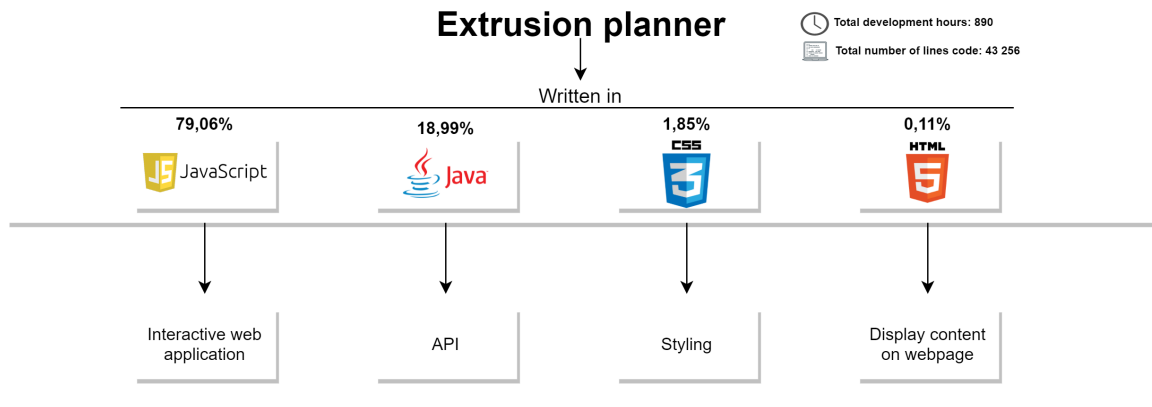


Figure 5.3: Overview over the code

### 5.1.1 JSON

For requests between backend and frontend we have used the text-based format Javascript object notation (JSON). This is a standard used for transmitting data in web applications<sup>1</sup>. We have chosen this standard because we thought it was an easy to use standard and it is a popular format that many developers swear to.

When we are sending large amounts of data from our web application we use the JavaScript function `JSON.stringify()`<sup>2</sup>. This is attached to the body of the fetch function and then sent to the backend. You can see in section 5.3.2 for more about how the server processes these requests.

Listing 5.2 contains a response from a request to fetch a specific Request based on Request number and Request version number. This request have been sent with Axios. Axios<sup>3</sup> is a promise based HTTP client for the browser and Node.js and is used to send HTTP requests to REST endpoints. In listing 5.1 Axios will perform a GET request that has the intention to fetch a specific Request with Request number and Request version number as parameters. This function returns the response from the GET request and we are saving the response by setting variables to the response of the HTTP request.

```

1 /**
2  * Function to retrieve a specific Request based on its id and version
3  * @param {Int} id the id of the request to fetch
4  * @param {Int} ver the version of the request to fetch
5  * @returns {JSON} Request data
6  */
7 getRequestByIDandVer = (id, ver) => {
8   return axios.get(API_URL + "request/${id}/${ver}");
9 }

```

Listing 5.1: Axios GET

<sup>1</sup><https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>

<sup>2</sup>[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/JSON/stringify](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify)

<sup>3</sup><https://www.npmjs.com/package/axios>

```

1 {
2   "Email_requestor": "test@test.com",
3   "Product_controller": "test@test.com",
4   "Request_date": "2021-04-26",
5   "Due_date": "2021-04-02",
6   "Sketch_no": "4",
7   "Sketch_rev": "1",
8 }

```

Listing 5.2: Response from GET request

## 5.2 Frontend web-interface

As for the frontend web interface we have chosen a layout that consists of (Figure 5.4):

- Topbar
- Side menu
- Main area
- Footer

Request Number	Sketch Number	Sketch revision	Request Name	Customer	Project Name
7993	54		testsimultaneous	BAS	34
7993	54		testsimultaneous	BAS	34
7994	1		Navn Navnesen	43	34
7994	1		test	43	34
7994	1		test2	43	34
7982	006-21		Testing2	Audi	AU40x
7982	006-21		Testing2	Audi	AU40x
7982	006-21		Jiri.Yo@benteler.com	Audi	AU40x
7982	006-21		Testing3	Audi	AU40x
7982	006-21		Jiri.Yo@benteler.com	Audi	AU40x
7995	006-21		testsackbar	Audi	AU40x

Figure 5.4: Layout of the application

The frontend web-interface was developed in React as described in 4.2.1. The main area of the application (as seen in Figure 5.4) is the place for the main content of the application which varies depending on which page the user is on, the components Sidemenu, Topbar, and the footer stays consistent and does not change when navigating through the application.

As for basic navigation around our application we have used React's own react-router DOM library. This allows us to easily navigate around the application by defining paths to different pages in the application, see listing 5.3. These paths are defined in the App.js file. This is the main component of the app and acts as a container for all other components in the application. The routes are wrapped inside a switch that looks through the routes and renders the ones that match the current URL. In addition to that, we also used the routes to ensure that the user is authorized for the different pages, this is written about in 5.2.7 Authentication & Authorization.

```
1 /* Sets the path to each page */
2 <Switch>
3   <AdminRoute path="/database">
4     <Database />
5   </AdminRoute>
6   <AdminRoute path="/users">
7     <Users />
8   </AdminRoute>
9   <Route path="/FAQ">
10    <FAQ />
11  </Route>
12  <Route path="*">
13    <NoPageFound />
14  </Route>
15 </Switch>
```

Listing 5.3: Example of Routing

This App.js file is then rendered in the index.js file (see listing 5.4). Index.js stores the main render call from ReactDOM, imports the App.js file and tells React to render it inside the root div which is located in the index.html file.[10] The ReactDOM.render() function therefore takes two arguments, HTML code and an HTML element which in our case is the root div in the index.html file.

```
1 /**
2  * Function that tells React to render inside the root div
3  * @param {app} The HTML code to render
4  * @param {document.getElementById("root")} Where to render the HTML
5  */
6 ReactDOM.render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>,
10  document.getElementById("root")
11 );
```

Listing 5.4: ReactDOM render function

In listing 5.5 we see that the index.html file includes a main div that the React app will show up inside. This is the source of the React app.

```

1 /* The main HTML file with the root div */
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <title>Extrusion planner</title>
6   </head>
7   <body>
8     <div id="root"></div>
9   </body>
10  </html>

```

Listing 5.5: Index.html

Figure 5.5 sums up the structure of the frontend.

1. The browser sends http requests to the server which then retrieves the index.html page.
2. This html page contains the main div that the index.js renders to.
3. The render function renders the app.js file which routes the component that matches with the current URL.

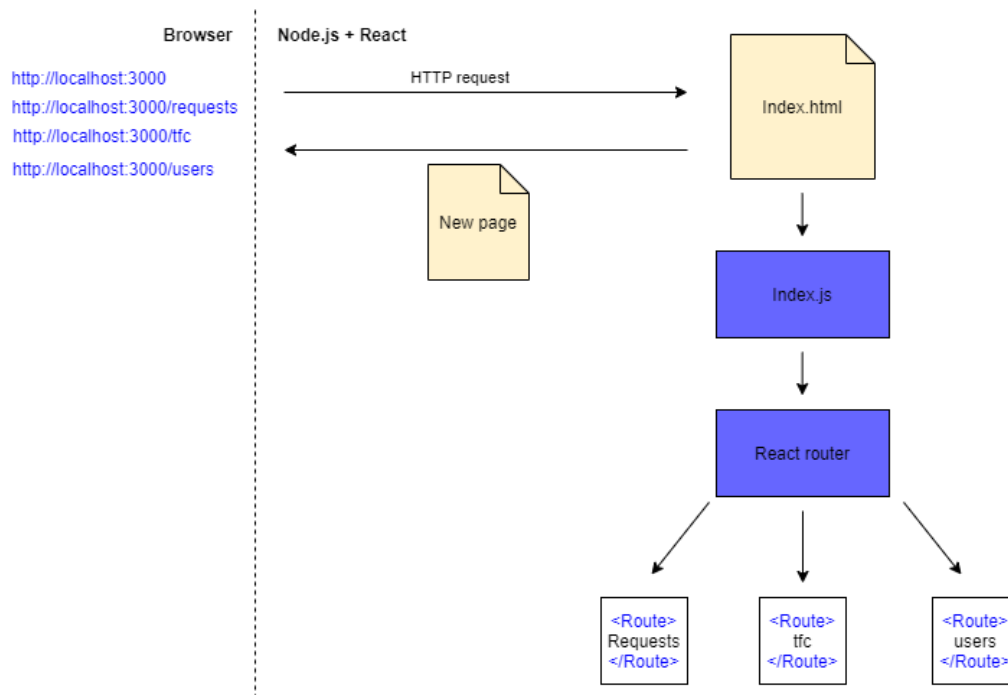


Figure 5.5: Frontend structure

## 5.2.1 Design

For the components that were often repeated throughout the application, we used the popular Material UI framework as described in 4.2.2. This includes all the buttons, text-fields and popup menus. These components came with a standard styling, but they were also highly customizable. For the Material UI buttons, we did not need to change anything other apart from the size and the color of the button. The color of all the buttons was easily customizable by creating a MUI theme and setting the color of the button to primary or secondary.

The text-fields were also easily customized. Here we could set different properties inside the element tag `<>`, but also overriding the style of the component by using class names. We could use `makeStyles()` to inject custom styles to the element and pass the class name to the `className` of the specific component [34]. In listing 5.6 we have created a style like this.

```

1 const useStyles = makeStyles((theme) => ({
2   root: {
3     "& .MuiTextField-root": {
4       /* Sets the default spacing between the textfields*/
5       margin: theme.spacing(4),
6     },
7     "& .MuiInputBase-root": {
8       color: "black",
9     },
10  },
11  /*Declares a own style "highlighted" that creates a border around the textfields
12   that gets this style as classname*/
13  highlighted: {
14    "& .MuiOutlinedInput-root": {
15      border: "6px solid yellow",
16    },
17  });

```

Listing 5.6: MakeStyles

The rest of the components we designed from scratch by ourselves. This includes among other things the progressbar, the top navbar, the sidemenu and the tables in the search pages. This was done with CSS. In listing 5.7 we can see a small part of the CSS code of the progressbar. This code colors the progressbar as the user clicks forward and backward in the form. This is done by setting the background color to Benteler's secondary color orange (Figure 6.1) when the `className` of an element in the progressbar is set to active.

```

1 /*Colors the main circle elements when a page is clicked, and sets shadow*/
2 .progressBar li.active:before {
3   background-color: rgb(228,108,10); //test
4   box-shadow: 0 3px 5px 2px rgba(0, 0, 0, .25);
5 }
6 /*colors the line after the main circle elements when a page is clicked*/
7 .progressBar li.active + li:after {
8   background-color: rgb(228,108,10);
9 }

```

Listing 5.7: CSS progressbar

## 5.2.2 Setup of the forms

When it comes to the setup of the forms we chose to develop it as a multi-step form [20] with a clickable progressbar where the user easily can navigate back and forward. The multi-step form was developed by making a default form page which contains a switch that switches between the different pages in the form (see listing 5.8). This default form page also includes all the parameters for the form, functions to set the number of the step, a function to handle the change of the values and more.

```

1 /**
2  * Switch function to switch back and forward in the form based on the number of the
3  * step
4  * @param step the number of the step
5  */
6 switch (
7   step
8 ) {
9 case 1:
10    return (
11     <Order
12      parameters...
13    >);
14 case 2:
15    return (
16     <Order2
17      parameters...
18    >);
19 ...
20 }

```

Listing 5.8: Structure of a form

## 5.2.3 Functions in TFC

When developing the Request- and Order-form we used the ES6 class component to define the components in the forms, see [16]. This class components use states to declare data, and the component re-renders every time the state gets updated. When these two forms were finished developed, we discovered React Hooks [17]. The team then decided to develop the TFC form and the rest of the application using React Hooks as we thought this was an easier and more modern way of coding in React. React hooks lets us use states and other React features without declaring a class. When we are declaring a hook we declare the value of the state, and a function to update it. This can be seen in listing 5.9. The function to update the hook is similar to the this.setState function in the class component.

```

1 /*State for reduction ratio*/
2 const [reductionRatio, setReductionRatio] = useState(tfcValues.Reduction_ratio);

```

Listing 5.9: React hooks

The TFC form consists of many functions that have the task to calculate different values. This is done by using Reacts useEffect hook [18]. The useEffect hook adds the ability to perform side effects when one or more states are updated. The useEffect hook runs after every render, but with a dependency array, the hook can be customised to run when the variables in the dependency array is updated. In listing 5.10 we are using the dependency array to calculate reductionRatio each time the profileArea, containerArea or tfcBufferValues are being updated.



```

1  /**
2   * Function to calculate ReductionRatio when profileArea and containerArea is changed
3   */
4  useEffect(() => {
5     setReductionRatio(profileArea > 0 ? (containerArea/profileArea) : (0))
6  },[profileArea, containerArea, tfcBufferValues]);

```

Listing 5.10: useEffect hook

### TFC confirm page optimization

Since the TFC-form contains almost 100 text-fields, drop-downs, functions and dates, we made the decision to split up the forms into different pieces to keep the application neat and high-performing. Our initial line of thinking was that Benteler would fill out the TFC streamlined from top to bottom. The confirm page, containing all the input fields, was initially meant for only checking that the values previously added were correct. After implementing and showing this solution to Benteler, they wished to be able to change every value in the confirm page as well. We expressed our concern for a drop in performance. As initially thought, changing values in the confirm page can be experienced as having a bit of latency on input or switching fields, but it is still fully functional.

### 5.2.4 Automatic version and status control

To eliminate having to rely on the user typing the correct ID and version number for any Request, TFC or Order, we decided to automate this process. This meant that for each time a user wanted to perform an operation on any type of form, whether it would be creating a new one, base off an old, change or update, the application would deal with keeping integrity.

For example at the Request main page, when a user selects a desired Request and the menu option "Change Request", the web interface automatically calls the API as seen in listing 5.11. This API call then is received and processed by the backend API as seen in listing 5.12. The API then receives which Request number the user has chosen, and then queries the database to find the newest version in use. This version is then returned to the web interface and added up to a new unique version number to be inserted automatically into the form.

```

1  /*Frontend function calling the API for the latest version*/
2  getLastVersionById = (id) => {
3     return axios.get(API_URL + "request/${id}/lastVersion");
4  }

```

Listing 5.11: Frontend API call for retrieving the latest Request version per a given ID

```

1 //API Endpoint function (receiving function)
2 @GetMapping(path="{id}/lastVersion")
3 public String getLastVersionById(@PathVariable("id") String id){
4     return(RequestService.getLastVersionById(id, con));
5 }
6
7 //Function called by API endpoint to retrieve the last request version
8 public static String getLastVersionById(String id, Connection con){
9     String version = null;
10    try{
11        ResultSet rs;
12        //Queries the database for the latest version
13        PreparedStatement st = (PreparedStatement) con.prepareStatement("Select MAX(
14        request_ver_no) as Version FROM Request WHERE Request_no = ?");
15        st.setString(1, id);
16        rs = st.executeQuery(); //Executing the statement
17        while (rs.next()) {
18            version = rs.getString("Version");
19        }
20    }catch(SQLException ee){
21        ee.printStackTrace();
22    }
23
24    return version; //Returns the latest version
25 }

```

Listing 5.12: Backend API function for retrieving the latest Request version per a given ID

The application also automatically keeps track of the status per a Request. A Request can have either status: **Open**, the Request is not used anywhere yet, just created.

**TFC**, the Request has been used in a Team Feasibility Commitment meeting (TFC) and is included in a TFC form.

**Order**, the Request has been used in a submitted Order

These statuses are automatically set whenever the Request is created, a TFC is made or Order is created. As seen in listing 5.13 on line 20, the web interface calls the accompanied web-interface service to call the API. The API then receives the Request number, Request version number and the status as a parameter (seen in listing 5.14). The status field for the specific request is then updated in the database to contain the attached status.

This status is then displayed when the user is browsing Requests in the search pages (See Figure 5.6). This is most useful for when the user is creating a new Order or TFC, to more easily keep track of which Request is already used and which are ready to be processed (See Figure 5.7).

```

1  /* Submit function for the TFC form */
2  const handleSubmit = () => {
3      fetch(configData.API_URL + "tfc/", {
4          method: "POST",
5          headers: {
6              "Content-type": "application/json",
7          },
8          body: JSON.stringify(tfcValues),
9      })
10     .then((response) => response.json())
11     .then((result) => {
12         setOpen(true);
13         if (result !== true){
14             setSeverity("error");
15             setMessage("Failed to add TFC to database!")
16         }
17         else{
18             /* Removed code for readability */
19             /* This next line sets the status of the given request to "TFC"
20              automatically */
21             RequestService.setStatus(tfcValues.Request_no, tfcValues.
Request_ver_no, "TFC");
22         }
23     });
24 };

```

Listing 5.13: Example functions for automatic status control in frontend following a submission of a form

```

1  //API Endpoint function (receiving function)
2  @PostMapping(path = "/setStatus/{requestno}/{requestver}/{status}")
3  public boolean setStatus (@PathVariable("requestno") int requestNo, @PathVariable("
requestver") int requestVer, @PathVariable("status") String status){
4      return(RequestService.setStatus(requestNo,requestVer,status,con));
5  }
6
7  //Function called by API endpoint to change status parameter of a request
8  public static boolean setStatus(int requestNo, int requestVer, String status,
Connection con){
9      try{
10         //Updates the status field in the database
11         PreparedStatement st = (PreparedStatement) con.prepareStatement("UPDATE
Request SET Status = ? WHERE Request_no = ? AND Request_ver_no = ?");
12         st.setString(1, status);
13         st.setInt(2, requestNo);
14         st.setInt(3, requestVer);
15
16         st.execute(); //Executing the statement
17
18         return true; //Returns true if successful
19     }catch(SQLException ee){
20         ee.printStackTrace();
21         return false; //Returns false if not successful
22     }
23 }

```

Listing 5.14: Functions for automatic status control in the backend API following a submission

TFC									
REQUEST   TFC									
Search							Search for	Request_no	+
Request Number	Request Version Number	Date	Sketch Number	Request Name	Customer	Project Name	Part Category	Status	
8005	1	2021-04-28	1	testSurface	BAS Tønder	1	Beam	Order	
8003	1	2021-04-26	4	Jesper	BAS Tønder	a2d2	Beam	TFC	
8003	2	2021-04-26	4	Jesper	BAS Tønder	a2d2	Beam	TFC	
8004	1	2021-04-26	4	Jesper	BAS Tønder	a2d2	Beam	TFC	
8004	2	2021-04-26	4	Jesper	BAS Tønder	a2d2	Beam	Open	

Figure 5.6: Status field/overview

Request Date	TFC number	TFC version	Sketch Number	Sketch revision	Request Name
2021-04-28	1				
2021-04-26	1	8005	1	2021-04-28	Order
2021-04-26	1	8003	1	2021-04-26	TFC
2021-04-26	1	8003	2	2021-04-26	TFC
2021-04-26	1	8004	1	2021-04-26	TFC
2021-04-26	1		4		Jesper

Search for request nr

Request Number	Version Number	Date	Status
8005	1	2021-04-28	Order
8003	1	2021-04-26	TFC
8003	2	2021-04-26	TFC
8004	1	2021-04-26	TFC

CANCEL    SELECT REQUEST

Figure 5.7: Status overview in popup after selecting an old TFC as base for a new TFC

### 5.2.5 Invalid requests

The process for setting a Request as invalid is code-wise pretty similar to setting status, besides it having dedicated buttons in the interface only viewable for the Admin/Superuser. In addition an invalid Request also triggers logic when selected that disables all buttons to process the Request further, so that the Request has to be set as valid to be further processed. In the main Request menu, a Request's valid/invalid can be toggled by buttons in the UI as seen in Figure 5.8.

**Requests**

Selected invalid request

Request Number	Request Version Number	Date	Sketch Number	Request Name	Customer	Project Name	Part Category	Status
8005	1	2021-04-28	1	testSurface	BAS Tender	1	Beam	Order
8003	1	2021-04-26	4	Jesper	BAS Tender	a2d2	Beam	TFC
8003	2	2021-04-26	4	Jesper	BAS Tender	a2d2	Beam	TFC
8004	1	2021-04-26	4	Jesper	BAS Tender	a2d2	Beam	TFC
8004	2	2021-04-26	4	Jesper	BAS Tender	a2d2	Beam	Open
8002	1	2021-04-26	4	Jesper Trøan	BAS Tender	a2d2	Chassis	Open
8000	1	2021-04-24	4	Jesper	BAS Tender	a2d2	Chassis	Order
8000	2	2021-04-24	4	Jesper Trøan	BAS Tender	a2d2	Chassis	Open
8001	1	2021-04-24	1	test	BAS Tender	43	THN	Open
8002	1	2021-04-26	4	Jesper Trøan	BAS Tender	a2d2	Chassis	Open

Toggle buttons

Figure 5.8: Toggle invalid Request with disabled buttons

## 5.2.6 Change measurement system

The switch in the topbar, see Figure 5.4, changes the measurement system used in the application between metric and imperial system. Since Benteler has divisions in the United States, one of the requirements was to be able to create Requests, TFC's and Orders in both systems. When pressing the switch, the page re-renders and every measurement is changed to the opposite. The measurement system is then stored in the browser so the users preference is saved.

To convert between units, we have used an existing JavaScript library called `convert-units` [1]. This library is easy to use and includes almost all the conversions needed for the application. The values are stored as metric values in the database, so if the user uses imperial, the values need to be converted when displayed in the application as well as when the user submits a form. An example is shown in listing 5.15 underneath.

```

1 /**
2  * A snippet from the billet length textfield in TFC, if the user has
3  * switched to imperial. Converts the billet length from millimeters to
4  * inches.
5  */
6   convert(tfcValues.Billet_length_mm).from('mm').to('in'))
7 /**
8  * Function to handle input from inches to millimeter. Used to convert users
9  * inch-input to millimeter stored in the database. parseFloat used to
10 * limit number of decimals.
11 */
12 const handleInputChangeINtoMM = (e) => {
13   const {name, value} = e.target
14   setTfcValues({
15     ...tfcValues,
16     [name]: parseFloat(convert(value).from('in').to('mm')).toFixed(3)
17   })
18 }

```

Listing 5.15: Conversion example

## 5.2.7 Security

In the research period, the team found a React Security online course [7] which we found to be relevant and the team decided to take. From this course we got ideas on what frontend security measures we should implement, which we will present in this chapter.

### Authentication & Authorization

Even though the application is only accessed through a VPN or by the client's internal network, there was a need for access control. Due to the requirement to limit what certain users can view and access in the interface, users were divided into multiple groups with different permissions.

The first thing the user meets when using the application and one of the first things the team decided to implement was the login functionality. It was important that this was secure and that we meet the authorization requirements.

We knew that the security of the application was highly dependent on how we were to implement the authentication and authorization, and we had to do a lot of research to come up with a good solution concerning this. The solution we landed on was to use JSON Web Token (JWT) to authenticate and authorize each user. By using JWT we prevented session hijacking and manipulation of the user's cookie to obtain access, which was a couple of the main security concerns. To read more about JWT and how we implemented it go to section 5.3.6.

In addition to the security aspect, we also knew that the login functionality had to cover the authorization requirements. To solve this we wanted each user to have a role saved with the user in the user database which should be returned from the API when the user submits a successful login.

The login functionality is implemented by creating a simple login page where the user can insert its username and password, when the user submits this the information will be sent to the API through Axios (listing 5.1).

```
1 /**
2  * Login function called every time a user clicks submit
3  * Sends the username and password to the API,
4  * which returns the JWT and role.
5  * This information gets saved to the user's cookie
6  * and is used to authenticate the user with the authenticate function
7  * in authContext
8  * Sets loading to false after the API response.
9  */
10 function login() {
11     var token;
12     var role;
13     var username = details.name;
14     setLoading(true);
15
16     /* sends username and password and returns JWT and role */
17     axios
18     .post(API_URL + "auth", details)
19     .then((response) => {
20         token = response.data[0];
21         role = response.data[1];
22         Cookies.set("token", token);
23         Cookies.set("role", role);
24         Cookies.set("username", username);
25         authContext.Authenticate({ token, role, username });
26         setLoading(false);
27     });
28 }
```

Listing 5.16: Authentication request

As you can see in listing 5.16 the login information is sent to the API on line 17 and gets the token and role as a response. The token, role and username are then saved to the user's cookie and the further authentication is done by the *Authenticate* function in the `authContext` file.

One of the things we learnt from the React Security online course is to make use of React's Context API. "Context provides a way to pass data through the component tree without having to pass props down manually at every level."<sup>4</sup>, this was something we considered to fit our authentication and authorization functionality. Because of this, every authentication and authorization code was then implemented in the file `AuthContext.js`.

The `AuthContext` file contains a state called `authState` which consists of the user's username, token and role and is used in many of the authentication and authorization functions. The *Authenticate* function used in `login` (Listing 5.17), sets the `authState` to contain the relevant values. The state can also be set by getting the information from the user's cookies, this ensures that a user stays logged in as long as the user contain the correct values in their cookies.

---

<sup>4</sup><https://reactjs.org/docs/context.html>

```

1  /** Function used in login class, to authenticate the username and password */
2  const Authenticate = (data) => {
3    setAuthState({
4      username: data.username,
5      token: data.token,
6      role: data.role
7    })
8  }

```

Listing 5.17: Authentication function in authContext

After the user has submitted the login information and the authState has been updated, the App.js file with all the routing gets rendered again. The App.js uses the authContext and the first function called is the *isAuthenticated* function listed in 5.18, which will change the authContext state "Approved" to true only if the token has been checked and is approved by the API. This function gets a new JWT and updates the user's cookies to always contain the newest JWT, this is important since the JWT contains an expiration time and it will not be approved if that time has expired. Since this function gets called on every App.js render (which is every time a user navigates through the application), the rest of the authentication in the application is done by simply checking the "Approved" state. The function approvedJWT is made for that and will return the value of the state.

```

1  /**
2  * This function retrieves the JSON Web Token from the cookie,
3  * and then calls the API with it in the HTTP body,
4  * the JSON Web Token is then asserted in the API
5  * and the response is the data contained in the JWT,
6  * which is then used to update the data in the cookie
7  * and updated the state that tells if the JWT was approved or not
8  * @state approved {boolean} Is the JWT approved or not?
9  */
10 const isAuthenticated = async () => {
11
12   await axios.post(API_URL + "auth/jwtCheck", {}, {
13     headers: {
14       "Authorization": Cookies.get("token")
15     }
16   })
17   .then((response) => {
18     if (response.data) {
19       Cookies.set("token", response.data.jwt);
20       Cookies.set("role", response.data.role);
21       setApproved(true)
22     }
23     else {
24       setApproved(false)
25     }
26   });
27 };

```

Listing 5.18: isAuthenticated in authContext



As you can see in listing 5.19 the *approvedJWT* function (on line 10) is used to check what route to return to the user. If the JWT is approved the user will be able to navigate through the main routes of the application, but if the JWT is not approved the user gets sent back to the login page. Since the App.js renders every time the user navigates through the application, the authentication happens constantly and the user will be logged out as soon as the JWT is no longer approved.

The authorization of the application is also implemented in the file *authContext*. Listing 5.20 shows the three authorization functions in the *authContext* file, and as you can see the functions returns true if it is either the specific role asked for or the roles above in the hierarchy. These functions are used every place in the code where the access is restricted to specific roles.

```

1  const AppRoutes = () => {
2  const auth = useContext(AuthContext);    /*Using authcontext for authentication*/
3  auth.isAuthenticated()                  /*Checks if the user is authenticated*/
4  /**
5   * Returns the whole application and the path to each page.
6   */
7  return (
8    <>
9    <Suspense fallback={<div><CircularProgress /></div>}>
10     {auth.approvedJWT() ? ( //Returns if the JWT is approved
11       <div>
12         <TopNavbar />
13         <Sidebar />
14         <div className="main-big">
15           <Switch>
16             <Route path="/" exact>
17               <Home />
18             </Route>
19             <Route path="/search">
20               <Search />
21             </Route>
22             ...
23           </Switch>
24         </div>
25         <Footer />
26       </div>
27     ) : (
28       /*Send user to login page if not logged in*/
29       <div>
30         <Redirect to="/login" />
31         <Route path="/login">
32           <Login />
33         </Route>
34       </div>
35     )}
36   </Suspense>
37 </>
38 );
39 };

```

Listing 5.19: The routing authentication

```

1  /**
2  * Functions to determine the user's role, used for determining available menu
   options
3  * @state authState  A const temporarily storing user data
4  */
5  const isAdmin = () => {
6      return authState.role === "admin";
7  };
8  const isSuperuser = () => {
9      return (authState.role === "admin" || authState.role === "superuser");
10 };
11 const isCostcalcuser = () => {
12     return (authState.role === "admin" || authState.role === "superuser" || authState
        .role === "costcalc");
13 };

```

Listing 5.20: The authorization functions

An example of using the authorization functions is in the routing in App.js. Pages like TFC and CostCalc needs to be restricted and only accessible for the roles that should have access to those, to solve this we created routes named "AdminRoute", "SuperuserRoute" and "CostCalcRoute" which uses the authorization functions to check if the user has the correct access and if not the user is redirected to the home page. Listing 5.21 shows the AdminRoute function and how it is used in the switch tag.

Another example of using the authorization function is in the sidemenu, which only shows the icons the user has access to viewing (Listing 5.22).

```

1  /**
2  * Function to redirect every non admin users if they try to access admin pages
3  * @param {*} children  The files using admin routes.
4  * @param {*} ...rest  Paths to each file.
5  */
6  const AdminRoute = ({ children, ...rest }) => {
7      const auth = useContext(AuthContext); /*Using authcontext for authentication*/
8      return (
9          <Route
10             {...rest}
11             render={() =>
12                 auth.approvedJWT() && auth.isAdmin() ? (
13                     <>{children}</>
14                 ) : (
15                     <Redirect to="/" />
16                 )
17             }
18         ></Route>
19     );
20 };
21 <Switch>
22     <Route path="/" exact>
23         <Home />
24     </Route>
25     ...
26     <AdminRoute path="/users">
27         <Users />
28     </AdminRoute>
29     ...
30 </Switch>

```

Listing 5.21: An example showing the authorization routing

```
1  /**SidebarData examples*/
2  {
3    id: 3,
4    title: "Requests",
5    path: "/requests",
6    icon: <AiIcons.AiOutlineFileAdd/>,
7    allowedRoles: ["admin", "superuser", "normaluser"]
8  },
9  {
10   id: 4,
11   title: "TFC",
12   path: "/tfc",
13   icon: <AiIcons.AiOutlineFileText/>,
14   allowedRoles: ["admin", "superuser"]
15 },
16
17 {SidebarData.map((item) => {
18   return(
19     div key={item.id}>
20       {(item.allowedRoles.includes(authContext.authState.role)) && (
21         <li className = "menu-text-close">
22           <NavLink to={item.path} exact activeClassName="active" >
23             <div>
24               <>{item.icon}</>
25               <p className="menu-text-p">{item.title}</p>
26             </div>
27           </NavLink>
28         </li>
29       )}
30     </div>
31   )
32 }}}
```

Listing 5.22: Authorization in the sidemenu

### Input validation

As our application relies heavily on input from the user, input validation had to be done properly. The Material-UI components in the frontend already provided us with basic numeric and date validation. In addition, we used a npm library called "validator" which is an API that we used to validate emails and text (Listing 5.23 line 11 and 12).

```

1 import validator from "validator"
2 ...
3 <TextField
4   id="Product_controller"
5   placeholder="Product Controller Email"
6   label="Product Controller Email"
7   margin="normal"
8   variant="outlined"
9   defaultValue={values.Product_controller}
10  onBlur={handleChange("Product_controller")}
11  /* Following line sets field border to red if validation fails */
12  error={validator.isEmail(values.Product_controller) ? false : true}
13  /* Following line displays a message if validation fails */
14  helperText = {validator.isEmail(values.Product_controller) ? "" : "Not a valid
    Email"}
15  inputProps={{
16    maxLength: 100,
17    style: {
18      width: "28ch",
19    },
20  }}
21  className={classes.textField2}
22 />
23 ...

```

Listing 5.23: Email validator

### Lazy loading

Another security measure we implemented was Lazy Loading<sup>5</sup>, which means that the website only loads the code that is needed. This was especially relevant to our application since we have divided the users into different roles, this means that the normal users will not load all the code that is only relevant to the superuser or admin. The reason we thought this was a good security measure was because it limits the access to the code, which prevents a restricted user from getting secret information from the code. In addition to being a security measure, it could also increase the efficiency of the application by not loading as much code at once.

To implement Lazy Loading we used React's functions *lazy* and *Suspense*<sup>6</sup> as shown in listing 5.24. We used the *lazy* function to import the different files in App.js. This function imports each file into a lazy-component, and the code in that file is not loaded until the lazy-component is rendered. The *Suspense* component is used in addition to the function to show fallback content (such as a loading screen) while waiting for the lazy-component to load, this was implemented by adding a *Suspense* around the routing with the lazy components.

<sup>5</sup>[https://en.wikipedia.org/wiki/Lazy\\_loading](https://en.wikipedia.org/wiki/Lazy_loading)

<sup>6</sup><https://reactjs.org/docs/code-splitting.html>

```

1  /**
2   * Lazy function used to import the files so that the application uses lazy loading.
3   */
4  const Sidebar = lazy(() => import("./components/Sidebar"));
5  const Home = lazy(() => import("./pages/Home"));
6  const Search = lazy(() => import("./pages/Search"));
7  const RequestForm = lazy(() => import("./pages/Request/RequestForm"));
8  const SearchRequest = lazy(() => import("./pages/Request/SearchRequest"));
9  const TFC = lazy(() => import("./pages/TFC/TFC"));
10 const CostCalc = lazy(() => import("./pages/CostCalc"));
11 const Order = lazy(() => import("./pages/Order/OrderForm"));
12 const Database = lazy(() => import("./pages/Database"));
13 const Users = lazy(() => import("./pages/Users"));
14 const TopNavbar = lazy(() => import("./components/TopNavbar"));
15 const Footer = lazy(() => import("./components/Footer"));
16 const TFCForm = lazy(() => import("./pages/TFC/TFCForm"));
17 const SearchOrder = lazy(() => import("./pages/Order/SearchOrder"));
18 const FAQ = lazy(() => import("./pages/FAQ"));
19
20 /**
21  * Returns the whole application and the path to each page.
22  */
23  return (
24    <Suspense fallback={<div><CircularProgress /></div>>
25      {auth.approvedJWT() ? ( //Returns if the JWT is approved
26        <div>
27          <TopNavbar />
28          <Sidebar/>
29          <div className="main-big">
30            <Switch>
31              <Route path="/" exact>
32                <Home />
33              </Route>
34              ...
35            </Switch>
36          </div>
37        </div>
38      </Suspense>
39    );

```

Listing 5.24: Lazy Loading

## Password

The criteria we have used to ensure the user changes to a strong password, are from OWASP Password Security Requirements [13]. OWASP [8] is an online community that produces freely available tools and documentation used for web application security.

The main requirements we have implemented is that the password should contain both numbers, lowercase letters and uppercase letters. The password length must be at least 8 characters, but 12+ characters are recommended. According to the OWASP requirements, we implemented a password strength meter (Figure 6.5). The user can see the strength of the password in realtime, which helps make good and strong passwords. In listing 5.25 is the function setting the progress of the strength meter. The meter increases if the password meets the different requirements. To max out the meter, the user has to have at least one special character, which can not be at the end, as well as a length of over 11 characters.

```

1  /**
2  * Function used to set the strength of the strength meter. Runs when the first
3  * password is changed. Checks if the user uses numbers, letters and special
4  * characters.
5  */
6  useEffect (() => {
7      let num = 0;
8      if(number) {    num += 15;    }
9      if(lowercase) {    num += 10;    }
10     if(uppercase) {    num += 15;    }
11     if(pwd1.length > 7) {    num += 10    }
12     if(pwd1.length > 9) {    num += 15    }
13     if(pwd1.length > 11) {    num += 15    }
14     if(pwd1.match("/[!@#%$%^&*()_+\\-=\\[\\]\\{\\};':\\|\\.,.<>\\/?]/")) {
15         num += 10
16     }
17     if(pwd1.match("/[!@#%$%^&*()_+\\-=\\[\\]\\{\\};':\\|\\.,.<>\\/?]/")
18         &&
19         !pwd1.charAt(pwd1.lastIndexOf).match("/[!@#%$%^&*()_+\\-=\\[\\]\\{\\};':\\|\\.,.<>\\/?]/"))
20     {
21         num += 10
22     }
23     setProgress(num);
24 }, [pwd1]);

```

Listing 5.25: Strength meter

Every password field has a visibility button, for the user to check the password entered. The goal of the user to view their password is to improve the usability and help motivate the users to choose long passwords. The users also have to enter their old password before they are allowed to change it.

## 5.3 Backend API

We divided the backend into multiple packages: API, model and service, to create an clear architecture (as seen in Figures 5.9 and 5.10). The API package contains all the API endpoints reachable by HTTP requests and we separated the different endpoints by defining separate paths or using separate HTTP request methods. The endpoint functions responsibility is calling the corresponding service in the service package. The services consist of classes that are responsible for:

- Inserting, fetching or modifying data
- Handling JSON Web Tokens received from the user by assertion, updating with new expiry and encryption.
- Mail service to utilize Gmail's SMTP service [22] to send both confirmation emails and notifying emails to the respective users. Changed to use Benteler's internal SMTP upon deployment.

The model package contains classes that are utilized by the services and are used to create objects from retrieved database data. These objects are then returned to the user in the shape of a JSON Object by the API classes.

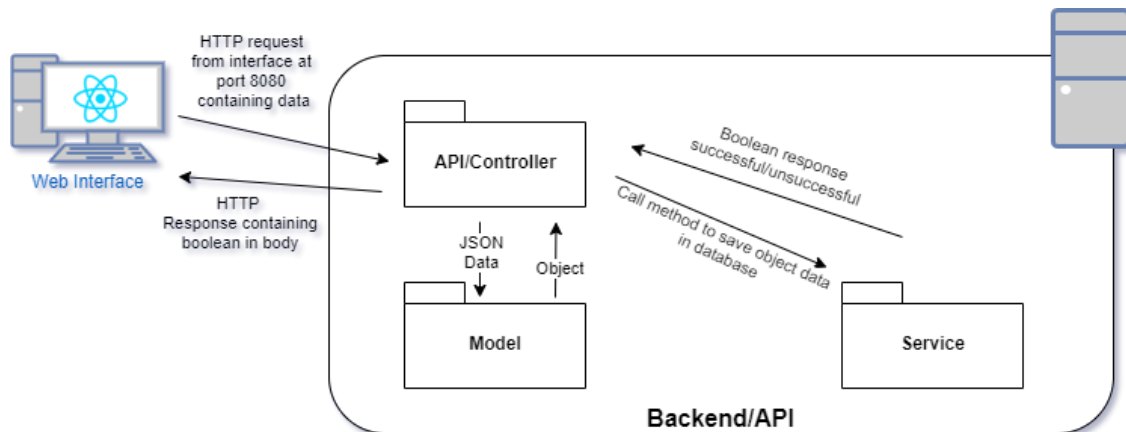


Figure 5.9: Backend API data flow when submitting data

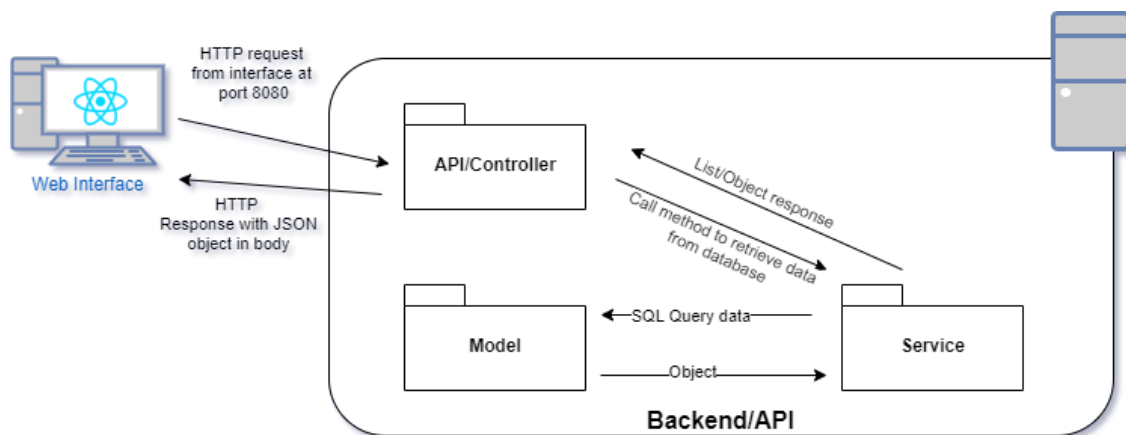


Figure 5.10: Backend API data flow when fetching data

The way parameters are sent from the web interface to the backend via HTTP requests is both in the HTTP body as well as in the HTTP path if the number of parameters are three or fewer. These parameters are then extracted using Spring Boot's `@PathVariable` notation, or `@RequestBody` if the HTTP body contains the data.

Spring boot also delivers an integrated tomcat server and handles all networking and routing. First of all, it lowered the required work to get up and running, and second provided us with a tried and tested protocol. This eliminates potential security, performance or instability issues from developing a similar protocol ourselves.

### 5.3.1 API Layer / Controller classes

As illustrated in the code example 5.26, all controller classes job is to convert incoming data contained in the URL or JSON objects in the HTTP body to a known object that the services can work with. `@PostMapping` tells Spring Boot that this function should be called when the defined URL is received by the built-in tomcat server.

Once the function is called, the `@RequestBody` notation is used to tell Spring Boot to convert the incoming data format into an object as defined following the notation. In this example the data is converted into the format of the `TFCForm` class in the models layer.

```

1 @PostMapping(path = "{id}/{ver}") /* Defining the path/URL variables anticipated */
2 /* JSON data converted to TFC Object with @RequestBody: */
3 public boolean updateTFC(@PathVariable("id") String id, @PathVariable("ver") String
4   ver, @RequestBody TFCForm tfcf) {
5   return(TFCService.updateTFCByID(tfcf, id,ver, con)); /* Calls TFCService.js */
6 }

```

Listing 5.26: Example of controller class

For classes where having a dedicated object was unnecessary (see listing 5.27), we first defined the path with `@PostMapping` while defining the variable names. Then we used the `@PathVariable` notation followed by the variable name as its parameter. Lastly defining the data type it should be converted to.

```

1 @GetMapping (path = "spacer/{name}")
2 /* URL parameter converted to function parameter with @PathVariable: */
3 public Spacer getSpacer(@PathVariable("name") String name) {
4   return(ParamService.getSpacerByName(name, con)); /* Calls ParamService.js */
5 }

```

Listing 5.27: Controller class receiving URL parameters



### 5.3.2 Model layer

The structure of the model classes is rather simplistic as illustrated in listing 5.28, with the purpose of serving objects usable for the service layer. To avoid creating too many unnecessary model classes, we decided to send certain parameters via the URL. This is done for the retrieving classes which for example receives index numbers and revision to search the database for.

As illustrated in listing 5.28, we had to implement the `@JsonProperty` notation in order to specifically tell the class constructor which JSON name/value pair is intended to be stored into the specified variable for the created object. The model class then contains all necessary retrieval "get" functions for extracting object data.

```
1 import com.fasterxml.jackson.annotation.JsonProperty;
2
3 public class Spacer {
4     String spacertype;
5     float weight;
6     float height;
7
8     /* @JsonProperty used to map JSON key-value pairs to function parameters */
9     public Spacer(@JsonProperty("Spacertype") String spacertype,
10                 @JsonProperty("Weight") float weight,
11                 @JsonProperty("Height") float height) {
12
13         this.spacertype = spacertype;
14         this.weight = weight;
15         this.height = height;
16     }
17
18     public String getSpacertype() { /* Functions used for retrieving object data */
19         return spacertype;
20     }
21
22     public float getWeight() {
23         return weight;
24     }
25
26     public float getHeight() {
27         return height;
28     }
29 }
```

Listing 5.28: Example of a smaller model class

### 5.3.3 Service layer

Listing 5.29 illustrates a common format of a function in the service classes. The illustrated function queries the database and returns a list with the result.

```

1  /* Function for retrieving all spacers from the database */
2  public static List<Spacer> getAllSpacers(Connection con) {
3      Spacer spacer = null; // Buffer
4      List<Spacer> spaceList = new ArrayList<>(); // List to be returned
5
6      try {
7          ResultSet rs;
8          PreparedStatement st = (PreparedStatement) con.prepareStatement("SELECT *
9  FROM Spacers;");
10         rs = st.executeQuery();
11
12         while (rs.next()) {
13             spacer = new Spacer(
14                 rs.getString("Spacertype"),
15                 rs.getFloat("Weight"),
16                 rs.getFloat("Height")
17             );
18             spaceList.add(spacer);
19         }
20     } catch (SQLException ex) {
21         ex.printStackTrace();
22     }
23     return spaceList;
24 }

```

Listing 5.29: Example of a smaller service class

The service layer consists mainly of functions querying the database using SQL. Where data is either stored, retrieved, modified or deleted. These service classes utilize the model classes for creating objects from retrieved database data, to make a returnable object or list with objects.

The service layer also contains the JWTService and MailService. In the backend these classes have the unique responsibility of managing user session tokens and sending notification by email (using SMTP) respectively.

### 5.3.4 Database

A central part of the application is the database, as the application heavily relies on the flow of information. The nature of the application did not require a database containing interconnected tables, besides the connection between a *TFC* entry and its corresponding *Request* done with a simple SQL query. This made the design of the database structure pretty simple, but still while providing a non-redundant data structure as you can see in Figure 5.11.

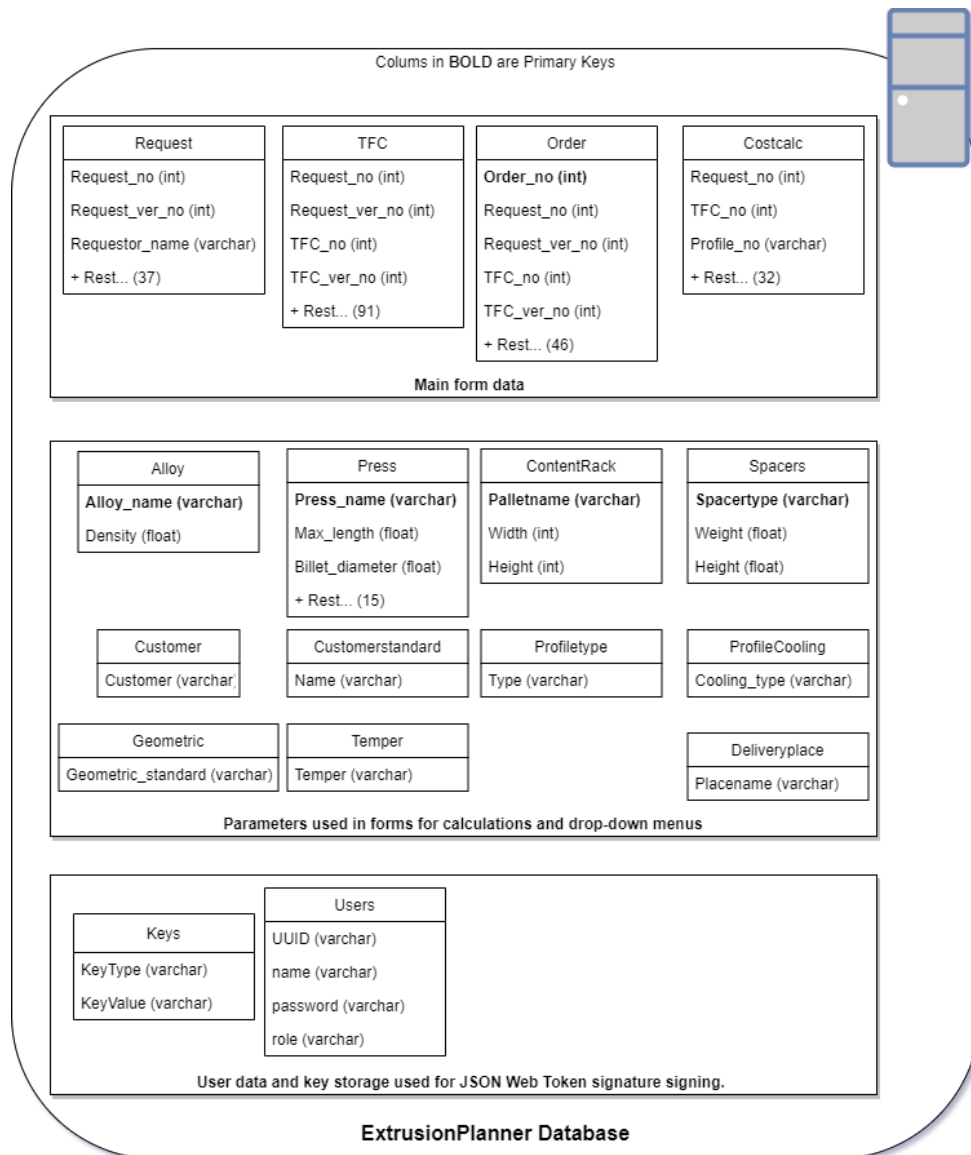


Figure 5.11: Database Structure

The CostCalc team at Benteler was unsure if exporting required data for profile cost calculations to Excel, or saving the required data to an SQL Database was the best solution. This led us to create a table in the database with the necessary fields called Costcalc, in case it would be of need in the future. 14th of April the team got an update that the CostCalc team preferred to have the export be in the Excel file format for the time being, but the Costcalc table is still kept in case they change their mind in the future.

The database was made using Microsoft SQL Server 2019 and managed using Microsoft SQL Server Management Studio software. After migrating the web server from Openstack to Benteler’s own server, the SQL database had to be downgraded to SQL Server 2012 due to licensing issues.

## Structure

The database consists of three types of tables, **the first type** is the main form data. The main form data tables are catering to the main forms used in the application; TFC, Request and Order. These are responsible for storing large amounts of data that has been filled in by the user using the web interface forms.

**The second type** of tables, are the tables used for storing the parameters that are used by the frontend to deliver drop-down selection fields where the user can select among the existing entries in the parameter table. The content of these tables can be altered using the database page in the web interface of the application. This means that the admin or superuser can modify what the users can choose in the drop-down menus, as well as alter formulas used in the TFC form by changing certain parameters used in the formulas. The TFC form also uses this data to calculate real-time warnings to warn the user if they are moving beyond possible values for certain fields.

**The last type** is the application metadata like the user data and key storage for security measures. Here all the users information is stored, like their ID, name, password and role. The password is stored using a hashing method to ensure a degree of confidentiality. The keys table stores the signing key used with signing the JSON Web Tokens before sending them to the user.

### 5.3.5 CostCalc/Product Controller data export document

```

1 public static byte[] getBridge(int requestNo, Connection con) {
2     try {
3         //Gets all costcalc relevant data for each TFC per a request number
4         List<Costcalc> costcalcs = getCostCalcPerRequest(requestNo, con);
5
6         //Retrieving the file from JAR
7         Path temp = Files.createTempFile("BridgeFile", ".xlsx");
8         Files.copy(CostcalcService.class.getClassLoader().getResourceAsStream("
9         BridgeTemplate.xlsx"), temp, StandardCopyOption.REPLACE_EXISTING);
10        FileInputStream inputStream = new FileInputStream(temp.toFile());
11        Workbook workbook = WorkbookFactory.create(inputStream);
12
13        //Function to fetch a specific sheet, did not work unless index=0
14        Sheet sheet = workbook.getSheetAt(0);
15
16        Costcalc costcalc; //Buffer
17
18        //Initializes the document for manipulation, avoids errors
19        initializeSheet(costcalcs, sheet);
20        int offset = 1; //Defines the row-offset into the document
21        for(int i = 2; i<=costcalcs.size()+1; i++) {
22            costcalc = costcalcs.get(i-2); //Indexing starts at 0
23
24            String suffix = "";
25            int press = Integer.parseInt(costcalc.getPress().substring(1));
26            if(press == 22 || press == 40 || press == 55)
27                suffix = " (NO)";
28            if(press == 16)
29                suffix = " (FR)";
30            if(press == 27 || press == 35)
31                suffix = " (US)";
32
33            sheet.getRow(0+offset).getCell(i).setCellValue(costcalc.getRequest_no());
34            // .. Rest of the cells (x34) .. //
35            sheet.getRow(99+offset).getCell(i).setCellValue(costcalc.getDie_life());

```

```

36     }
37
38     inputStream.close();
39
40     ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
41     //Writes the workbook to the stream to be returned to the web interface
42     workbook.write(outputStream);
43     workbook.close();
44
45     //Returns the ByteArray to be returned to the web interface
46     return outputStream.toByteArray();
47     //outputStream.close(); //Not needed for a byteArrayStream
48 } catch (FileNotFoundException e) {
49     e.printStackTrace();
50 } catch (IOException e) {
51     e.printStackTrace();
52 }
53     return null;
54 }

```

Listing 5.30: Snippet of CostCalc main function see appendix D for full code

One of the functional requirements of the application was having a data export in the form of an Excel document. This document is intended for the Product Controller (earlier named CostCalc) to utilize in order to calculate the necessary costs derived from the process parameters decided in the TFC meeting. We started by having a simple export of the CostCalc document to a blank sheet all created in the frontend web interface. Later in the development process the client realized it was too simple for their needs. They wanted a solution that modified the existing Excel document they had used in their previous program which contained different formulas, parameters and drop-downs. Some fields of this document were then automatically calculated and filled in based on stored parameters on another sheet that was interconnected.

The issue with this was that it could not be done directly in the frontend web interface, we had to instead use the API to do most of the process. The only functionality of the process we could keep in the frontend was the selection of Request numbers from which to gather all data contained in the corresponding TFCs. The API then receives this Request number and starts out by retrieving the necessary data in the shape of CostCalc objects in a list.

We then had to implement their existing file as a template file in the API that was to be packaged in the JAR during deployment. To modify the template document, we used the Apache POI library for Java<sup>7</sup>, which provides an API for handling Microsoft Office document formats. This was necessary to enable modification of an existing Excel template. By traversing the excel document using the coordinate index system, specific cells could be chosen and modified. A for loop for each unique TFC per Request number was used to fill in multiple columns.

Unfortunately, time was spent to make the program work with the client-supplied document, as it was later found out that the function from the Apache POI API that fetches an Excel sheet from a specific index did not properly work unless the index was 0. Choosing any other index would result in an empty document, where as when we moved the sheet to be modified to index 0 and changed hereafter in the code, it worked flawlessly.

When the Excel document was finalized in the API, the file was converted into a ByteArray and then returned to the frontend. In the frontend the file-saver library was used, which delivered an API to save the generated Excel file to the users machine.

<sup>7</sup><https://poi.apache.org/>

## 5.3.6 Security

### JSON Web Tokens

The core of our applications authentication lies in the JSON Web Token or *JWT* for short. Using *JWT* for authentication moves the responsibility of storing the session data from the server to the users own machine. We configured our *JWT* token as seen in listing 5.31, which illustrates the function that is responsible for creating a JSON Web Token upon a successful login by a user or a successful *JWT* assertion. This token is then sent in return to the user and then stored on the users machine in form of a cookie.

```

1  /* Function responsible for creating a JSON Web Token to be sent to the user */
2  public static String createJWT(String id, String issuer, String subject, String role)
3  {
4      SecretKey originalKey = getKeyFromDatabase();
5
6      Calendar cal = Calendar.getInstance();
7
8      cal.setTime(new Date());
9      Date currentTime = cal.getTime();
10     cal.add(Calendar.HOUR_OF_DAY, 1); //Adds one hour to current time
11     Date expiryTime = cal.getTime(); //Sets expiry time to one hour
12
13     String jwt;
14     jwt = Jwts.builder() //Build a new JSON Web Token with key
15         .setIssuer(issuer)
16         .setSubject(subject)
17         .claim("id", id)
18         .claim("role", role)
19         .setIssuedAt(currentTime)
20         .setExpiration(expiryTime)
21         .signWith(originalKey)
22         .compact();
23
24     return jwt;
25 }

```

Listing 5.31: JWTService: Creation of a JWT

Whenever the user decides to navigate to a new route (see listing 5.3), the web interface retrieves the cookie from the browser and sends it to the API for authentication as illustrated in listing 5.32. The assertion function first verifies the tokens signature, integrity or expiry. This is done using *JJWT* from the *io.jsonwebtoken* library [19] by retrieving the signature key from the database and calling the *parseBuilder* supplied by *JJWT*. The *parsebuilder* then checks for modification, signature and expiry of the token in one API call.

If the *JWT* was accepted, the function will return a new *JWT* or null if any of the violations mentioned was found. The API endpoint controller class responsible for handling *JWT* requests then uses this return value to either log the user out by returning null, or recreating the *JWT* using the function illustrated in listing 5.31. The *JWT* data, called *claims*, is extracted from the asserted *JWT* and then fed into the creating function as parameters. The new *JWT* is then returned to the user and saved in the browser, while the web interface allows the user to view the requested page.

The expiry time claim is then set to one hour from the current time, in order to log out users after an hour of inactivity when changing routes in the web interface. This means that the users are not logged out while filling out the forms, as this is a process that may take a while. Meanwhile, if they wish to navigate the side menu after the expiry, a new login is required.

```

1 public static Jws<Claims> assertJWT(String jwtString) {
2     Jws<Claims> jwt; //Buffer to store JWT content/claims
3     try {
4         SecretKey key = getKeyFromDatabase();
5         jwt = Jwts.parserBuilder() //JWT is parsed and validated
6             .setSigningKey(key)
7             .build()
8             .parseClaimsJws(jwtString);
9
10    }
11    catch (SignatureException | MalformedJwtException | ExpiredJwtException me){
12        System.out.println("Assertion failed");
13        return null;
14    }
15    return jwt; // JWT content is returned
16 }

```

Listing 5.32: JWTService: Assert a JSON Web Token

```

1 /*Function responsible for receiving a JSON Web Token from the user and then
2 asserting it*/
3 @PostMapping(path = "/jwtCheck")
4 public Response authJWT(@RequestHeader(value="Authorization") String jwt) {
5     Calendar cal = Calendar.getInstance();
6
7     Jws buffer = JWTService.assertJWT(jwt);
8     if(buffer!=null) {
9         Claims body = (Claims) buffer.getBody();
10        String updatedJWT = JWTService.createJWT(body.getId(), body.getIssuer(),
11            body.getSubject(), body.get("role").toString());
12        System.out.println(cal.getTime() + " - JWT Approved for " +
13            body.getSubject());
14
15        Response rs = new Response(updatedJWT, body.get("role").toString());
16
17        return rs;
18    }
19    else
20        return null;
21 }

```

Listing 5.33: JWT Controller class (API Endpoint)

### SQL prepared statement

To ensure a level of protection against SQL injection, we used prepared statements in all the database SQL queries in the backend. "Prepared statements are resilient against SQL injection because values which are transmitted later using a different protocol are not compiled like the statement template. If the statement template is not derived from external input, SQL injection cannot occur" [38]. This limits an abuser, removing the ability to query the database maliciously like retrieving sensitive data, hashed passwords and modifying with malicious intent. In listing 5.34 you can see an example of a prepared statement in the API.

```

1 public static boolean saveAlloy(Alloy alloy, Connection con) {
2     try {
3         PreparedStatement st = (PreparedStatement) con.prepareStatement(
4             "INSERT INTO Alloy " +
5             "(Alloy_name, " +
6             "Density) " +
7             "VALUES (?, ?);"
8         );
9         //Bind first parameter in the statement
10        st.setString(1, alloy.getAlloy_name());
11        //Bind second parameter in the statement
12        st.setFloat(2, alloy.getDensity());
13        st.execute();
14        return true;
15    } catch (SQLException ex) {
16        ex.printStackTrace();
17        return false;
18    }
19 }

```

Listing 5.34: Prepared statement

### Hashing passwords

Password hashing was implemented in order to put a level of security in the storage and handling of the passwords. The API automatically hashes passwords when new users are created, passwords are updated or when a user tries to log in. A hashing mechanism avoids the passwords being stored in plain text in the database. The code for the hashing method can be seen in listing 5.35.

Unfortunately, we did not have time to implement the infrastructure for adding salts to the passwords, which would have increased the obscurity of the passwords. A salt is random data that is added to the hash of a password [39], meaning that the hashed password will be stored with nonsense data. An intruder that has access to such a salted hash of a password, will not know this specific salt and will not be able to decode the password.

Having to use Benteler's internal network or the Benteler VPN to access the application limits the potential vulnerability of not having salts, but it is still something that we would like to have implemented to ensure a good level of password obscurity.

```

1 /* Function that hashes the incoming string parameter using SHA-1 */
2 public static String hashPass(String password) {
3     MessageDigest messageDigest = null;
4     try {
5         messageDigest = MessageDigest.getInstance("SHA");
6         messageDigest.update(password.getBytes());
7         byte[] resultByteArray = messageDigest.digest();
8
9         StringBuilder sb = new StringBuilder();
10
11        for(byte b: resultByteArray) {
12            sb.append(String.format("%02x",b));
13        }
14        return sb.toString();
15    } catch (NoSuchAlgorithmException e) {
16        e.printStackTrace();
17    }
18    return "";
19 }

```

Listing 5.35: Hashing function



## 5.4 Deployment of the application

Like most software, the application had to be deployed somewhere for the user to reach and use. This seemed intriguing as none of the team members had experience of properly deploying a web application to a server. While deploying we met a set of challenges that had to be solved.

Initially, we only had access to the server supplied by NTNU, where we did not have the need for regular deployment as we easily could run the API with IntelliJ and test the backend using NodeJS development server locally. When we got access to a web server supplied by Benteler, we had to explore the properties on how this was set up to tailor the deployment strategy.

### 5.4.1 Backend deployment

We initially thought the task of deploying the backend API was rather easy, as the Maven API<sup>8</sup> simplified the task of building the deployment JAR file with all necessary dependencies (as long as the POM.xml file was correctly set up). We ran the JAR file on the server but soon discovered that the server would log out the active user after disconnecting from the remote desktop service used. This meant that we had to instead install it as a service to prevent the API from being stopped automatically.

Luckily we found the open-source software called WinSW<sup>9</sup> which did exactly that. The software package consisted of a .exe file and an xml document for the service properties (see Figure 5.12). This XML document had to be edited with the intended properties of the service, like name, automatic start and so on. When all this was configured, we placed the packaged JAR file into this folder and ran the WinSW executable file and voila the service was installed and automatically started. The only drawback of this was that you had to open the output file and scroll all the way down to see the console output from the API.

To re-deploy, all we had to do was stop the service, replace the JAR and then start the service. Besides the slow transfer speed of the deployment server, this solution was fast and easy to perform.

Name	Date modified	Type	Size
extrusionAPI	5/6/2021 1:33 PM	Executable Jar File	66,965 KB
WinSW.NET4.err	5/6/2021 1:36 PM	Text Document	1,185 KB
WinSW.NET4	4/18/2021 10:18 PM	Application	833 KB
WinSW.NET4.out	5/11/2021 1:37 PM	Text Document	197 KB
WinSW.NET4.wrapper	5/6/2021 1:42 PM	Text Document	13 KB
WinSW.NET4	4/18/2021 10:49 PM	XML Document	1 KB

Figure 5.12: API Service folder

### 5.4.2 Frontend deployment

Deploying the frontend web interface came with its own challenges, we already deployed the frontend (and backend but only by running the JAR) on our testing server supplied from NTNU. This meant that we figured out how to do this rather early in the development process and could redo the process on the server supplied by Benteler when the opportunity was presented.

<sup>8</sup>[https://en.wikipedia.org/wiki/Apache\\_Maven](https://en.wikipedia.org/wiki/Apache_Maven)

<sup>9</sup><https://github.com/winsw/winsw>

We had a meeting with our supervisor and an expert in this field to consult on how we could deploy the web interface in the best way possible. We learned that the best solution was to use IIS<sup>10</sup>, which is a service already included with copies of Windows 10. This feature only had to be activated, and then we could start the process with deployment. First, we had to create a production optimized build of the application using the NodeJS API, then we specified this folder in IIS in a newly created website in IIS.

When we finished following the tutorial we got recommended for deploying with IIS<sup>11</sup>, we realized that CORS (Cross Origin Resource Policy) and the routing we used in React did not work at all. This was due to the nature of IIS, this was remedied by installing the URL Rewrite Module<sup>12</sup> and CORS Module<sup>13</sup> from the IIS web page. In addition we had to configure these modules by including a web.config (see listing 5.36) file inside the build folder specified in IIS. This web.config had to specify certain rules for both in order for the web page to work as expected.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <configuration>
3   <system.webServer>
4     <cors enabled="true">
5       <add origin="*" />
6     </cors>
7     <rewrite>
8       <rules>
9         <rule name="React Routes" stopProcessing="true">
10          <match url=".*" />
11          <conditions logicalGrouping="MatchAll">
12            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
13            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
14            <add input="{REQUEST_URI}" pattern="~/api" negate="true" />
15          </conditions>
16          <action type="Rewrite" url="/" />
17        </rule>
18      </rules>
19    </rewrite>
20  </system.webServer>
21 </configuration>
```

Listing 5.36: web.config IIS module configuration

<sup>10</sup>[https://en.wikipedia.org/wiki/Internet\\_Information\\_Services](https://en.wikipedia.org/wiki/Internet_Information_Services)

<sup>11</sup><https://dev.to/sumitkharche/how-to-deploy-react-application-on-iis-server-1ied>

<sup>12</sup><https://www.iis.net/downloads/microsoft/url-rewrite>

<sup>13</sup><https://www.iis.net/downloads/microsoft/iis-cors-module>

## Chapter 6

# User interface

In the planning phase, when we discussed the requirements with our contacts at Benteler, we were told that they did not have any specific design-requirements and we could design the application as we saw fit.

We then had a meeting to decide the main layout of the application and what design principles we should prioritize and implement. Our goal was to create an application that is user friendly and also had a modern look to it. One way of doing this is by having a recognizable layout and recognizable components, which we accomplished when using the framework Material UI for our most repeated components. As written earlier Material UI features React components that uses Googles material design, and are components that is widely used and easily recognizable.

Later in the project the project owner came with the wish that we use Benteler's color scheme and guidelines for the application, since that would fit with their other websites and applications. This was easily implemented by using Material UI's default themes where we put the primary and secondary colors of Benteler. We also made sure to use the colors on the components that were not Material UI components.

The team reviewed Benteler's old system to obtain a good understanding on how the employees were used to move around in the application, and from there started brainstorming on how to improve the efficiency and make it more user friendly.

### 6.1 Color pallet

Figure 6.1 is an extract of the design guidelines provided by Benteler. We have mostly used their main colors in the application, but in some cases we used the further colors to make content stand out.

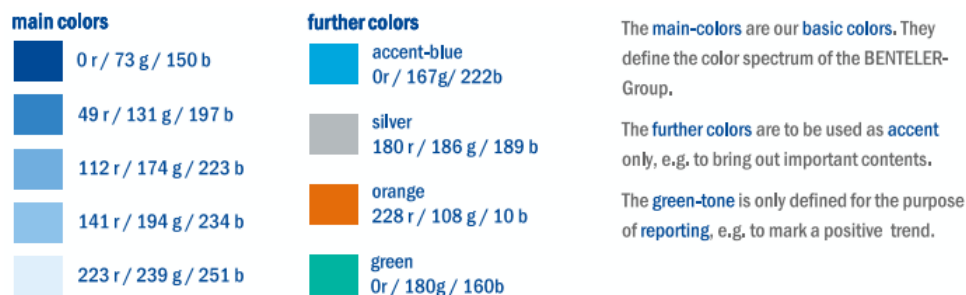


Figure 6.1: Benteler color guideline

## 6.2 Layout

For our layout we decided to use well known components like topbar, side menu and a footer which would be visible at every page after login (see Figure 5.4).

### 6.2.1 Side menu

With the side menu, the users are easily able to move around the application and change pages, in contrast to their old system where they would have to go back to their main page every time they wanted to see the menu (as seen in section 1.2.5).

The main area of the application contains content like forms and tables, which rely on a lot of space. Accordingly, the team wanted to keep the menu small and compact. The solution was a small side menu with icons, which extends when the mouse hovers on it and text would appear below the icons.

The side menu is one of the main components of the system, and the team decided to design this without a framework. The color of the menu is Benteler's primary color, dark blue, and the icons and text are white. The menu is designed with a transition so that when the mouse hover over it, it will smoothly slide out to a bigger size and slide in again when the mouse leaves the menu.

Another detail on the menu is the button colors. The buttons are transparent on default but turns a brighter blue when hovering over it. This is a very popular approach which most big websites follow, and it makes it easy for the user to understand that it is a clickable button. We also wanted a color to mark which button was active so it will show which part of the application the user is on. A light blue color was chosen for this. The colors were both chosen from Benteler's color pallet (see Figure 6.1).

#### Content

The application is divided into 9 different parts: Home, Search, Request, TFC, Cost Calc, Order, Edit Database, Users and FAQ. The 9 parts appear in the side menu with an icon and with text on the extended version. The icons were chosen from React Icons<sup>1</sup>, and the team wanted to have recognizable icons for the different part. An example is a house icon for the home page. The administrator users are the only users that will see all of the 9 parts, other roles will only see the icons available to their role. In Figure 6.2 and 6.3 is two examples with open and closed side menu.

Dividing Request, TFC, CostCalc and Order into their own pages was something the team decided on early. Having these on their own pages made it easy to divide the access to different roles, as well as just being a clear way of dividing the application.

In addition to these four main pages of the application, the team also decided to add a search page. In this page the user could search and find the needed data from the categories the user has access to. The Edit Database page was added as an easy way for the administrator to add or change values used in formulas to the database. Users is another administrator page, in which the administrator can add or edit users of the system. More about the different pages and content is written below.

---

<sup>1</sup><https://react-icons.github.io/react-icons/>

The screenshot shows the 'Extrusion Planner' application interface. At the top, there is a navigation bar with 'Us', 'Me', and 'admin@admin.com'. Below this is a 'Search from' section with tabs for 'REQUEST', 'TFC', and 'ORDER'. A search bar is present with a dropdown menu set to 'Request\_no'. The main area contains a table with the following data:

Request Number	Request Version Number	Date	Sketch Number	Request Name	Customer	Part Category	Place of Delivery	Alloy	Max Depth	Status
7926	5	2021-01-14	085-20	Christoph.Levenig@benteler.com	Audi	Chassis	HARA	6060.35	N/A	Open
7926	4	2020-12-16	085-20	Odd.Perry.Sovik@benteler.com	Audi	Chassis	HARA	6060.35	N/A	Open
7926	3	2020-11-23	085-20	Odd.Perry.Sovik@benteler.com	Audi	Chassis	HARA	6060.35	N/A	Open
7924	3	2021-01-18	083-20	Bernardo.Figueiredo@benteler.com	Porsche	Beam	HARA	6060.35	na	Open
7922	3	2021-01-20	081-20	Bernardo.Figueiredo@benteler.com	Porsche	Crash can	HARA	7003.30	N/A	Open
7922	4	2021-01-20	081-20	Bernardo.Figueiredo@benteler.com	Porsche	Crash can	HARA	7003.30	N/A	Open
7922	5	2021-01-20	081-20	Bernardo.Figueiredo@benteler.com	Porsche	Crash can	HARA	7003.30	N/A	Open
7922	2	2021-01-18	081-20	Alejandro.Velasquez@benteler.com	Porsche	Crash can	HARA	7003.30	N/A	Open
7921	2	2020-11-16	080-20	Bernardo.Figueiredo@benteler.com	Porsche	Beam	HARA	7108.50	N/A	Open
7921	3	2020-11-16	080-20	Bernardo.Figueiredo@benteler.com	Porsche	Beam	HARA	7108.50	N/A	Open

On the right side of the table, there are two buttons: 'DOWNLOAD SELECTED REPORT' and 'DOWNLOAD RESULT REPORT'. The left sidebar menu is currently closed.

Figure 6.2: Closed menu

This screenshot is identical to Figure 6.2, but the left sidebar menu is now open, showing the following items from top to bottom: Home, Search, Requests, TFC, CostCalc, Order, Edit database, Users, and FAQ. The main content area remains the same, displaying the search bar and the table of request data.

Figure 6.3: Opened menu

## 6.2.2 Topbar

From reviewing other popular applications and discussing what we liked and wanted to implement on our own, we decided that a topbar would fit our application well. As it is a well-known component and popular websites like Facebook, Google and Youtube have it.

For the topbar (Figure 6.4) we wanted to go with the popular approach which is having the users settings at the far right corner, this is something that we considered very recognizable since the approach is used by many big websites. With this button, a user can choose to change their password or log out of the application. Another thing we added to the topbar was the metrics switch. Here the user can change between U.S. customary units and the Metric system.

The design of the topbar was kept very simple, with a black background and white text for the logo and buttons. Just like with the rest of the application the button changes when the mouse hovers over it, this time to a white background with black text. We also decided to use one of Benteler's accent colors, orange, for the metrics switch.

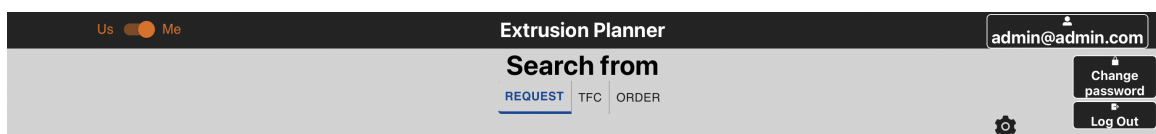


Figure 6.4: Topbar, with the user settings opened

The "Change password" button opens a popup menu (Figure 6.5), with three fields to enter the old and the new password as well as information about the password criteria. To change the password the user must choose a password that covers every criteria set, this assures that the users of the application create strong and safe passwords. The popup dialog consist of a password strength meter, as well as an interactive checklist to make sure the users new password is strong enough.

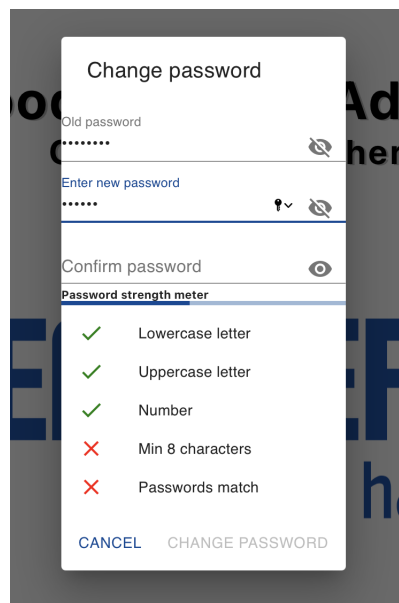


Figure 6.5: The change password popup

### 6.2.3 Footer

Footer (Figure 6.6) was another one of the components we decided to add after reviewing other websites. The footer on the application does not have any functionality and was only added as an extra design component to tie the whole design together. We wanted the footer to resemble the topbar, so we chose to give it a black background color. In the middle of the footer we also added Benteler's logo with the primary color dark blue, which match with the side menu and other components around the application.



Figure 6.6: Footer

## 6.3 Pages

### 6.3.1 Login page

The login page, as seen in Figure 6.7, is the first page the user meets when opening the website. The design of the page is kept very simple and the page only consist of a white login box where the user can enter the login criteria. We wanted to keep the login page as simple as possible. We also wanted the login page separated from the rest of the application to not give away any unnecessary information about the application.

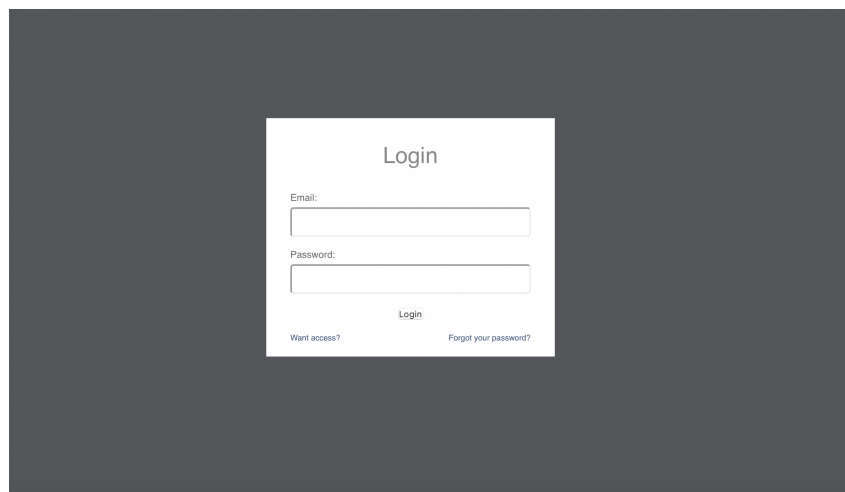


Figure 6.7: Login page

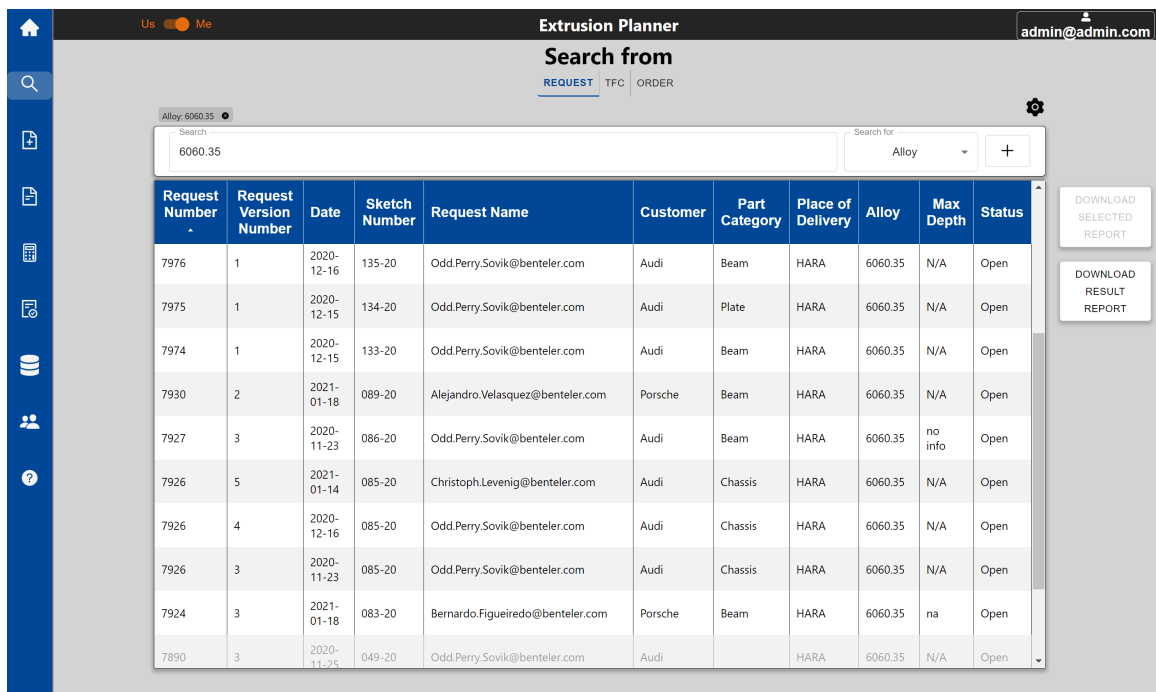
At the bottom of the box is two buttons "Want access?" and "Forgot your password?" which both open a popup with the administrators contact information. This is because the administrator has the right to add new users or to update an existing user. This implementation was made because only employees of Benteler should gain access to the system, and the product owner wanted an administrator to control this.

Another functionality in the login page is that the user gets an error message if the login fails. "The email or password you have entered is not correct." The error is very generic and does not explain whether the problem is the email or password. This is to prevent hackers from trying to "brute force" the account by guessing the password of the account.

After the login, the user will be sent to the home page and can navigate through the application with the side-menu. The user will be logged out again if it is inactive for one hour or if the user logs out with the button on the top-bar.

### 6.3.2 Search page

The search page was made as an easy way for the user to find the data they are looking for. The options for what you can search for varies from role to role. For example, a normal user is only able to search for Request data and Order data, while a superuser and an admin can search for Requests, TFCs and Orders. The Request section of the search page can be seen in Figure 6.8.



The screenshot shows the 'Extrusion Planner' search interface. The search bar contains '6060.35' and the filter is set to 'Alloy'. The table below displays the following data:

Request Number	Request Version Number	Date	Sketch Number	Request Name	Customer	Part Category	Place of Delivery	Alloy	Max Depth	Status
7976	1	2020-12-16	135-20	Odd.Perry.Sovik@benteler.com	Audi	Beam	HARA	6060.35	N/A	Open
7975	1	2020-12-15	134-20	Odd.Perry.Sovik@benteler.com	Audi	Plate	HARA	6060.35	N/A	Open
7974	1	2020-12-15	133-20	Odd.Perry.Sovik@benteler.com	Audi	Beam	HARA	6060.35	N/A	Open
7930	2	2021-01-18	089-20	Alejandro.Velasquez@benteler.com	Porsche	Beam	HARA	6060.35	N/A	Open
7927	3	2020-11-23	086-20	Odd.Perry.Sovik@benteler.com	Audi	Beam	HARA	6060.35	no info	Open
7926	5	2021-01-14	085-20	Christoph.Levenig@benteler.com	Audi	Chassis	HARA	6060.35	N/A	Open
7926	4	2020-12-16	085-20	Odd.Perry.Sovik@benteler.com	Audi	Chassis	HARA	6060.35	N/A	Open
7926	3	2020-11-23	085-20	Odd.Perry.Sovik@benteler.com	Audi	Chassis	HARA	6060.35	N/A	Open
7924	3	2021-01-18	083-20	Bernardo.Figueiredo@benteler.com	Porsche	Beam	HARA	6060.35	na	Open
7890	3	2020-11-25	049-20	Odd.Perry.Sovik@benteler.com	Audi		HARA	6060.35	N/A	Open

Figure 6.8: Search page with an Alloy filter

With Benteler's old system they were only able to search for 4-7 parameters from Requests, TFCs and Orders. Sometimes it can be necessary to search for other parameters as well so they came with a wish to expand the available parameter-search from the old system. Another desired functionality was to be able to search for multiple parameters at the same time. This is all something that we managed to do with the application's search function.

Another thing we wanted to accomplish with the search pages was to conserve the performance of the application. We knew that Benteler's database could contain big tables and loading it all at once could really slow down the application. We discussed this with Benteler and found out that they typically only searched



for the newest data in the database. Our solution to conserve the application's performance was then to only load 100 rows from the database each time. This means that each table contains 100 rows as default, but we also added a button at the bottom of the table which loads a hundred more rows each time it is pressed.

The search function works in a way that a user can choose which parameter to search for in a drop-down menu and then type what that parameter should contain, the filtered result will then show up in the table. For example, if a user is looking for a Request with a specific alloy. The user can then choose alloy as the search parameter and type the desired number in the text-field, the table will then be filtered and only show the Requests containing that specific alloy number. The user also has the option to save the search as a filter, which allows the user the opportunity to make a new search with another parameter. The user can choose to add as many filters as desired and can also easily delete them when needed.

In contrast to Benteler's old system it is now possible to search for all the relevant parameters. The search-type drop-down menu contains all the parameters from the different databases. This means that the drop-down menu for Request contains all Request parameters and the same goes for TFCs and Order. After reviewing the functionality the product owner came with a suggestion to make it possible to search for Request values in the TFC table. This is because Requests and TFCs are linked together and it can be relevant to search for Request values when looking for a specific TFC. The functionality was then improved by letting the users choose from both Request parameters and TFC parameters when looking for TFCs.

In addition to that, the table is customizable. This means that the users can choose which parameters to show in the table. There is a setting-button above the table which opens a popup (Figure 6.9) menu with the parameter choices. This makes it easy for the user to customize the table to its own wishes. The customization will be saved in the user's local storage.

Another functionality in the search page is the report-functionality. On the right side of the search page there are two buttons. With these, the user can download reports with desired data. For example the user can select a row in the Request-table and download data for that Request. A PDF or an Excel file is then downloaded based on the users choices to the user's computer. Another option is if the user wants to download the result from a search, the user can then add the search as a filter and click the "Download result report" button. Here the user has the option to choose between downloading the search results in a PDF or an Excel file.

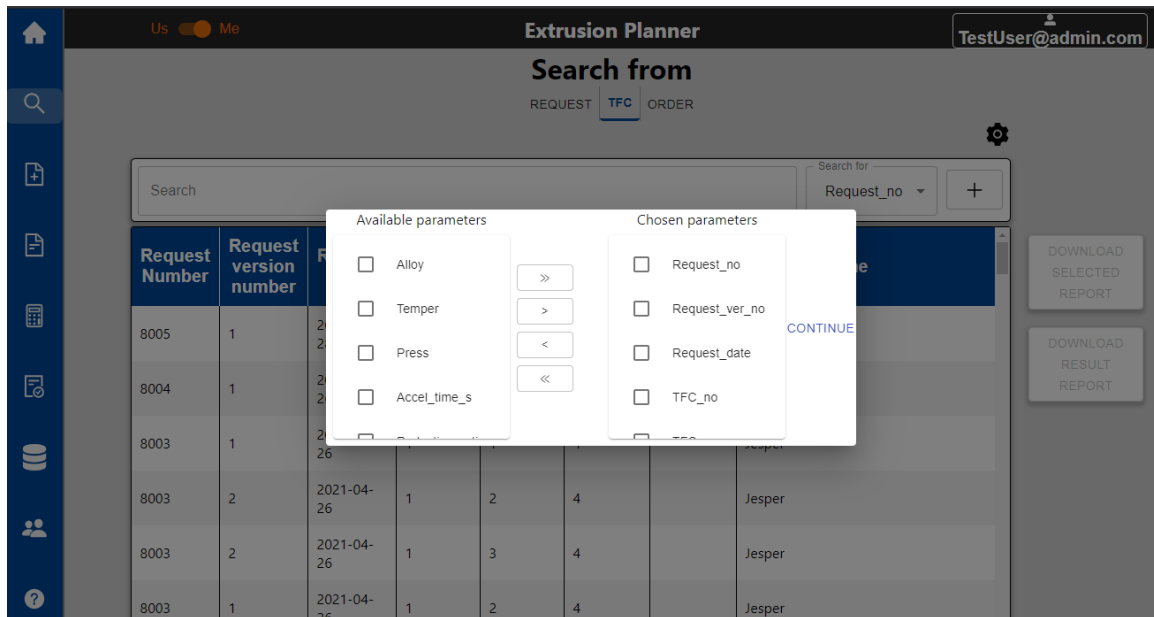


Figure 6.9: Popup menu with parameter choice

### 6.3.3 Request, TFC, CostCalc and Order main pages

The Request, TFC (see Figure 6.10), CostCalc and Order main pages are all very similar to the Search page, they all contain a customizable table with a search function. The difference is the buttons on the right, which varies in all three categories. Another difference is the data the user can search for. The user only has the options to search for the data that is relevant to the category, for example for TFC the user has the option to search for either Requests or TFCs.

The screenshot shows the 'Extrusion Planner' interface. At the top, it says 'TFC' and 'REQUEST | TFC'. There is a search bar with 'Search' and a dropdown for 'Request\_no'. Below the search bar is a table with the following data:

Request Number-	Request Version Number	Date	Sketch Number	Request Name	Customer	Place of Delivery	Alloy	temper	Max Depth	Status
7981	1	2021-01-18	005-21	Bernardo.Figueiredo@benteler.com	Porsche	HARA	7108.50	T5/T6	N/A	Open
7980	1	2021-01-18	004-21	Bernardo.Figueiredo@benteler.com	Porsche	HARA	6082.27	T5/T6	N/A	Open
7979	1	2021-01-18	003-21	Bernardo.Figueiredo@benteler.com	Porsche	HARA	6082.27	T5/T6	N/A	Open
7978	1	2021-01-06	002-21	Amin.Farjad.Bastani@benteler.com	Daimler AG	HARA	6063.85	T1/T4	N/A	Open
7977	1	2021-01-06	001-21	Amin.Farjad.Bastani@benteler.com	Daimler AG	HARA	6063.85	T1/T4	N/A	Open
7976	1	2020-12-16	135-20	Odd.Perry.Sovik@benteler.com	Audi	HARA	6060.35	T5/T6	N/A	Open
7975	1	2020-12-15	134-20	Odd.Perry.Sovik@benteler.com	Audi	HARA	6060.35	T5/T6	N/A	Open
7974	1	2020-12-15	133-20	Odd.Perry.Sovik@benteler.com	Audi	HARA	6060.35	T5/T6	N/A	Open
7973	1	2020-12-03	132-20	Odd.Perry.Sovik@benteler.com	Porsche	HARA	7003.30	F	N/A	Open
7972	1	2020-12-03	131-20	Bernardo.Figueiredo@benteler.com	Porsche	HARA	6063.85	T5/T6	N/A	Open

On the right side of the table, there are three buttons: 'CREATE A NEW TFC FROM REQUEST', 'CREATE A NEW TFC BASED ON AN EARLIER TFC', and 'UPDATE TFC'.

Figure 6.10: TFC search page

The different options in the main pages:

- Request
  - View Request
  - New Request from scratch
  - New with old as base
  - Change Request
  - Recalc
  - Correct Request (only for admins)
  - Set valid (only for admins)
  - Set invalid (only for admins)
- TFC
  - Create a new TFC from Request
  - Create a new TFC based on an earlier TFC
  - Update TFC
- CostCalc
  - Download CostCalc Bridge report

- Order
  - R&D Order
  - Create a new Order from Request
  - Update Order

### 6.3.4 The forms

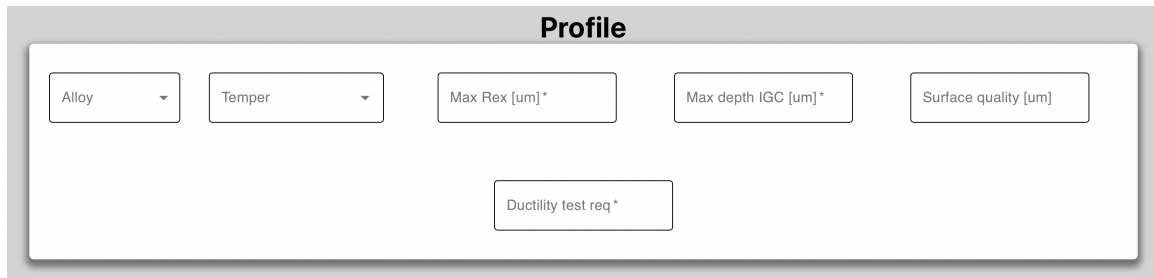
The application has three different forms, the Request form, TFC form and Order form. These are used to send more data to the database. The three forms all have a very similar design and consists of the same components.

Instead of having every text-field pressed into one page like their old system, we decided to divide each form into different parts and add a progress bar to gain a clear overview and to easily navigate between the parts. This approach is very popular in applications with larger forms, and is something we thought could improve the user experience. We suggested the idea to the project owner early in the project and they agreed that it could be a good solution. It could also make the application's performance better by not having to load all the text-fields at once.

Figure 6.11: Request form

In addition to the progress bar, the forms also have buttons at the bottom to navigate through the form. These buttons are to go forward to the next part or backward to the previous part. This makes it easy for the user to move forward after filling in all the values. The forms also have a button at the top left to leave the form and go back to the main page again, when clicking this the user gets a warning that they are about to leave the form and they will have the option to cancel or continue.

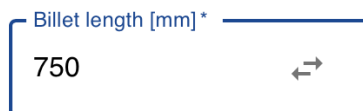
The parameters were divided into different categories, and each category has its own box with the corresponding text-fields (Figure 6.12). Each part in the progress bar contains one or more of these boxes.



The image shows a 'Profile' category in a request form. It contains five input fields: 'Alloy' (a dropdown menu), 'Temper' (a dropdown menu), 'Max Rex [um]\*' (a text input field), 'Max depth IGC [um]\*' (a text input field), and 'Surface quality [um]' (a text input field). Below these fields is a 'Ductility test req\*' (a text input field).

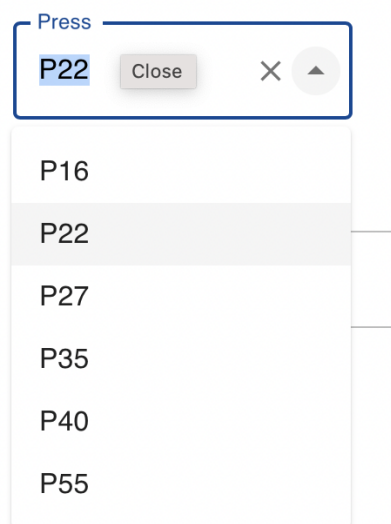
Figure 6.12: Profile category in the Request-form

The forms consist of many different types of text-fields, which are all Material UI components. The main types are drop-down-menus, input-fields and disabled-fields. The drop-down menus (Figure 6.14) make the user choose from a predefined set of values, and can be empty on default or have a default value. The input-fields (Figure 6.13) are fields that the user can type into, some of these fields are restricted to numbers only which means the user can only type in a number. The last type of field is the disabled field (Figure 6.15), these fields contain values from data being sent to the form or from formulas calculating its value.



The image shows an input field for 'Billet length [mm]\*'. The field contains the value '750' and has a clear button (X) and a swap button (↔).

Figure 6.13: Input field



The image shows a drop-down-menu field for 'Press'. The field is currently displaying 'P22'. The menu is open, showing a list of options: P16, P22 (highlighted), P27, P35, P40, and P55. The field also has a 'Close' button, a clear button (X), and a dropdown arrow.

Figure 6.14: Drop-down-menu field



Gross extr length [m]\*

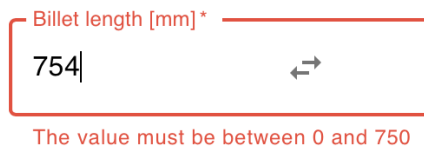
29,06

↔

The image shows a text input field with a light gray border and a light gray background. The text 'Gross extr length [m]\*' is at the top left. The value '29,06' is entered in the field. A double-headed arrow icon is on the right side of the field.

Figure 6.15: Disabled field

In addition to this the fields also have different types of properties depending on the parameter. Some will have warnings that pop up if the value is outside of a certain limit (Figure 6.16). Another property on the fields is that some of the number fields have a button to view the value in another unit (Figure 6.17). For example if the user writes a value in millimeters and wants to view the value in inches, he can hover over this button.



Billet length [mm]\*

754|

↔

The value must be between 0 and 750

The image shows a text input field with a red border. The text 'Billet length [mm]\*' is at the top left. The value '754' is entered in the field, with a cursor at the end. A double-headed arrow icon is on the right side of the field. Below the field, a red warning message reads 'The value must be between 0 and 750'.

Figure 6.16: Input-field with a limit warning.



Butt end [mm]\*

25|

↔

25 mm to in: 0.98

The image shows a text input field with a blue border. The text 'Butt end [mm]\*' is at the top left. The value '25' is entered in the field, with a cursor at the end. A double-headed arrow icon is on the right side of the field. Below the field, a blue tooltip box contains the text '25 mm to in: 0.98'.

Figure 6.17: Input-field with conversion to inches.

The last part of each form is the confirm-page. This page was added so the user could have an overview of the filled in text-fields before the data is sent to the database. On this page, the user will receive an error on each text-field that must be filled in and is not, and the send button will be disabled until all these fields are filled. The user also has the opportunity to change values or fill in an empty field on this page, but the fields are only updated after the user exits the field to conserve the performance of the application. In addition to a send button, the confirm page also has a button to download the data to an excel sheet. This option allows the user the opportunity to save the data locally on their computer as well as it is a option to secure the data if the data is not successfully sent to the database. In the case of the data not being sent to the database, the user will receive a message and stay on the confirm page.

The biggest form is the TFC form (see Figure 6.18), which is unique by having a summary-box to the left of the page. This is because the TFC form consists of a lot of formulas and many of the fields are dependent on each other. With this reasoning, the product owner wanted an effective way to view earlier values in the form while keep moving forward in the progress. We thought that adding a summary-box to solve this could

be a good solution that would still conserve the applications performance. The summary-box is filled with every Request and TFC value on the form, and is updated every time a field is changed, the user can use this to view earlier values and values that are forward in the form.

**Request values**

Request	Values
Name:	
Request number:	8005
Request number:	1
Requester name:	testSurface
Product controller:	test@test.com
<b>Project</b>	
Customer:	BAS Tender
SOP:	2
Aug. volume by:	2
Project name:	1
Place of delivery:	M17 (Bg 53)
Peak volume by:	
Project no (PPA):	21
Project nr SAP:	2
Volume total:	1
<b>Part</b>	
Part description:	Beam
Part category:	Beam
Parts pr vehicle:	3
Part length (mm):	224
SAP/CAD model:	2
<b>Material and Tolerances</b>	
Alloy:	6005.63
Temper:	4
Customer standard:	D6L4919.20
Geometric standard:	EN12020-2
Surface quality:	

**Die**

Currency: [US\$] Die type and size: [2] Bore: [2] Die cost: [2] Bore cost: [3] Mandrel set: [2] Run rate: [43] Die life: [2]

Blank bend (°): [3] Net length (mm): [-217] Die cost (USD/kg): [-0.05]

**Saw**

Prof. width on saw table (mm): [2] Profile length (mm): [3] Number of saws: [3] Die thickness table (mm): [8.5] Die time (saw table) (min): [188.8]

Number of pallettracks: [36100] Side by side: [ ] Spacers per pallettrack: [32]

**Packing**

Pallettrack type: [LL\_P55] Profiles per layer: [950] Profiles per pallettrack: [36100] Spacers per pallettrack: [32]

Crash weight pallettrack (kg): [91224]

Figure 6.18: TFC-form

### 6.3.5 Edit Database page

The Edit Database page (Figure 6.19) is a page only the administrators have access to. The page was made as a way for the administrator to add or change values that is used in formulas and drop-down menus. The page is split into three parts: Press, Alloy and Others.

	Press	Max length	Billet diameter	Acceleration timeH	Acceleration timeO	DCT time	Die change time	Table length	Container diameter	Max CCD	Mandrel area6xxx	Mandrel area7xxx
DELETE UPDATE	P16	700mm	178mm	10 s	8 s	15 s	60 s	36m	185mm	139mm	6000mm <sup>2</sup>	4500mm <sup>2</sup>
DELETE UPDATE	P22	750mm	203mm	20 s	15 s	19.4 s	90 s	42m	210mm	158mm	7000mm <sup>2</sup>	5000mm <sup>2</sup>
DELETE UPDATE	P27	940mm	203mm	10 s	8 s	22 s	120 s	44m	213mm	160mm	7000mm <sup>2</sup>	5000mm <sup>2</sup>
DELETE UPDATE	P35	965mm	254mm	10 s	8 s	26 s	120 s	42m	264mm	198mm	8000mm <sup>2</sup>	6500mm <sup>2</sup>
DELETE UPDATE	P40	950mm	280mm	10 s	8 s	20 s	90 s	43m	288mm	216mm	9800mm <sup>2</sup>	7000mm <sup>2</sup>
DELETE UPDATE	P55	1500mm	280mm	8 s	5 s	17.5 s	90 s	63m	289mm	217mm	9800mm <sup>2</sup>	7000mm <sup>2</sup>

Figure 6.19: The Edit Database page

### Press

The first part of the page is the Press part. This part consists of a table containing all presses and their values. The presses are used in the TFC form in a drop-down menu, and the values corresponding to each press are used in formulas and limits. The administrator has the option to delete or change the values on existing presses. The administrator also has an option to add a new press with the corresponding values. When updating or adding a press a popup menu (see Figure 6.20) appears where the user can input the correct values.



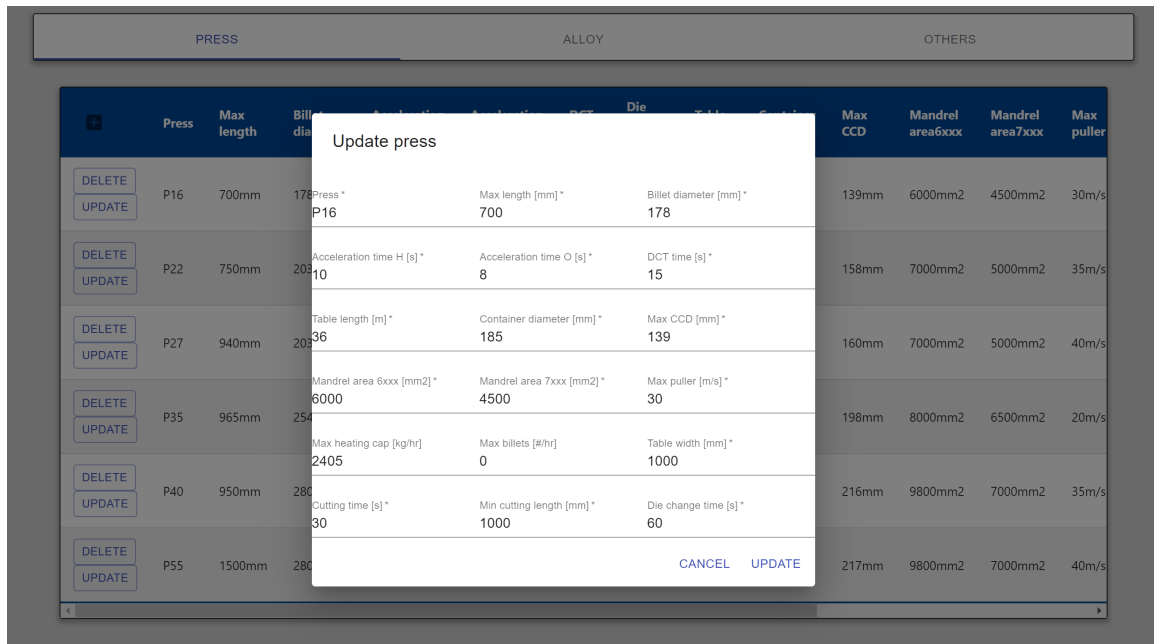


Figure 6.20: Popup to update a press

### Alloy

The next part of the page is the Alloy part (Figure 6.21). This part consists of an alloy table, with the alloy number and corresponding density, a box to add new alloys and a text-box explaining the "Reduction Ratio Recommendations". The alloy values are used in all three forms and are presented in a drop-down menu similar to the press values.

The screenshot shows the 'Edit Database - Alloy' interface in the Extrusion Planner. At the top, there are tabs for 'PRESS', 'ALLOY', and 'OTHERS'. Below the tabs is a table with the following data:

	6005.19	6005.24	6005.40	6005.63	6005.71	6005A	6005BU	6010.57	6060	6060.06	6060.35	6060.85	6060low
<b>Alloy</b>													
<b>Density</b>	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7

Below the table is a form to 'Add new alloy' with fields for 'Alloy \*' and 'Density \*', and a '+ ADD' button. A text box provides 'Reduction Ratio Recommendation' guidelines:

**Reduction Ratio Recommendation**  
**General guideline for extrusion planning of alloy 6xxx:**  
 0<RR<40: **BIG PRESS COMPETITIVE.** Is recommended to extrude the profile in a bigger press.  
 35<RR<65: **WINNER.** The press used in the calculations is the optimum for extruding this profile.  
 65<RR<80: **ALSO GOOD.** The press used in the calculations is good for extruding this profile.  
 80<RR: **SMALL PRESS COMPETITIVE.** Is recommended to extrude the profile in a smaller press or by multicavity.  
**General guideline for extrusion planning of alloy 7108 or higher alloyed hollow sections:**  
 0<RR<25: **BIG PRESS COMPETITIVE.** Is recommended to extrude the profile in a bigger press.  
 25<RR<45: **WINNER.** The press used in the calculations is the optimum for extruding this profile.

Figure 6.21: Edit Database - Alloy

### Others

The others part (Figures 6.22 and 6.23) consist of the other parameters used in the drop-down menus and formulas that are not in the press or alloy part. It consists of 10 lists where the user has the option to delete or add new values to each list. The lists appear in drop-down menus in different forms.

The screenshot shows the 'Edit Database - Others' interface in the Extrusion Planner. The interface has tabs for 'PRESS', 'ALLOY', and 'OTHERS'. Below the tabs is a grid of 10 drop-down menu boxes, each with a plus sign for adding new values:

- Cooling
- Customers
- Profiletype
- Geometric Standards
- Temper
- Deliveryplace
- Customer standards
- Contentrack
- Spacers
- Product Controllers

Figure 6.22: Edit Database - Others

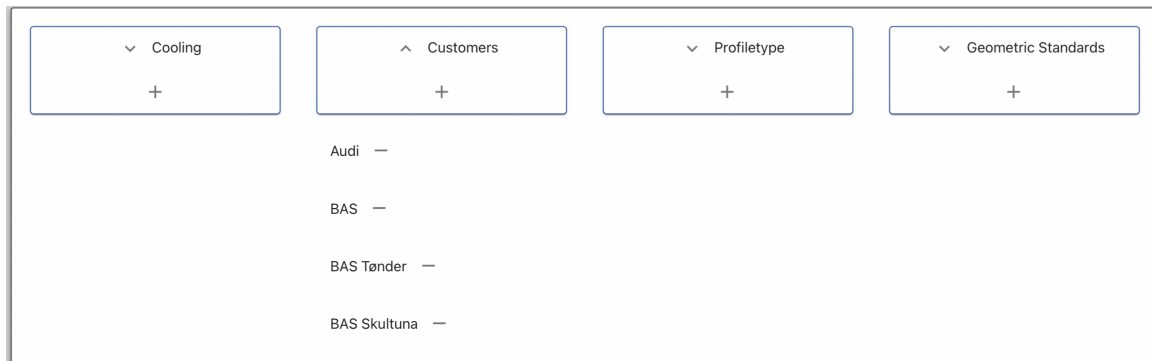


Figure 6.23: Edit Database - Others with Customers list opened

### 6.3.6 Users page

The users page (Figure 6.24) is another page only for the administrators. This is the page where the administrators can add, update or delete users. The design of the page is kept very simple and it only consist of a table containing all the different users and their roles.

UUID	Name	Role/Permissions	
ddf0f06f-00c9-4be0-9c8b-a060a3486ef5	Jesper	admin	DELETE UPDATE
df410950-31e6-4879-b7da-c13371bdfeba	ola.nordman@test.com	admin	DELETE UPDATE
dc25f858-8e75-4d57-81fe-083acc8c5343	emma@admin.com	admin	DELETE UPDATE
33360183-d2c3-4d3d-be81-24594eb27f19	navn.navnesen@domain.com	admin	DELETE UPDATE
8ed93291-2da8-4ede-96bc-8480de299663	navn.navnes.navnesen@domain.com	admin	DELETE UPDATE

Figure 6.24: Users page

Every row on the table has an update and a delete button, so that the administrator can easily update or delete the desired user. At the top right of the table is there also an add button that the administrator can use to add a new user to the system. The update and add buttons open a popup-menu (see Figure 6.25), where the administrator can input the users email address and temporary password as well as choosing a role for the user. When an administrator clicks the delete button, a popup asking if they are sure will show up.

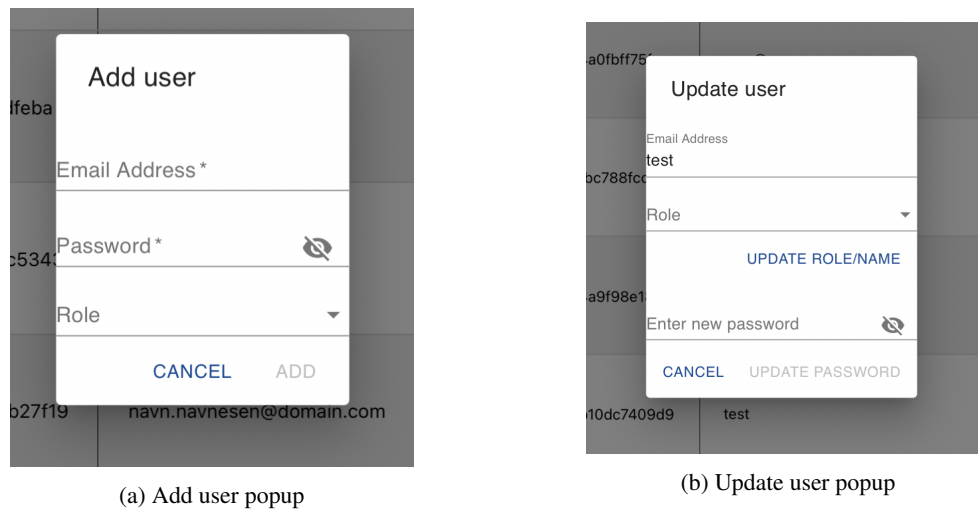


Figure 6.25: The user popup menu

### 6.3.7 Home and FAQ page

The Home and FAQ page was not a part of the functionality requirements, but it was something we decided to add to improve the design and user experience.

The home page (Figure 6.26) is the first page the user sees after login. It is also the page that is returned on every reload for safety reasons. The home page consists of a welcome message wishing the user a good morning, afternoon, evening or night depending on the time of day. It also shows a big version of Benteler's logo.



Figure 6.26: Home page of the application

The FAQ page was implemented to provide answers to frequently asked questions or anticipated questions regarding the application or process. The product owner liked this idea, and they wanted to contribute to creating questions and answers that could be added to this page after the testing phase. The FAQ page can be seen in Figure 6.27. We decided to use ExpansionPanels supplied by the Material UI framework. This provided us with a sleek interface where all the answers to the questions displayed is hidden to save space until the user clicks on one and displays the corresponding answer.

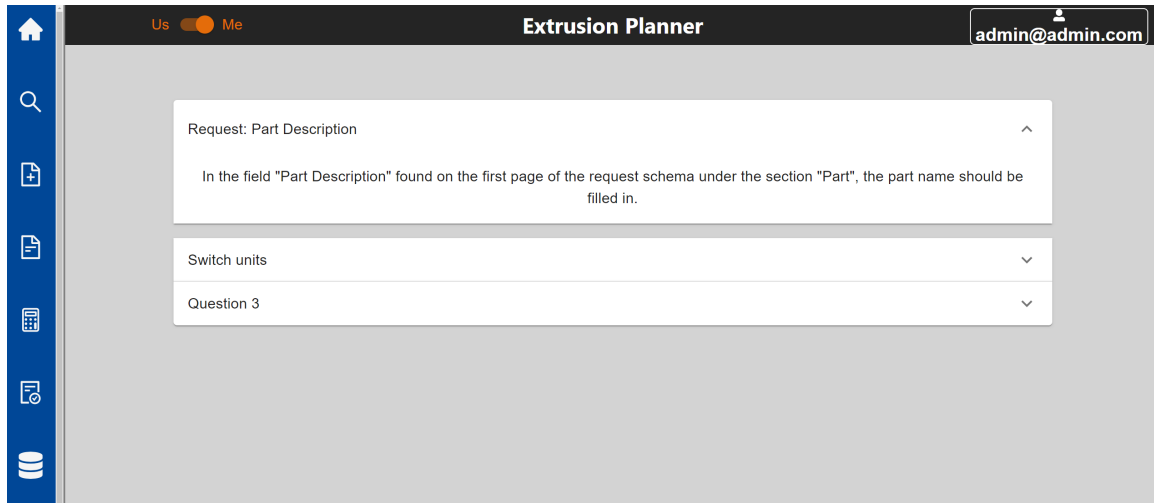


Figure 6.27: The FAQ page displaying questions and answers in expandable UI boxes

## Chapter 7

# Overview of the development environments

In this chapter we explore the environments and tools used to facilitate the development of the application. The environments had to allow for effective and fast testing of newly developed code as well as being able to easily deploy the code to both the testing server and the deployment server. In Figure 7.1 we see an overview of all the environments and tools we have used for this project.

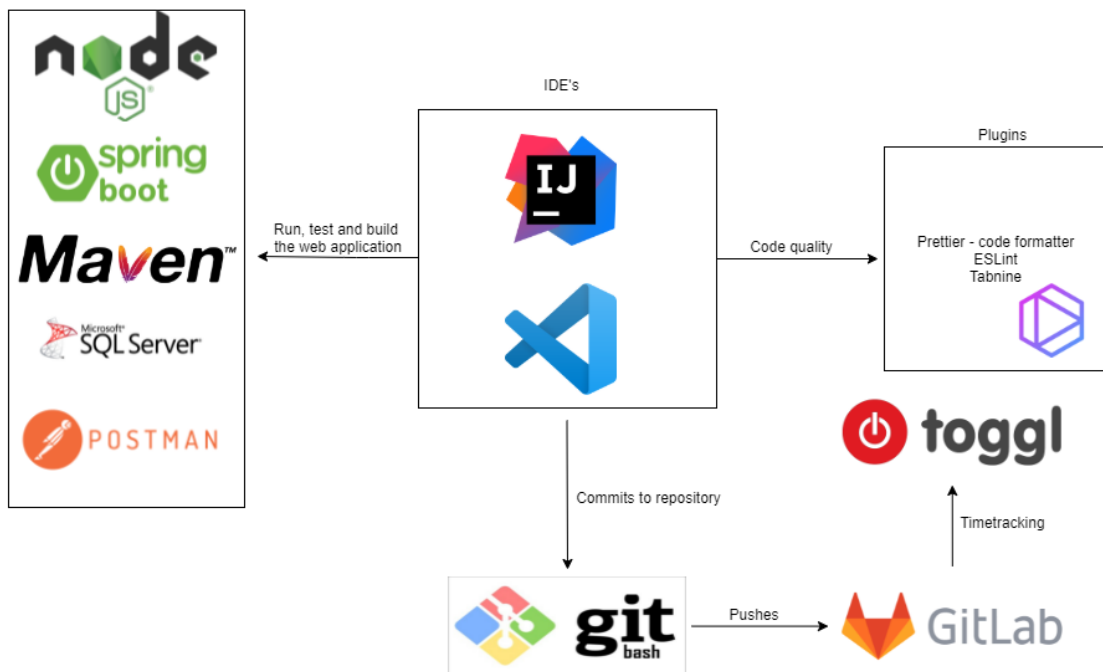


Figure 7.1: Overview of the development environments

## 7.1 Frontend

For the development of the frontend web interface, the team decided to use Visual Studio Code<sup>1</sup>. This is a great tool that integrates a lot of neat features that help in the development process. Like integrated Git<sup>2</sup> for version control, terminal window to perform npm commands from the NodeJS API<sup>3</sup>, as well as the numerous plugins available for VS Code to make developing code faster and easier.

The NodeJS framework allowed us to continuously run and test the frontend web interface locally on the machine with a local development server. This local server continuously updated every time a new code change was saved so any changes made could be seen in real time. This made learning React, HTML and CSS much easier as results could be seen right away.

NodeJS also served as a tool to build the project into a production optimized build ready to be deployed on the web server. To deploy to the deployment server we used the command to create a build, and then replaced the old build folder followed by a restart of the web page in IIS<sup>4</sup>. As doing this procedure was only necessary when updating the production server, the team chose to save time by doing this manually rather than creating automated scripts for this.

## 7.2 Backend

As the application's backend API was chosen to be developed in Java, the team chose to utilize the go-to IDE (Integrated Development Environment) for Java development called IntelliJ<sup>5</sup>. In conjunction with IntelliJ, we used Apache Maven<sup>6</sup> as a project management and comprehension tool. Maven served us with a structured way of managing dependencies, dependency versions, custom configurations and generally most parameters regarding the whole API. The product of the Spring Boot Initializr<sup>7</sup> project generator was already a maven project, which saved time.

On the OpenStack<sup>8</sup> testing server supplied by NTNU, we also simplified the process of deployment of development builds. We decided to not keep building the JAR or implement a solution like Jenkins<sup>9</sup> to automate the process of deploying the JAR. Instead we installed Git and IntelliJ so new code easily could be pulled from the central GitLab repository, and then run the backend API code within IntelliJ using its built in capabilities. This also provided us with the opportunity to test certain API code on the testing server first before pushing tested and working code to the repository when finished.

When deploying the backend API to the deployment server we tried to build the JAR and then run it on the machine. We quickly figured out that this was the wrong approach, due to the fact that the windows server supplied as our deployment server automatically logged out the user after exiting the remote desktop. This meant that we had to install the API as a service instead by using WinSW Service Wrapper<sup>10</sup>. WinSW allowed us to have the API running as a service continuously without interference, while still keeping the API output available in a dedicated file.

This allowed us to build the API locally using maven, and then replacing the old JAR file on the deployment server followed by a restart of the API service. This can be read more about in section 5.4. As this was

---

<sup>1</sup><https://code.visualstudio.com/>

<sup>2</sup><https://no.wikipedia.org/wiki/Git>

<sup>3</sup><https://nodejs.org/en/>

<sup>4</sup>[https://en.wikipedia.org/wiki/Internet\\_Information\\_Services](https://en.wikipedia.org/wiki/Internet_Information_Services)

<sup>5</sup><https://www.jetbrains.com/idea/>

<sup>6</sup><https://maven.apache.org/>

<sup>7</sup><https://start.spring.io/>

<sup>8</sup><https://www.ntnu.no/wiki/display/skyhigh/Openstack+at+NTNU>

<sup>9</sup><https://www.jenkins.io/>

<sup>10</sup><https://github.com/winsw/winsw>

a process only done when the team decided the new additions to the application were significant enough to demand a redeployment, we felt this solution was sufficient and efficient enough.

### **7.3 Database**

As the team used Microsofts SQL Server, using the Microsoft SQL Server Management Studio<sup>11</sup> (*SSMS*) was the obvious choice. *SSMS* provided us with easy interfaces for developing the database with tables and user permissions. The built in table designer, data editor and SQL Query editors and executors came in handy as the database tables had to be manipulated regularly, following the changing requirements and needs of the client. For example, when we needed to increase the size of certain columns due to a change in permitted values in the frontend, we could increase its size just using SQL alter table queries.

---

<sup>11</sup><https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>



## Chapter 8

# Code quality

Code quality has been important to us during this whole development process. After we were done developing, Benteler's IT-department took over maintaining and further development. They have limited coding experience, and need the code to be readable and understandable.

### 8.1 Code review

One of the methods we used to ensure good code quality was code review of other team member's code. We had a column in the Trello-board (Figure 2.2) called QA (Quality Assurance) where every Trello-card is moved after the team member responsible for that task had finished it. Every Trello-card had a checklist and a description of the functionality and in which files the changes were made (as seen in Figure 8.1). The card could not be archived until every team member had tested the implemented functionality and understood the code. Every section of the report has been through Quality Assurance from every team member.

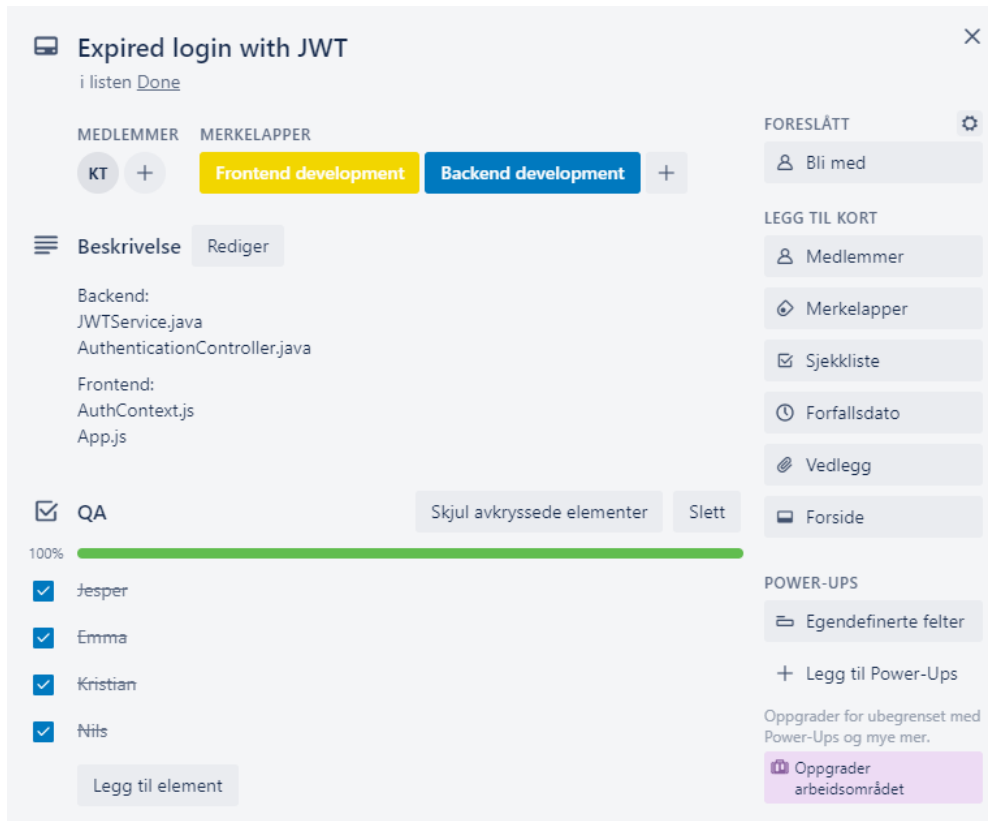


Figure 8.1: A card with QA in trello

## 8.2 Code documentation

Almost all functions in the web application are documented with a docstring<sup>1</sup>. This is a type of comment which is parsed by the editor/compiler and retained during the run time of the program. It makes the documentation more interactive and can be accessed across all modules. Docstrings in Java is called Javadoc [15] while in JavaScript they are called JSDoc [2].

```

1 /**
2  * A short description of the function goes here.
3  * <p>
4  * A more detailed description follows here if necessary. That depends on
5  * the size and how advanced the function are.
6
7  * @param nameOfFirstParam A short description of the first parameter, if
8  * it exist. Followed by further * parameters.
9  * @return If the column returns anything, it should be described here. If
10 * possible, it should also mention what will be returned in special cases or
11 * if the function fails in any way.
12 */

```

Listing 8.1: Example of docstring in Java

<sup>1</sup><https://en.wikipedia.org/wiki/Docstring>

Some functions in the application are not documented with docstrings. These are mainly small/self-explanatory functions. An example could be a getter in the backend, where the function's only mission is to publicly return the value of a private variable. In those cases our focus was to make the variable names and function names descriptive.

Figure 8.2 shows an example of what a docstring looks like when you hover over a documented function in the IDE. In our case this will be helpful for the team taking over any further development of the application, since it works like a contract where the next person to use the function knows what to expect.

```

780     Alloy: true,
781     Temper: true,
782     Surface_quality: true,
783     ...
784     (local function) change(): void
785
786     Function to perform the necessary modifications of the states to allow for a user
787     to change a request
788
789     It sets all optional, permanent and confirm states to a custom ruleset The code
790     inserts the required fields to be edited to enable the submit button into
791     mustChange The code listens to this const to activate the button Request
792     number is then set to the latest request number from the database +1.
793     @state — {array} selectedData - containing all the values for a chosen request
794
795     change();
796     setPermanent({
797         ..permanent,
798         Customer: true,
799         Project_name: true,
800         Project_SAP: true,
801         Part_description: true,
802         Part_category: true,
803         Sketch_profileno: true,
804         Profile_comments: true,
805         Request_comments: true
806     })

```

Figure 8.2: Docstring interface example

## 8.3 Frontend

We have used a couple of plugins to assure that the code quality was on point. As mentioned, the IDE we have used is VScode, which offers a broad selection of linters and extensions to help the developer. A linter analyses the code, find problems and suggests what can be done. We have also used a code formatter to ensure our coding styles and formatting is the same through the entire frontend of the application, which makes the code easier to further develop and maintain. For coding effectivity we have used tabnine.

- ESLint - A pluggable linter for JavaScript used to secure good coding practices and remove problems that can cause a drop in performance. Comes with a set of rules for good syntax [32].
- Prettier - A VScode plugin that automatically formats code so all code is formatted in the same way [30].
- Tabnine - A powerful AI assistant that helped us a lot with coding faster, reduce mistakes and discover better coding practices [33].

## 8.4 Backend

In the backend we have used a plug-in named SonarLint [11]. This linter is quite similar to ESLint, and also has a set of rules that help make the code as good as possible. It also gives an interactive view while coding, where it squiggles flaws so they can be fixed before running the code through the debugger.

## 8.5 Code redundancy

As the team faced a new challenge in form of the unexplored coding language, React, it led to us not immediately be able to use Reacts capabilities to its full extent. This applied to the big input forms in the web interface, where the user could input TFC, Request and Order data. As a solution to keep pace in the project, the team decided to develop similar code for all the pages of a form as on the last confirmation page. This meant that code had to be changed at two locations to update the form or three places as the Request form is displayed in the TFC form as well. We later figured out we could have developed components instead, which could have taken in parameters to customize the input fields that way. Implementing this would have saved lines of code, and possibly performance too but that would be negligible.

### 8.5.1 Large SQL queries in the API

The forms used in the application were rather large, which led to us having to write large SQL queries when saving or updating/modifying a form. This led to the need to update every SQL query when a new variable/parameter was introduced to that specific form. We explored solutions to simplify these SQL queries but it was not recommended as it would make us rely on the exact layout of the tables. This was deemed a rather bad approach by several members of the StackOverflow community [5], so we stuck with what we had implemented.

### 8.5.2 Large ResultSet to object conversions and prepared statements value bindings

Following the SQL Queries, we also had to convert the result set returned from the database into usable objects by our application, whenever we wanted to retrieve data from the database.

Having a layer of security by using prepared statements in the API, we also had to use large sections where we set a certain parameter by its index in the prepared statement. As the biggest form, TFC, consists of 96 parameters, we had to bind 96 parameters for the necessary prepared statements. The most important issue with this is again having to add/delete/change the same parameter multiple places in the code, which can be time-consuming and could lead to errors if not done correctly. We tried to explore the internet for any libraries that would simplify the SQL process in Java but we were not successful in finding any.

## Chapter 9

# Testing

To be sure that the application is stable and working as it should, we have through the whole development process tested the application in different ways. The team researched to find the best practices for web applications, and we found out that some of the most important ones for our application were acceptance testing, unit testing, compatibility testing, and performance testing [36].

### 9.1 Cross-browser compatibility

The web application was tested in Google Chrome, Safari and Microsoft Edge, and we found out that the elements were being displayed properly in all those browsers. We also found out that requests from all the different browsers worked as expected. We did not prioritize testing the compatibility on mobile devices, with the reasoning that the application is only going to be used on computers.

### 9.2 Acceptance tests

Acceptance testing also refers to the type of testing done by the users of the application <sup>1</sup>. This is one of the most important steps of the testing phase, and is the final stop in the testing phase [27]. In sprint 5 that started March 31 we started on this type of testing. In this period we asked Benteler to test the application to see if they could find any minor or major faults that we have not been able to see. The people at Benteler created an excel sheet where everybody could add things they came over consecutively.

Most of the faults added by Benteler were quite easy for us to fix, and we chose to not implement some of them due to lack of time and that they were not major faults. Many of the faults were about different wishes they had regarding the forms. Examples are adding new text-fields, implement drop-down fields, implement autofill from the username, which characters that should be allowed to type in and more.

Another thing that was added to the list was a few wishes for new functionality. An example of this was implementing a button located in the confirm pages that prints all the values typed into excel. This we considered an important functionality because if the API is not reachable they need a backup solution to save the values, so that they do not lose everything.

Another part of the faults was about the functions in the TFC form. Here we got feedback that a couple of functions did not calculate the correct values, some limit values were wrong and wishes to highlight certain important values.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Acceptance\\_testing](https://en.wikipedia.org/wiki/Acceptance_testing)

Another example of things Benteler added to the list was which questions we should add to the FAQ page. This helped the FAQ page becoming as effective as possible in helping new users.

We also gave Benteler an overview of two different designs for the application, and asked them which design they preferred. They preferred the design with a grey background, and this was implemented. The whole excel sheet "LOP action Item List can be found in appendix C, but in Figure 9.1 you can see an excerpt from this excel sheet.

39	4/23/2021	GLN / KIA	TFC	Side 5 under Die - Mandrel set må ikke være et obligatorisk felt	Fikset	30/04/21	Emma	Done
40	4/23/2021	GLN / KIA	Order	Oversikten / search tabellen for request nr må vise kun de request der det allerede finnes en TFC, ellers kan man ikke bestille på basis av request nr	Viser nå bare requests der det allerede finnes minst en TFC.	30/04/21	Jesper	Done
41	4/23/2021	GLN / KIA	Order	Request nr, request versjon eller order nr SKAL ikke kunne endres, du velger først request nr og versjon, mens ordre nr autogenereres	Satte disse disabled	30/04/21	Jesper	Done
42	4/23/2021	GLN / KIA	Request	Sketch nr skal ikke kunne endres, det er låst dersom en kjører update request	Sketch nr er nå låst på menyvalget "Correct Request"	30/04/21	Kristian	done
43	4/23/2021	GLN / KIA	Request	Recalc: SAP nr skal være låst, kan få lov til å skrive i feltet bare hvis verdi mangler	SAP nr er låst hvis felt er fylt ut, men har satt det slik at feltet kan endres hvis felt er tomt, inneholder "?" eller "na"	30/04/21	Kristian	done
44	4/23/2021	GLN / KIA	Request	Recalc: Comment felt er ikke aktivt, får ikke lov å legge til kommentar, det må vi kunne gjøre	Comment feltet er aktivert for endring på ReCalc og Change	30/04/21	Kristian	done
45	4/26/2021	GLN	Request	Når jeg taster inn verdi i et felt og forslag popper opp. Eksempel skriver Au i felt for kunde, da kommer 3 forslag opp, bruker jeg piltast ned til riktig forslag og tab for å velge og gå videre så velges ikke dette slik som i excel eller andre skjema, jeg må aktivt klikke på alternativet for å få det valgt. Kan vi få "Tab" til å virke som enter når feltet er endret? Dvs på volum så ser det ut til at tab virker slik men ikke på drop lister muligens...	Vi skal se på hva muligheter vi har. Feltene på skjemaene er Material UI komponenter, det er et framework vi har tatt i bruk for visse deler av applikasjonen for å få bedre design og brukervennlighet enn det vi foler vi hadde klart å få til uten framework. Vi skal se litt om det går an å endre bruken av tab, men siden dette er en ferdiglagd komponent er det ikke sikkert vi kan gjøre det. Dette er nå implementert.	30/04/21	Jesper	Done

Figure 9.1: Excerpt from LOP action items list

### 9.3 Lighthouse

To test the performance of the application we have used an open-source, automated tool called Lighthouse [35]. Lighthouse can be run against any web page, and generates a report for performance, accessibility and best practices.

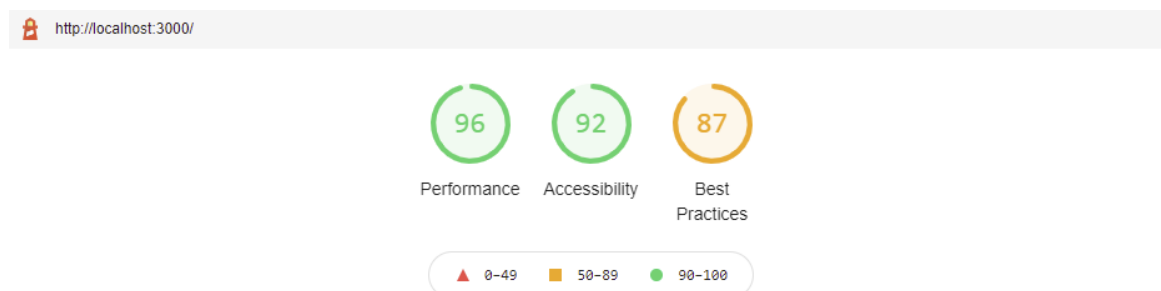


Figure 9.2: Our lighthouse score

As you can see in Figure 9.2, our Lighthouse scores were to satisfaction. The performance score is a weighted average of different metrics. The most weighted metrics are total blocking time, largest contentful paint and time to interactive which in total is two thirds of the score. The only issue Lighthouse sees in its report is that the Benteler logo used in the footer does not have explicit height and width.

On accessibility we got a score of 92 out of 100. Accessibility, in Lighthouse setting, refers to the experience of users outside what would be considered a "typical" user. This would be someone who accesses and interact with the application in another way than what we initially think of. It can also be other issues like struggling to navigate the desktop site on a mobile phone [3]. To receive a good score on accessibility on Lighthouse it is important to name labels, pictures and buttons correctly, so it helps people using screen

readers. Although this was not a functional requirement from Benteler we did our best to keep it in mind, since it is a good coding practice.

The last part of the report, best practices, was our lowest score of the three. The main part Lighthouse points out is the lack of HTTPS. This was not prioritized enough to complete within the time frame, since the application is only available through a secure VPN or from within their internal network.

## 9.4 Traffic testing of the server

We also wished to test how much traffic the server could handle under pressure. This type of testing is important because it can identify inefficient code and it can make customer satisfaction better because they become aware of the limitations of the system [28]. To test this we used an HTTP benchmarking tool called Autocannon [4]. This is a tool written in node, and it allows us to generate many requests to the web server every second.

We chose to perform this type of test on our test server provided by NTNU because of the fact that the tool generates many requests every second and this can cause the server to crash or getting slower. We compared the two servers before starting this test, and we found out that it is minimal differences between the two servers. Both servers utilize Intel Xeon CPUs and both have 4 GB of RAM. Therefore the results we got are easily comparable to the production server.

We ran the test for 30 seconds each with 2 minutes break between all the API endpoints. In listing 9.1 we see the command we ran in our terminal to execute the test. This command starts generating many GET requests with 150 connections (-c) for 30 seconds (-d).

```
autocannon -c 150 -d 30 http://129.241.150.48:8080/api/...
```

Listing 9.1: Command to run traffic test

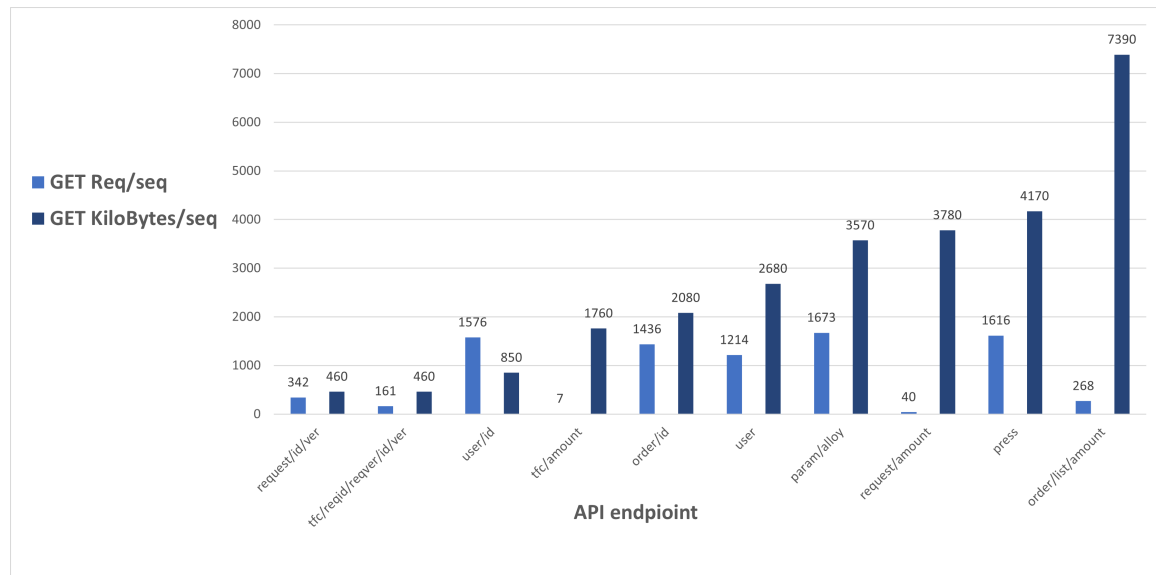


Figure 9.3: Results of traffictesting using autocannon

In Figure 9.3 we see the results of the test. We can see that the GET requests per second vary from 1673 at the most, to only 7 per second at the least. This has a logical explanation because the amount of data that is retrieved during these requests varies a lot. That the server only handles 7 GET requests per second on the `tfc/amount` endpoint is because this endpoint retrieves 100 TFC's per GET request if the parameter is 1 (that we used in this test), and this is taking time to retrieve. This is shown in the results by that although this endpoint only handles 7 GET requests per second it retrieves 1760 kilobytes per second which is a good result. This also applies to the `request/amount` endpoint that only handles 40 GET requests per second, but retrieves 3780 kiloBytes per second. It is also important to remember that there are not so many users that have access to the TFC page, so the fact that the function that gets TFC's have the lowest GET requests per second score will not be a problem. We also have to mention that the explanation for that the amount of kiloBytes retrieved per second for one request, one TFC, and one user is quite low is due to what is returned does not contain many bytes.

Further on we see that many of the API endpoints handle over 300 GET requests per second, and that the amount of kiloBytes retrieved per second for many API endpoints lies on over 800 kiloBytes per second. This is a result we are satisfied with, and is a result that will satisfy Benteler's needs by a good margin.

## 9.5 Unit tests

Unit testing is a type of software testing where individual components/functions are tested to see if they are performing as expected. We chose to use this testing method because it is a great way to find and fix bugs early in the development process which leads to saving time [14]. We have used Postman [29] to test the backend API's correctness. Postman is a tool to help building your API faster and better. We have used the Postman desktop client mainly to test the response of the API to ensure the functionality is working before implementing the functions in the frontend.



The screenshot shows a Postman interface for a GET request to `http://129.241.150.48:8080/api/press/p55`. The response status is 200 OK, with a response time of 175 ms and a size of 787 B. The response body is a JSON object with the following properties:

```

1  {
2    "Press_name": "P55",
3    "Max_length": 1500.0,
4    "Billet_diameter": 280.0,
5    "Acc_timeH": 8.0,
6    "Die_change_time": 90,
7    "Table_length": 63,
8    "Container_diameter": 289,
9    "Max_ccd": 217,
10   "Mandrel_area6xxx": 9800,
11   "Mandrel_area7xxx": 7000,
12   "Max_puller": 40,
13   "Max_heating_cap": 0,
14   "Max_billets_hr": 22,
15   "Table_width": 1200,
16   "Cutting_time": 26,
17   "Acc_time0": 5.0,
18   "cutting_length": 700,
19   "dct_time": 17.5
20 }

```

Figure 9.4: Postman example: Testing `getPressByName` functionality

Inside the green rectangle in Figure 9.4 we see the API path to fetch the press named P55. In the red rectangle, we see the info regarding the API request. Firstly is the status code. A status code 200 simply means OK and that the request was received and understood. The time is the response time, the time from the request is sent to the response is returned. Size is the size of the returned object. The blue rectangle contains what is returned from the API. In this case the response consists of a JSON object with all press P55's parameters. The response can be in many forms, depending on the code in the API. For example for a POST request, meaning adding something to the database, the response could be "true" if successful or "false" if something goes wrong.

After the function had been tested in Postman, we knew the API was working. The next step was then to call the function in the frontend. In this case the function would look like this:

```

1  /**
2   * Function to retrieve a specific press based on its name.
3   * @variable {String} API_URL The server ip-address.
4   * @param {String} name The name of the press to be returned.
5   * @returns {JSON} The press object.
6   */
7  getPressByName = (name) => {
8    return axios.get(API_URL + `press/${name}`);
9  }

```

Listing 9.2: Function to retrieve a specific press

# Chapter 10

## Conclusion

### 10.1 Summary of contributions

Benteler wanted an application that would replace the old Excel-based program and exceed in user-friendliness, ease of use, user experience and meanwhile implementing new features and conveniences. They wanted a more integrated and complete experience, where everything was centralized. We definitely feel we delivered on these aspects. After the end of project, we had finished a complete application that delivers a flexible experience compared to the old solution. First of all, the only required step to begin using the application is to enter the address in the web browser, where the user is greeted with the same [interface](#), [data](#), [parameters](#), [forms](#) as everyone else, leading to a consistent experience. The old Excel was more fragmented in terms of not being able to update forms/formulas/parameters in the document without having to distribute it again to everyone.

The old Excel program did not have any form of security, an area where we did not receive specific requirements, regardless we knew it had to be dealt with in the application. First, we implemented an [authentication system with multiple roles](#), making sensitive parts of the system not accessible to everyone with access to the web interface. Then we implemented security in form of password hashing and [JSON Web Tokens](#) to avoid session hijacking and other security risks associated with using server-side session tokens and client-side cookies with session IDs. As described in [5.2.7](#), the material UI components provided us with numeric and date validation, and we also implemented email validation. The most important security measure in the API is the use of prepared statements. We also implemented criteria to ensure strong passwords, that follows the OWASP Password Security Requirements [[13](#)].

A big and central part of the application is the [searching functionality](#), which was an important requirement from the client. We catered to this requirement by implementing a refined searching experience, that both looked good and is intuitive to use. It hosts great flexibility, like being able to [search for any parameters](#) and having as many filters active as there are parameters available. Contrary to the old program, we also saw the opportunity to automate as much as possible to create a consistent experience. We automated many of the starter fields in TFC and Request, where data is automatically filled in to save time. We implemented [automatic version control](#), which enumerate everything automatically based on the existing contents of the database. In addition, we implemented [automatic status control](#) and a way to [mark requests as invalid](#) instead of deleting and creating gaps.

To enable the extrusion process to happen, the requirements were to implement the three forms for data input: Request, TFC and Order. These forms started pretty simple after we received our first basic requirements. As time passed we soon received more detailed and advanced requirements regarding the forms, ranging from automatic version control, automatic input of data without user interaction, drop-down menus

presenting options stored in the database and regulated by the admin from the database page and so on. Later we received input regarding implementing formulas in the TFC form for automatic calculations of certain fields. We fulfilled these requirements, but we must admit that developing such a big form as TFC with all the automatic formulas attached was not entirely straightforward in React. We also encountered a couple of bugs after implementing the formulas, which were fixed after we received feedback. See appendix C.

As described in 5.2.3 the team did not have a full overview of all the functionalities of React in the beginning of the development process, so the first two forms were developed using classes instead of hooks. Since the rest of the application was chosen to be developed with hooks, we had some start issues with sending parameters between classes and functions. This was not such a big issue since we managed to find out how we could solve it quite fast. However, in our next development project we will decide which method to use before we start developing.

Another point we had to deliver, was being able to easily export the data from the application to a format accepted by the cost calculation team. We initially only knew that this was a requirement, but it took a while before the CostCalc team decided on what solution they felt would be the best. We discussed having the data be exported to an Excel document, storing the data in the SQL database for later retrieval or full integration with the application. The team determined that the last option was out of the question due to time constraints as this suggestion came rather late. The CostCalc team ended up deciding that exporting to an Excel format would be the ideal solution for the time being. Initially, we developed a simple export with data only in an empty sheet, but this was found to be inadequate. We then reworked the process to be done in the API which modifies their existing document by inserting all the data at the correct indexes in the document. The modified document is then sent from the API to the requesting user's browser and downloaded on the machine. This was found to be a good solution.

## 10.2 Feedback from Benteler

We wanted to receive written feedback from Benteler to evaluate our project and how we as a team have worked during these last months, to help us improve in the future. To reach this purpose, we decided to create a survey for the people who had tested the application at Benteler. The survey consisted of 16 questions, but some questions were meant for special user groups, so only half of the questions were mandatory. Unfortunately, Benteler only had the capacity for three people to answer the survey. The full survey can be found in Appendix J.

The survey mainly consisted of questions where the respondents rated different parts of the application from 1 to 5, where 5 is the best. We also added a long answer question where the respondent could elaborate on their previous answers. Table 10.1 consists of an extract of a couple of the key questions.

Table 10.1: Extract of survey result

Nr	Question	Average rating	Total answers
1	In your opinion, where does the new application compare in relation to the old application?	4,33/5	3
2	Considering your complete experience with our software, how likely would you be to recommend replacing it with the old software?	4,33/5	3
3	How satisfied are you with the solution of exporting data to a bridge file on the CostCalc page? (Only available for CostCalc, Superusers and Admin)	2,66/5	3

The answers to question 1 show that we have made an improved application, compared to the old one. We feel that the new solution outperforms their old in every way, but we agree with one respondent who writes that the application still needs a bit of bug-fixing.

Question 3 was something we predicted would receive a lower score than the rest. As mentioned in section 10.1, Benteler spent time deciding what kind of solution they wanted, which led us to getting a late start. We did not quite understand their proposed solution at first, so we did not start developing the cost calculation in the way they wanted before 22. April, when we had a meeting with Benteler dedicated to the cost calculation process. This led us to only have a week to finish the CostCalc functionality requested from Benteler. Eventually, this functionality was finished behind schedule on the 6th of May (which were after the survey answers). One of the respondents wrote:

"Other lack of functions caused by Benteler late or missing input. A more frequent cross check on planned solutions from students vs Benteler expectations before putting too much work into solving task could have increased the completion level a bit to date".

In hindsight, we should have made sure we were in complete understanding when they first decided what they wanted according to the CostCalc functionality, instead of starting developing.

### 10.3 Future Work

We feel that our finished product is complete, but like most development projects there is always something that can be improved. In the case of our application, we would like to improve the efficiency and latency of the forms. We would like to explore and find an alternative to how the data is stored and handled, as modifying the large TFC data buffer induced a bit of latency on the last confirm page. This is due to the fact that the confirm page displays all the fields. When anything in the state would be updated, all the 96 fields would re-render and cause latency. A workaround of this would be to split the buffer state into multiple parts and merge it before submission but we believe there is a better solution.

We would also like to decrease the amount of code written to create the forms, where we would rather make reusable and customizable field components. This component could then instead take all the necessary customization as parameters and thus limit the amount of code. Creating components containing the different sections with fields in the forms would also be done, to re-use these in the confirm pages without having to have the sections defined in two places.

To further improve the security of the application, we wanted to finish implementing HTTPS and adding salt to the passwords to improve the obscurity of the passwords rather than using hash only.

## 10.4 Learning Outcomes and Concluding Remarks

In the past, none of the team members had ventured into the world of web development. At first, it seemed like a big task to dive into a completely new way of making applications while simultaneously having to execute the bachelor thesis project in the best way possible. It was tempting to go for something known, but the team instead saw this as a wonderful learning opportunity! Choosing to utilize web development to solve the task gave us two things; experience within an important development field and the premises to develop the best solution suited for the client's needs. We learned to develop an application that could be reached from anywhere, requiring only a device and an internet connection. We learned to develop intuitive solutions, good designs, user experience, data management, inter-connectivity with an API, routing, navigation within the application, and much more. In hindsight, this was the right and obvious choice and has now equipped us with the necessary tools to combat more diverse tasks in the future.

# Bibliography

- [1] jiminikiz & benng. *convert-units*. <https://www.npmjs.com/package/convert-units>. [Online; Cited: 10.05.2021].
- [2] *A Guide to using JSDoc for React.js | Better Documentation in React*. <https://www.inkoop.io/blog/a-guide-to-js-docs-for-react-js/>. [Online; Cited: 30.04.2021].
- [3] *Accessibility*. [https://developers.google.com/web/fundamentals/accessibility?utm\\_source=lighthouse&utm\\_medium=devtools](https://developers.google.com/web/fundamentals/accessibility?utm_source=lighthouse&utm_medium=devtools). [Online; Cited: 04.05.2021].
- [4] *Autocanon*. <https://github.com/mcollina/autocannon>. [Online; Cited: 04.05.2021].
- [5] *Automatically match columns in insert into select from*. <https://stackoverflow.com/questions/1787634/automatically-match-columns-in-insert-into-select-from>. [Online; Cited: 04.05.2021].
- [6] Benteler. *Business Unit Structures*. <https://www.benteler-automotive.com/en/products-competencies/structures/>. [Online; Downloaded: 13 january 2021].
- [7] Ryan Chenkie. *React Security Fundamentals*. <https://courses.reactsecurity.io/courses/react-security-fundamentals>. [Online; Cited: 02.02.2021].
- [8] Wikipedia contributors. *OWASP*. <https://en.wikipedia.org/w/index.php?title=OWASP&oldid=1017293558>. [Online; Cited: 05.05.2021].
- [9] P.J. Deitel. *Java : how to program*. [Book; Cited: 06.05.2021]. 2012.
- [10] Manjula Dube. *Get Started with Your First React.js App*. <https://we-are.bookmyshow.com/get-started-with-your-first-react-js-app-part1-34eee7b8559b>. [Online; Cited: 16 April 2021]. Apr. 2017.
- [11] *Fix issues before they exist*. <https://www.sonarlint.org/>. [Online; Cited: 03.05.2021].
- [12] OpenJS Foundation. *Node.js*. <https://nodejs.org/en/>. [Online; Cited: 02.02.2021].
- [13] The OWASP® Foundation. “V2.1 Password Security Requirements”. Version 4.0.2. In: (Sept. 2020). [Cited: 05.05.2021].
- [14] Guru99. *Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE*. <https://www.guru99.com/unit-testing-guide.html>. [Online; Cited: 01.05.2021].
- [15] *How to Write Doc Comments for the Javadoc Tool*. <https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>. [Online; Cited: 30.04.2021].
- [16] Facebook Inc. *Components and Props*. <https://reactjs.org/docs/components-and-props.html>. [Online; Cited: 02.02.2021].
- [17] Facebook Inc. *Introducing Hooks*. <https://reactjs.org/docs/hooks-intro.html>. [Online; Cited: 20.02.2021].

- 
- [18] Facebook Inc. *Using the Effect Hook*. <https://reactjs.org/docs/hooks-effect.html>. [Online; Cited: 20.02.2021].
- [19] *Java JWT: JSON Web Token for Java and Android*. <https://github.com/jwtkt/jjwt>. [Online; Cited: 22 April 2021].
- [20] Traversy media. *Multi Step Form With React & Material UI*. <https://www.youtube.com/watch?v=zT62eVxShsY>. [Online; Cited: 25.01.2021].
- [21] Mindmanager. *Bring clarity and structure to plans, projects, and processes*. <https://www.mindmanager.com/en/?link=wm>. [Online; Cited: 02.02.2021].
- [22] netcorecloud. *Send email in java using Gmail SMTP*. <https://netcorecloud.com/tutorials/send-email-in-java-using-gmail-smtp/>. [Online; Cited: 25 March 2021].
- [23] Inc npm. *Build amazing things*. <https://www.npmjs.com/>. [Online; Cited: 02.02.2021].
- [24] NTNU. *IMT2243 - Software Engineerings*. <https://www.ntnu.edu/studies/courses/IMT2243/2016#tab=omEmnet/>. [Online; Downloaded: 12 february 2021].
- [25] NTNU. *IMT3281 - Application Development*. <https://www.ntnu.edu/studies/courses/IMT3281#tab=omEmnet/>. [Online; Downloaded: 12 february 2021].
- [26] Max Pekarsky. *Does your web app need a front-end framework?* <https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/>. [Online; Cited: 02.05.2021]. Feb. 2020.
- [27] Performancelab. *What is User Acceptance Testing (UAT) for Websites? Definition & Examples*. <https://performancelabus.com/what-is-uat-for-websites/>. [Online; Cited: 27.04.2021].
- [28] Performancelab. *Why Is Load Testing Important for Web Applications*. <https://performancelabus.com/load-testing-for-web-applications/>. [Online; Cited: 27.04.2021].
- [29] Inc. Postman. *Send Requests and View Responses*. [postman.com/product/api-client/](https://postman.com/product/api-client/). [Online; Cited: 05.05.2021].
- [30] *Prettier*. <https://prettier.io/>. [Online; Cited: 03.05.2021].
- [31] ReactJS. *Code Splitting*. <https://reactjs.org/docs/code-splitting.html>. [Online; Cited: 03 April 2021].
- [32] *Rules*. <https://eslint.org/docs/rules/>. [Online; Cited: 03.05.2021].
- [33] *Tabnine Autocomplete AI*. <https://marketplace.visualstudio.com/items?itemName=TabNine.tabnine-vscode>. [Online; Cited: 20.01.2021].
- [34] Material UI team. *MATERIAL-UI*. <https://material-ui.com>. [Online; Cited: 02.02.2021].
- [35] *Tools for developers*. <https://developers.google.com/web/tools/lighthouse>. [Online; Cited: 04.05.2021].
- [36] Rebecca Vogels. *A 6-Step Guide to Web Application Testing*. <https://usersnap.com/blog/web-application-testing/>. [Online; Cited: 27.04.2021].
- [37] Wikipedia. *Benteler International*. [https://en.wikipedia.org/wiki/Benteler\\_International](https://en.wikipedia.org/wiki/Benteler_International). [Online; Downloaded: 13 january 2021]. Dec. 2020.
- [38] Wikipedia. *Prepared Statement*. [https://en.wikipedia.org/wiki/Prepared\\_statement](https://en.wikipedia.org/wiki/Prepared_statement). [Online; Cited: 12.05.2021].
- [39] Wikipedia. *Salt (cryptography)*. [https://en.wikipedia.org/wiki/Salt\\_\(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography)). [Online; Cited: 04.05.2021].

## **Appendix A**

# **Project plan**



# Bachelor Thesis

## Project plan

Kristian Tveiten Jesper Trøan Emma Sofie Søvik Nils Olav Tuv

January 2021

# Contents

<b>1</b>	<b>Goals and frames</b>	<b>3</b>
1.1	Background . . . . .	3
1.2	Project Goals . . . . .	3
1.2.1	Outcome goals . . . . .	3
1.2.2	Performance goals . . . . .	3
1.2.3	Learning objectives . . . . .	4
1.3	Project frames . . . . .	4
<b>2</b>	<b>Scope</b>	<b>4</b>
2.1	Problem area . . . . .	4
2.2	Limitations . . . . .	5
2.3	Task description . . . . .	5
<b>3</b>	<b>Project organization</b>	<b>5</b>
3.1	Responsibilities and roles . . . . .	5
3.2	Routines and group rules . . . . .	6
3.2.1	Routines . . . . .	6
3.2.2	Group Rules . . . . .	6
<b>4</b>	<b>Planning, follow up and reporting</b>	<b>7</b>
4.1	Choice of system development models . . . . .	7
4.1.1	Scrum + Kanban . . . . .	7
4.1.2	Time estimation models . . . . .	8
4.2	Main division of the project . . . . .	8
4.3	Plan for status meetings and decisions in the period . . . . .	9
<b>5</b>	<b>Quality assurance</b>	<b>9</b>
5.1	Standard tools . . . . .	9
5.1.1	General tools . . . . .	9
5.1.2	Issue tracking tools . . . . .	9
5.1.3	Documentation tools . . . . .	10
5.1.4	Development and testing tools . . . . .	10
5.1.5	Time tracking Tools . . . . .	10
5.2	Documentation . . . . .	10
5.3	Risk analysis . . . . .	11
<b>6</b>	<b>Plan for execution</b>	<b>12</b>
6.1	Activities . . . . .	12
6.2	Gantt schema . . . . .	13

# 1 Goals and frames

## 1.1 Background

Benteler is a big worldwide company that has its headquarters in Salzburg, Austria. The company employs around 30 000 employees from around 100 locations around the globe [3], and Raufoss in Norway is one of them. Extrusion is done to make profiles with a specific shape, and Benteler are currently doing extrusion in Germany, USA and Norway. [1] The cost for producing the profiles are given by the size and complexity of the profile as well as the type of aluminium alloy. Therefor Benteler came up with a task suggestions that was approved by NTNU and we thought was interesting. In the task description they describe that they want us to create an application with a database connection for adding new profiles and retrieving data from previous assessed profiles.

## 1.2 Project Goals

Goals for the project is divided into three categories, outcome goals, performance goals and learning objectives.

### 1.2.1 Outcome goals

Outcome goals are the goals describing the wanted result from the project.

- When the project is over Benteler should have a well-functioning application with the needed functionality.
- The application should connect to the Database in an safe and secure way.
- The application should make it easy to add, remove or edit needed properties in the database.
- The application should be easy to use. It should have a clean and easy design, with recognizable parts (like header, sidemenu, footer etc.)

### 1.2.2 Performance goals

Performance goals is the goals describing what Benteler wish to accomplish with the application.

- Make it easier for employees to use and understand the system.
- Extend the use of the system, and give more employees access to a restricted part of the application.
- Save time using the new application, as their old program is slow and ineffective.

### 1.2.3 Learning objectives

Learning objectives is the knowledge and experience we hope to gain during the project.

- Gain experience on the system development process. A project like this will be relevant experience for future development project and future jobs.
- Gain more knowledge and experience on web-applications and web development. The group members does not have previous experience on web development.
- More experience with database and database connections.
- Gain experience on backend development.
- Gain experience on frontend development.

## 1.3 Project frames

This bachelor thesis has a time frame. The project starts 11.01.21 and has to be delivered by 20.05.21. The software should then be handed over to Benteler together with the bachelor thesis report. Apart from that there is no frames we need to take into consideration, and we are free to choose the development tools and development languages we see fit.

## 2 Scope

### 2.1 Problem area

As Benteler already have an existing application for this problem they wish to develop a new solution that simplifies and makes things easier for them. The old system lacks key functionality in places where the new system will fill in. The old system is slow and inefficient, and Benteler wants a new program to take over its place.

**The program is divided into four main parts:**

- **Request** A request to produce a new profile. Creating a new request can happen in two ways, from scratch or based on an earlier request. It is also possible to update or view existing requests.
- **TFC** TFC meetings occurs for each new request, and in those meetings the attendees will create a TFC form. One request can result in multiple TFCs, depending on the machines being used. There are two ways of creating new TFC form, based on a request or based on an existing TFC with the possibility of collecting new request data and update the needed values. Viewing and updating existing TFCs is also possible.

- **CostCalc** CostCalc is a report with specific parameters from the request and TFC, these values and input of more data are used by the CostCalc-team to calculate the cost of the new profile.
- **Profile order** The order is put together with information from the request, TFC and calculations from the CostCalc-teams. Also possible to Re-order and order single materials without reference to a specific request.

## 2.2 Limitations

Benteler has a desire that they want us to improve their old firmware in a way that makes the system faster and more intuitive to use. To get this system out to as many Benteler employees as possible we will write the code documentation, user guide and the program itself in English.

## 2.3 Task description

In order to plan the operating parameters and the cost for each profile Benteler wants us to create a application with a database connection for this purpose. The application is going to be used by Benteler both nationally and globally. Users should be able to enter data for new profiles and retrieve data for previously assessed profiles in a clear and simple way. The user interface should be simplistic and in English. The application should also link the model/drawing to the corresponding forms and accept both International System of Units (SI) and United States customary units (US). As Benteler already have an old solution for some of this, our main tasks is to:

- Reduce amount of manual parameter handling
- Make a new easier and faster solution that suits their needs
- Provide more flexibility in searchable parameters
- Make the solution work closer with the CostCalc team

# 3 Project organization

## 3.1 Responsibilities and roles

The group chose Jesper Trøan as the Project leader(Scrum master). The project leader is responsible for keeping the MindManager(5.1.1) node map up-to-date, planing agendas for each meeting, and must monitor the group progress so the schedule is followed.

The group also decided to have a document manager, which has the responsibility of taking backups of documents, writing minutes of meetings, document important decisions and monitor the time tracking for the project. Kristian Tveiten got the role of document manager.

Every member has the responsibility of their own to keep track of their working hours and log them.

## **3.2 Routines and group rules**

### **3.2.1 Routines**

#### **Weekly work**

Every group member is expected to work approximately 30 hours every week with the project. The project backlog management Trello (5.1.2) is used to always have a grasp on available tasks. The Trello board should be updated during every meeting. The members will have an overview on how much they work each week in the time tracking program Toggl (5.1.5), this will make it easier for each member to stay on track with the working hours.

#### **Meeting times**

The group has scheduled weekly meetings with both the supervisor and the client (Benteler). Supervisor meetings is every Monday 11:00-12:00, and with Benteler every Tuesdays 11:30-12:30. Every meeting will result in a written meeting report. These meetings is obligatory and every member has to attend. If it is not seen necessary by the members and Benteler or the supervisor to have a meeting one week, the meeting can be canceled, this should be decided in advance.

The group has also planned group work on campus every Monday from 08:00-12:00, Tuesdays 08:00-12:00 and Wednesdays 08:00-12:00. This is not obligatory and it is set up as a way to communicate with the other team members while working individually. It is possible to add more days or hours to the group work sessions if the team members wish to do so.

### **3.2.2 Group Rules**

To ensure the groups well being and effectiveness is kept, we decided to implement group rules applying to every member.

#### **Absence**

Every member should and is expected to attend all meetings with either our supervisor, Benteler and sprint meetings. If this is not possible the group member should let the other group members know ahead of time with a reason (minimum two hours in advance, unless an emergency).

#### **Effort**

Every group member is expected to work 30 hours during the week with the bachelor thesis project. This includes meetings and collective work.

#### **Conflicts**

In the case of an internal conflict where two or more group members disagree on something, the whole group should discuss this and come to a solution by discussion or voting, where the group leader has two votes if there is a draw.

#### **General rules**

After each meeting, every group member should have a clear picture of what they are supposed to work with towards the next meeting. If this is not clear, the member should let the group know and together find a solution. If it is not clear due to unorganized meetings, the meeting format should be improved.

If a member is stuck on a task, the member should try to get help from the other members in the group. If no one can figure out the problem, the group should contact for example our supervisor, Benteler or people with relevant knowledge to solve the issue.

#### **Consequences of breaking the rules**

In the case of a member breaking the rules, like being absent a period of time or not working enough hours, the person will be contacted by the other members and given a written warning. If this does not solve the problem the group will contact our supervisor or the bachelor thesis managers for discussion on further corrective action.

## **4 Planning, follow up and reporting**

### **4.1 Choice of system development models**

We have discussed both agile and plan-driven development methods. We have come to the conclusion that plan-driven development methods don't match the challenges in this project. This is mainly because of the projects flexibility.[2] The stakeholder has stated that they want a core functionality, but that changes can appear and the final product can be a bit different from what they initially thought.

Extreme programming (XP) was also a option that could fit the project well, as it's a fast-paced working environment that values teamwork and good communications. A con with XP would be the lack of written documentation. A big part of this project is to write the report and document the process, therefore a development method with the possibility to add good documentation routines is needed. The group also lack experience in the XP practices like Test-first development and Pair programming. And with these reasons we decided to look for a different approach.

We also evaluated Kanban as it is a great visual development model, which helps to keep track of the backlog and the current backlog elements that is either worked on, finished or put to testing/quality assurance phase. With Kanban we can also limit the amount of concurrent backlog tasks on a given stage, so backlog elements does not accumulate.

#### **4.1.1 Scrum + Kanban**

Combining Kanban with scrum is an excellent choice for this project, as the Kanban board can be revised/balanced during sprint meetings/scrums meetings, to catch up to tasks that have stayed in a certain phase for too long. This combats congestion of tasks in a certain phase. The average team velocity will also help us estimate time even better for upcoming backlog tasks.

As we are not experienced at all in web-languages, using scrum will also allow us to come with ideas of our own or change programming language for a part of the

system if we see fitting. Product owner has not set limitations on this, so we are free to choose the technologies we believe is the best fit.

Regular scrum meetings with the product owner every two weeks will allow us to present finished software modules to Benteler, and get response and input on what needs to be changed or improved, which would then be added to the backlog and used in a coming sprint. Scrum is also fitting for a smaller development team and it will also allow us to prioritize the most important parts early so we can deliver a functional program early.

The group will have scrum meeting at least 3 days of the week, to review progress and possibly re-prioritize, change or break down certain backlog elements. The sprint duration will be two weeks, starting on Wednesdays and ending on Tuesdays when we have meetings with the product owner. On Fridays the group will have presentations internally for newly developed software modules if the group see it fitting, and the software is complex enough to require a walk through for the other group members to gain understanding of the code.

Scrum is also a fitting method to add the documentation routines we need to complete the project with the report. We just simply add the documentation task in the backlog, so that they can easily be added as a task to complete during a Sprint.

We decided that the role of scrum master should be delegated to the project leader Jesper Trøan, and Frode from Benteler is the product owner.

#### **4.1.2 Time estimation models**

Planning poker will be utilized in combination with the group velocity for average tasks completed in a week. As we are not very experienced in time estimation, we will both use the group velocity and our supervisor to get a pinpoint on which time-range we should expect, and then use planning poker to come to a agreement.

## **4.2 Main division of the project**

The project starts with us getting familiar with what is required for the task. After that we can start with developing the diagrams such as an use case diagram and also get the requirements in place. We also have to set up a fixed time to focus on researching languages and tools for the project considering that we do not have enough experience with web development to start right away. Once we have control over that, the project will mainly consist of four main parts:

- Develop the frontend of the web application
- Develop the database
- Develop the backend of the web application
- Write the report



### **4.3 Plan for status meetings and decisions in the period**

We have planned weekly meetings with our supervisor Sony George where the goal is to share the progress and discuss coming issues. This can change depending on the necessity. Since we have two week long sprints we will have the opportunity to get feedback before, during and after the sprints.

## **5 Quality assurance**

### **5.1 Standard tools**

For the project we have a large toolbox of tools we have learnt that we will utilize to keep us as organized and effective as possible. We also utilize some new tools which simplify the process of documenting and reporting.

We utilize the following tools:

#### **5.1.1 General tools**

##### **Notion**

Notion is a great productivity application, with a spectrum of different functions that are great for planning and being productive. We have chosen to use this application as a shared calendar, where we will have an overview on all the meetings and deadlines relevant to the project.

##### **MindManager**

MindManager is used for planning the meetings for the project. All meeting will be added to a mindmap with nodes like agenda, Meeting Minutes and parkinglot. The application makes it possible to have a systematic overview over all the meetings, and makes it easy to find information about specific meetings when needed. The Meeting Minutes are written in Google Docs, but are added into the mindmap when completed.

#### **5.1.2 Issue tracking tools**

##### **Trello**

This tool is used as a kanban board <sup>1</sup> for the scrum backlog, where we have the following columns: To Do, Doing, Quality Assurance, and done. Quality assurance is used for when a group member is finished with a task, either coding or writing a section in the report, and the artifact is subject to quality checking by the other members. We can then read and edit, and mark the issue with their name once they are finished, and once that is done, the issue is moved from Quality Assurance to done.

---

<sup>1</sup>A visualisation tool to help optimize work flow

### 5.1.3 Documentation tools

#### Overleaf

Overleaf <sup>2</sup> is used for writing the report and project plan with precise structure and layout.

#### Google Docs

Google Docs is used for writing meeting reports as well as agendas for upcoming meetings. We found that having a docs is a easy way for everyone to take notes together and have control over agendas.

### 5.1.4 Development and testing tools

For concurrency control and repository we utilize NTNU's GitLab solution, which ensures us safety in regards of uptime. When it comes to development environment we are free to use the technologies and tools we see fit. In our opinion a web application will be the most fitting solution for Benteler and for solving this specific task. Considering the lack of experience the group has in web development, proper research is needed before the languages and tools are decided. After the fixed research period the group will have an overview on all the technologies that are needed for the solution, and can then decide on extra tools like for example what linter tool to use for static code analysing.

### 5.1.5 Time tracking Tools

#### Toggl

For keeping track of time we use a service called Toggl, it enabled us to selectively log time spent working on a single task. It is organized so all of the group members log spent time on a shared project, and names the activity they worked on like "Quality assurance", "Creating use case diagram", "Internal meeting", etc. The activities is also tagged with "Meetings", "Project Plan", "Report" and "Coding" to enable structured time log extraction. This tool gives the group members an overview of the time spent on the project, and it makes it possible to check each members weekly effort on the project.

## 5.2 Documentation

To properly document time usage for the different activities concerning the project, we have set up a Excel document where the document manager will collect data from Toggl and fill in time usage each Friday after the week ends, to keep track of time usage and possibly catch anomalies in the reported time, to ensure as correct time logging as possible.

All our meetings with either the supervisor, Benteler or internally is documented in the form of meeting minutes, where whatever was discussed and agreed on is writ-

---

<sup>2</sup>Cloud based Latex editor

ten. This way we have a clear picture of all the information that has been given and decisions that has been made previously.

Trello will also aid in documenting tasks that has been done, who has reviewed tasks in the "Quality Assurance" phase, which development activity a task belongs to and so on.

Our calendar in Notion will document our plan for the weeks, being meetings, campus work and similar.

### 5.3 Risk analysis

Below is a table that shows the risks of the project and their probability and the possible consequences they have. Probability is divided into 4 cases: low, moderate, high and very high. The consequences the risks are divided into: insignificant, tolerable, serious and catastrophic.

Number	Risk	Probability	Consequence
1	Project is not completed before deadline	Moderate	Serious
2	One team members get sick and is absent an extended amount of time.	Low	Tolerable
3	Loss of data, code or report.	Moderate	Catastrophic
4	Supervisor is not available as needed.	Moderate	Tolerable
5	Benteler (Product owner) is not available as needed.	Moderate	Serious
6	Lack of documentation	High	Serious

#### Measures

This is the measures that are planned to decrease the probability of the risk happening, as well as lower the consequence of the risk if it does happen.

Measure	Number
Have a plan for the entire project that is realistic and possible to follow, and easy to check if the project is on time or not. If a delay should happen, inform the product owner and discuss if removing some planned functionality is an option.	1
To reduce the consequences of a member being sick the members will constantly review the others work. This is done by adding a finished task to the Quality Assurance list (Kanban board), and then every member has to review the task before it is moved further. The members will then always be up to date on what has been done and how to do it.	2
The code will be in a repository in Gitlab, and also locally on all the members' computers. The report will be written in overleaf, the group leader has the responsibility to regularly download and keep a backup of the report.	3
Weekly meetings with the supervisors are planned. If the supervisors do not attend these meetings, contact them and ask for new meetings. If the supervisors are not available consider contacting other NTNU employees to ask for the help needed.	4
Weekly meetings with Benteler are planned. If Benteler does not attend these meetings and are not able to reschedule, the members will make assumptions about the product and Bentelers preferences.	5
Documentation tasks will be added to the backlog and will be seen as high priority as it is important to get proper documentation on the project.	6

## 6 Plan for execution

### 6.1 Activities

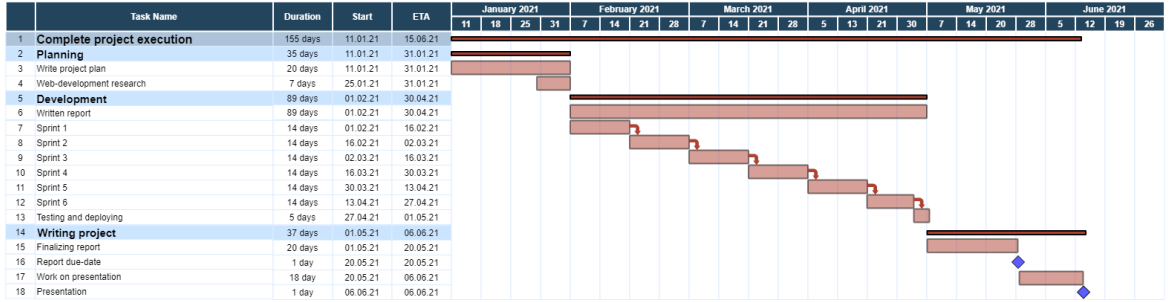
The Gantt schema is divided in to three main parts: Planning, Development and Writing report.

Planning is dedicated to writing the project plan and researching web-development. The Development part is divided in to two week long sprints, starting on Wednesday and ending on Tuesdays as decided in 4.1.1. The content of the sprints is decided in the coming sprint meetings.

We hope to get more knowledge on what to include in each Sprint and when different part of the system should be done after our research period. Although we use an agile development method, having an overview of when we expect to be done with certain parts of the system can help us plan each Sprint accordingly and increase our chances of completing the project in time.

The final part, Writing project, is divided in to two main parts. The first priority is to finalize the written report before the due date May 20th. The second is to work on the presentation taking place two weeks later.

## 6.2 Gantt schema



## References

- [1] Benteler. *Business Unit Structures*. <https://www.benteler-automotive.com/en/products-competencies/structures/>. [Online; Downloaded: 13 january 2021].
- [2] Ian Sommerville. *Software engineering*. Pearson, 2016.
- [3] Wikipedia. *Benteler International*. [https://en.wikipedia.org/wiki/Benteler\\_International](https://en.wikipedia.org/wiki/Benteler_International). [Online; Downloaded: 13 january 2021]. Dec. 2020.

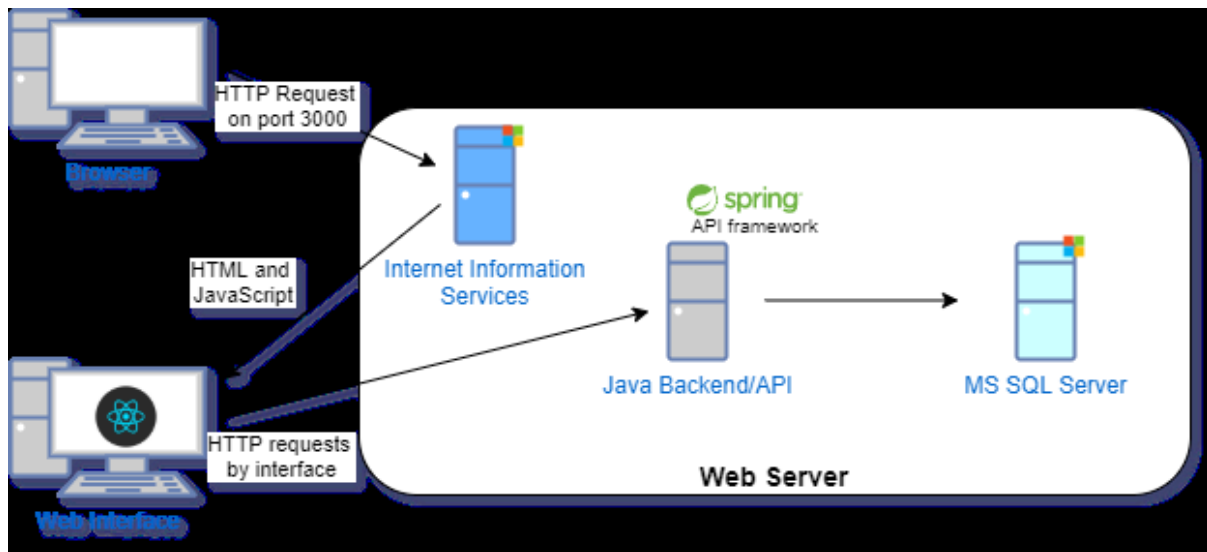
## **Appendix B**

**Documentation and procedures  
regarding the code, web-server and code  
assuring longevity and re-usability of the  
application**

<b>Code structure</b>	<b>2</b>
Frontend	2
Backend	3
<b>How to add, change or delete parameters to the database and code.</b>	<b>5</b>
<b>Database</b>	<b>5</b>
Backend (All forms)	8
<b>Request</b>	<b>11</b>
<b>Database</b>	<b>11</b>
<b>Backend</b>	<b>11</b>
<b>Frontend</b>	<b>11</b>
<b>TFC</b>	<b>14</b>
<b>Database</b>	<b>14</b>
<b>Backend</b>	<b>14</b>
<b>Frontend</b>	<b>14</b>
<b>Order</b>	<b>16</b>
<b>Database</b>	<b>16</b>
<b>Frontend</b>	<b>16</b>
<b>Backend</b>	<b>17</b>
How to add, change or delete a simple textfield in the forms.	17
<b>How to re-deploy the application/update the application.</b>	<b>19</b>
Required software that must be installed:	19
Procedure (Backend Java API)	20
Procedure (Frontend React web interface)	21
<b>How to add or delete an Alloy in the ranked list on dropdowns.</b>	<b>23</b>
<b>Suggested reading/viewing:</b>	<b>27</b>
<b>The server setup (if the application is to be moved to another server)</b>	<b>28</b>

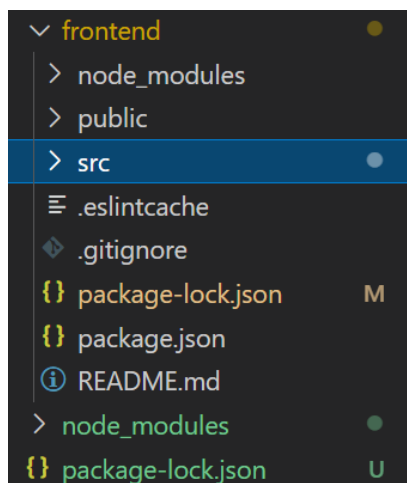


## Code structure



This is a rough description of the entire application. First we have our frontend web application coded in React. This is what the user sees and where the interaction with the application happens, it is considered a thin client as all information is fetched server side and nothing besides cookies is stored client side. The interface is delivered with Microsoft Internet Information Services on the web server.

## Frontend

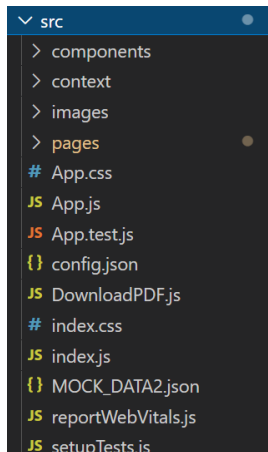


**node\_modules** is where all packages are downloaded to.

The **public** folder is used to contain things used public, in our application it is not much used.

**src** is where the entire frontend of the application lays.

The **.json** files is where the dependencies lay. Here you can controll whitch version of the packages is installed.



**components:** This folder is for components much used in the application. Here we have the footer, sidemenu, topmenu and a popup file containing most dialogs in the application. This includes dialogs to add and update press, alloys and etc.

**context:** This folder is containing files used to make calls to the backend. Every call made to the backend goes through files in this folder. The files are named after what place in the backend they make the calls to. For example UserService.js is containing functions to add, update and delete users.

**images:** Contains benteler logos.

**pages:** Pages includes folders for order, request and tfc, as well as the files for every other page in the application. Each page has its own .css file which includes all css used for each file.

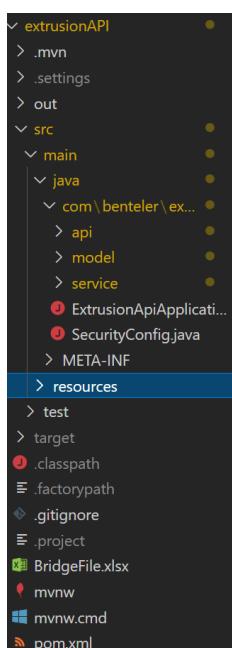
**App.js:** This is the application's "main" file. It does all the routing in the application. Here the users privileges are checked and returns the sites the user is allowed to see.

**index.js:** Here the application is rendered. The root of the application is rendered:

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

## Backend

The backend is structured like this:



The **src** folder contains all the code for the backend, as well as other resources used.

The backend code is divided into three parts: api, model and service.

**api:** The api is the controller folder and is responsible for controlling the application. The controllers returns the response of the requests.

**model:** The model components represent the data being transferred. For example the user controller will retrieve user models from the database.

**service:** Service folder resources the methods to perform operations with one or more models.

To explain this a bit better, I will include a full example to add a user:

Let's start with the user-model:

**pom.xml:** Includes all dependencies for the backend.

```

public class User {
    private final UUID id;
    private final String name;
    private final String password;
    private final String role;

    public User(@JsonProperty("id") UUID id,
               @JsonProperty("name") String name, @JsonProperty("password") String password, @JsonProperty("role") String role){
        this.id = id;
        this.name = name;
        this.password = password;
        this.role = role;
    }
}

```

This is the main part of the user model. The model also has getters.

So over to the service-layer. Here is the function to create a new user.

```

public static boolean createNewUser(UUID id, String name, String password, String role, Connection con){
    try{
        PreparedStatement st = (PreparedStatement) con.prepareStatement("INSERT INTO Users (UUID, name, password, role) VALUES (?, ?, ?, ?);");
        st.setString(1, id.toString()); //Inserting id in the prepared statement
        st.setString(2, name); //Inserting name in the prepared statement
        st.setString(3, password); //Inserting password in the prepared statement
        st.setString(4, role); //Inserting role in the prepared statement

        st.execute();
        return true;
    }catch(SQLException ee){
        ee.printStackTrace();
        return false;
    }
}

```

The function in the service-layer includes the prepared statement to insert the user to the database. The use of prepared statements is important to secure the application from SQL-injections.

Create user in the controller:

```

@PostMapping
public boolean addUser(@RequestBody User user) {
    return userService.createNewUser(UUID.randomUUID(), user.getName(), userService.hashPass(user.getPassword()), user.getRole(), con);
}

```

Model

Service

The controller requests a body containing the attributes from the user-model. It then runs the *createNewUser*-function in the service-folder with the attributes from the User-model sent from the frontend.

To see how this function is called in the frontend we have to take a look in the context-folder. In the file *UserService.js* we find every function connected to the user-backend.

```

addUser(data) {
    return axios.post(API_URL + "user", data);
}

```

This is the function that calls for the *addUser*-function in the *UserController*. The parameter sent, *data*, is containing the user-data as the user-model. We use *axios*, a promise based http client, to connect the frontend with the backend. The *API\_URL* is the ip of the backend. In this case the function returns a boolean, either true if the request is successful or false if it fails.

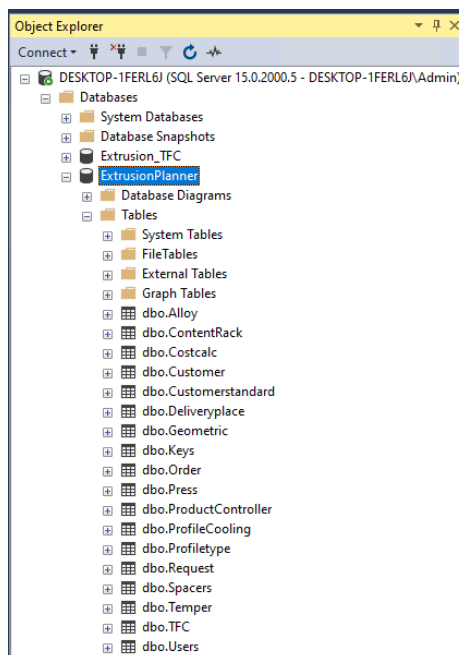
# How to add, change or delete parameters to the database and code.

## Database

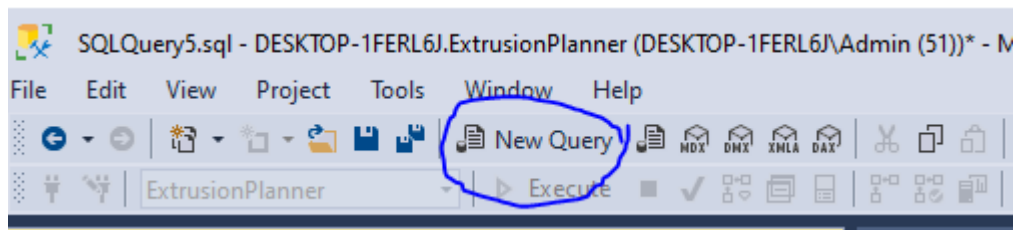
Generally when you are adding, changing or deleting parameters in the backend and frontend code, you have to do the same changes in the corresponding database too. Now we will go through step by step how to make these changes in the database. On all these operations you first need to open SQL Server management studio (SSMS) on the server.

### How to add a parameter in a table in SSMS:

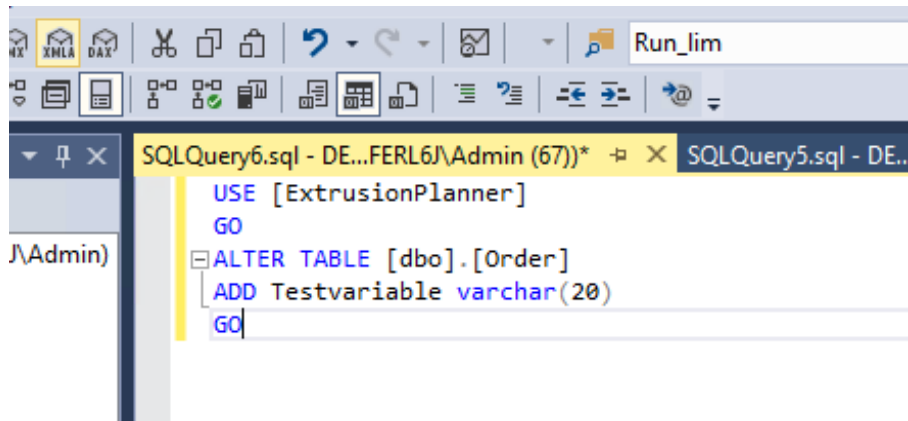
1. Head over to the left corner in the object explorer to get an overview over what the different tables are named.



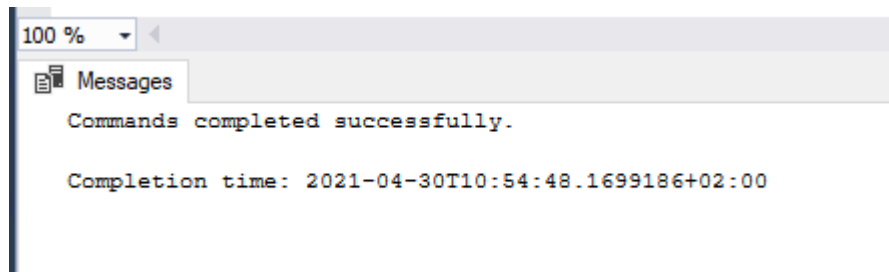
2. Click on new query in the topnavbar of the SSMS:



3. A new query editor will open. Here you have to write in this SQL query:  
USE [ExtrusionPlanner] ( ←----- The name of the database)  
GO  
ALTER TABLE [dbo].[Order] ( ←----- Name of the table)  
ADD Testvariable varchar(20) ( ←----The name of the new variable, and then the datatype)  
GO



After you have put in the right names of the database, tables, name and datatype of the new parameter you just have to right click in the editor and then click on Execute or just F5. If you get a message under the query editor that the command completed successfully, you have done it right.

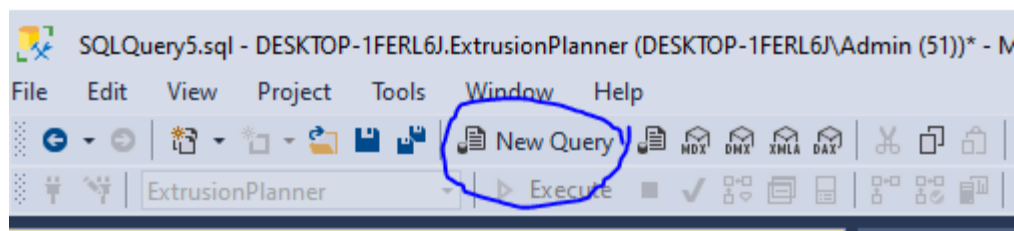


There are many different datatypes to choose from, but the most common ones is int (if the type is a number) and varchar(size) (if the type consists of letters). You can get an overview of the different datatypes here:

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15>.

## How to modify a parameter in a table in SSMS:

1. Open a new query by clicking the new query button in the topnavbar in SSMS.



2. Write this SQL query in the query editor:
 

```
USE [ExtrusionPlanner] ( ←---- Name of the database)
GO
ALTER TABLE [dbo].[Order] ( ←---- which table to modify)
ALTER COLUMN Part_category varchar(20) ( ←-- Which parameter to modify, and
what new datatype to change to. )
GO
```

```

SQLQuery1.sql - DE...FERL6J\Admin (64))* ~vsA6EF.s
USE [ExtrusionPlanner]
GO
ALTER TABLE [dbo].[Order]
ALTER COLUMN Part_category varchar(20)
GO

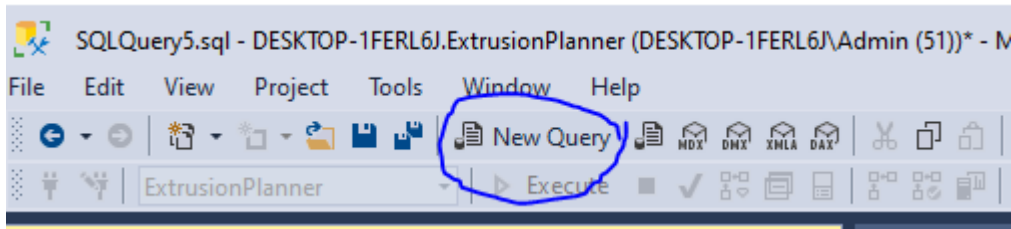
```

Here we will change the column Part\_category in the order table to maximum take 20 characters instead of 10.

Right click inside the editor and then click Execute, or click F5. If you have done it right you get the message under the query editor that the command completed successfully.

### How to delete a parameter in a table in SSMS:

1. Open a new query by clicking the new query button in the topnavbar in SSMS.



2. Write this SQL query in the query editor:  
 USE [ExtrusionPlanner] ( ←----- Name of the database)  
 GO  
 ALTER TABLE [dbo].[Order] ( ←---- which table to modify)  
 DROP COLUMN Part\_category ( ←-- Which parameter to delete)  
 GO

```

SQLQuery1.sql - DE...FERL6J\Admin (64))* ~vsA
USE [ExtrusionPlanner]
GO
ALTER TABLE [dbo].[Order]
DROP COLUMN Part_category
GO

```

Here we will delete the column Part\_category in the database. Right click inside the editor and then click Execute, or click F5. If you have done it right you get the message under the query editor that the command completed successfully.

## Backend (All forms)

### Adding a new parameter:

When we are adding new parameters to the tables we have to add them in the backend code too. The first place we have to add this new parameter is in the model of the corresponding forms. An important note is when you are adding new parameters to the TFC or Request form you also have to add this parameter to the TFCandRequest model.

path:

src → main → java → com/benteler/extrusionAPI → api → model → The specific form

Here is an example of adding a parameter to the orderform. The procedure is very similar in the TFC and requestForm, except from that you then have to add this parameter to the TFCandRequest model too.

```
String order_comment;
String crush_test;
String sketch_no;
int profile_length_mm;

public OrderForm(@JsonProperty("Request_no") int request_no,@JsonProperty("Request_ver_no") int request_ver_no, @JsonProperty("TFC_no")
@JsonProperty("Customer") String customer,@JsonProperty("Project_name") String project_name,
@JsonProperty("Order_date") String order_date, @JsonProperty("Order_name") String order_name,@JsonProperty("Email_ord
@JsonProperty("Product_controller") String product_controller,
@JsonProperty("Project_SAP") String project_SAP,@JsonProperty("Project_no")String project_no,
@JsonProperty("SOP") String sop,@JsonProperty("Place_of_delivery") String place_of_delivery,
@JsonProperty("Average_volume_y") int average_volume_y,@JsonProperty("Peak_volume_y") int peak_volume_y,
@JsonProperty("Volume_total") int volume_total,@JsonProperty("Part_description") String part_description,
@JsonProperty("Part_category") String part_category,@JsonProperty("Part_length") float part_length
```

1. Here you can add a new parameter under the last parameter with the datatype first and then the name of the parameter. You also have to add this new parameter in the same order as where you declared the parameter in the constructor by typing `@JsonProperty("New_parameter") int new_parameter`. Here is an example of doing this:

```
int profile_length_mm;
int new_parameter;

public OrderForm(@JsonProperty("Request_no") int request_no,@JsonProperty("Request_ver_no") int request_ver_no, @JsonProperty("TFC_no") int tfc_no,@JsonProperty("TFC_ver_no")
@JsonProperty("Customer") String customer,@JsonProperty("Project_name") String project_name,
@JsonProperty("Order_date") String order_date, @JsonProperty("Order_name") String order_name,@JsonProperty("Email_orderer") String email_orderer,
@JsonProperty("Product_controller") String product_controller,
@JsonProperty("Project_SAP") String project_SAP,@JsonProperty("Project_no")String project_no,
@JsonProperty("SOP") String sop,@JsonProperty("Place_of_delivery") String place_of_delivery,
@JsonProperty("Average_volume_y") int average_volume_y,@JsonProperty("Peak_volume_y") int peak_volume_y,
@JsonProperty("Volume_total") int volume_total,@JsonProperty("Part_description") String part_description,
@JsonProperty("Part_category") String part_category,@JsonProperty("Part_length") float part_length,
@JsonProperty("Parts_pr_vehicle") int parts_pr_vehicle,@JsonProperty("SAP_CAD_model") String sap_CAD_model,
@JsonProperty("Profile_no") String profile_no, @JsonProperty("Profile_rev") String profile_rev,
@JsonProperty("TFC_sketch") String tfc_sketch,
@JsonProperty("Alloy") String alloy,@JsonProperty("Temper") String temper,
@JsonProperty("Surface_quality") String surface_quality,@JsonProperty("Rex") String rex,
@JsonProperty("TGC") String igc,@JsonProperty("Ductility") String ductility,
@JsonProperty("Customer_standard") String customer_standard,@JsonProperty("Geometric_tolerances") String geometric_tolerances,
@JsonProperty("Other_tolerances") String other_tolerances,@JsonProperty("Tensile_test") String tensile_test,
@JsonProperty("Structure_test") String structure_test,@JsonProperty("Split_test") String split_test,
@JsonProperty("Certificate") String certificate,@JsonProperty("BTM") String btm,@JsonProperty("Profile_cutting_length") String profile_cutting_length,
@JsonProperty("Place_of_delivery_2") String place_of_delivery_2,@JsonProperty("Week_of_delivery") String week_of_delivery,
@JsonProperty("Profile_quantity") String profile_quantity,@JsonProperty("WBS_number_profile") String wbs_number_profile,
@JsonProperty("WBS_number_die") String wbs_number_die,@JsonProperty("Profit_center") String profit_center,
@JsonProperty("Order_comment") String order_comment,@JsonProperty("Crush_test") String crush_test,
@JsonProperty("Sketch_no") String sketch_no, @JsonProperty("Profile_length_mm") int profile_length_mm, @JsonProperty("New_parameter") int new_parameter) {
this.request_no = request_no;
```

2. You also have to add this sentence: "this.new\_parameter = new\_parameter;" under the declaration of the Jsonproperties, and add a getNew\_parameter function with the parameter type before the functionname like this:

```

        this.profile_length_mm = profile_length;
        this.new_parameter = new_parameter;
    }

    public int getNew_parameter() {
        return new_parameter;
    }

```

3. Now you are done updating the model.
4. The next step is to update the functions in the corresponding service file.  
 path: src → main → java → com/benteler/extrusionAPI → api → service → The specific form. The new parameter has to be added in all the functions that are using all the parameters. So for the orderform you have to add this new parameter in the same order as it is in the model in Saveorder, getOrderById, getmostRecentOrders and updateOrderById. As the saveform files uses preparedstatement we have to add the new parameter under the last parameter in the sql query, and also add an extra ? like this:

```

        " Sketch_no," +
        " Profile_length_mm" +
        " New_parameter)" +
        " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,"
    );
    st.setInt(1, of.getRequest_no());
    st.setInt(2, of.getRequest_ver_no());

```

We also have to set the value of this new ? by adding one to the number to 52 and using the getfunction we made in the model. You have to use st.setInt if the parameter is an int, and st.setString if the parameter is an String like this:

```

        st.setInt(51, of.getProfile_length_mm());
        st.setInt(52, of.getNew_parameter());

        st.execute();
        return true;
    }

```

In the getOrderById and getMostRecentOrders functions we just have to add this:

```

        rs.getString("Sketch_no"),
        rs.getInt("Profile_length_mm"),
        rs.getInt("New_parameter")
    );

```

Also here we have to use rs.getInt if the parameter is an int and rs.getString if the parameter is an String.

For the updateOrderById function we have to add the new parameter in the sql query like this:

```

PreparedStatement st = (
    PreparedStatement) con.prepareStatement(
        "UPDATE [Order] SET" +
        " Request_no = ?, Request_ver_no=? , TFC_no = ?, TFC_ver_no = ?, Customer = ?, Project_name = ?, Order_date = ?, O"
        " Product_controller = ?, Project_SAP = ?, Project_no = ?, SOP = ?," +
        " Place_of_delivery = ?, Average_volume_y = ?, Peak_volume_y = ?, Volume_total = ?, Part_description = ?," +
        " Part_category = ?, Part_length = ?, Parts_pr_vehicle = ?, SAP_CAD_model = ?, Profile_no = ?, Profile_rev= ?," +
        " TFC_sketch = ?, Alloy = ?, Temper = ?, Surface_quality = ?, Rex = ?, IGC = ?," +
        " Ductility = ?, Customer_standard = ?, Geometric_tolerances = ?, Other_tolerances = ?, Tensile_test = ?, Structu"
        " Split_test = ?, Certificate = ?, BTM = ?, Profile_cutting_length = ?, Place_of_delivery_2 = ?," +
        " Week_of_delivery = ?, Profile_quantity = ?, WBS_number_profile = ?, WBS_number_dia = ?, Profit_center = ?," +
        " Order_comment = ?, Crush_test = ?, Sketch_no = ?, Profile_length_mm = ?, New_parameter = ?" +
        " WHERE Order_no = ?";
    );

```

And then set this parameter as the second last like this:



```

st.setInt(50, of.getProfile_length_mm());
st.setInt(51, of.getNew_parameter());
st.setInt(52, Integer.parseInt(id));

st.execute();
return true;

```

5. You have now updated the service.

### How to modify a parameter in the backend:

1. Find the parameter you want to modify in the model. Change the name of the parameter or/and datatype everywhere where the parameter you want to change is listed in the model. If you are changing the datatype it is important to also change this everywhere in the form, also in the getfunction of that specific parameter.
2. Go to the corresponding service file and rename the parameter you are changing to the new name everywhere where the parameter is listed. In the getByID and getMostRecent functions it is important to change the getfunction if you have changed the datatype.

```

rs.getString("Sketch_no"),
rs.getInt("Profile_length_mm")

```

3. You have now updated the parameter in the backend

### How to delete a parameter in the backend:

1. Find the parameter you want to delete in the model. Remove this parameter everywhere where it is listed in the model.
2. Head over to the service file. Remove this parameter everywhere where it is listed. For the save and updatefunction it is important to also remove a ? from the SQL query. For these two functions it is also important that the numbering is in the right order according to the parameters in the SQL query:

```

st.setString(41, of.getProfile_cutting_length());
st.setString(42, of.getPlace_of_delivery_2());
st.setString(43, of.getWeek_of_delivery());
st.setString(44, of.getProfile_quantity());
st.setString(45, of.getWbs_number_profile());
st.setString(46, of.getWbs_number_die());
st.setString(47, of.getProfit_center());
st.setString(48, of.getOrder_comment());
st.setString(49, of.getCrush_test());
st.setString(50, of.getSketch_no());
st.setInt(51, of.getProfile_length_mm());

```

# Request

## Database

For changing the database for the Request, we have to write [dbo].[Request] in the name of the table to modify. See the section about adding, modifying and deleting parameters in the database for the rest of the steps.

## Backend

Follow the steps for adding, modifying and deleting steps in the backend above.

## Frontend

The Request data exists in several parts of the frontend code. In the application there is Request data in the Search page, Request page, TFC page, CostCalc page and Order page, this means that if the data is changed it should also be changed in these different pages. There are also textfield in the Request and TFC form that contain Request-data, this might also have to be altered if the Request-data is changed. How to change textfields is written about in a section below.

## Search

To make the Request data in the Search page consistent with the database and backend, you have to make changes in the Search.js file. The search page has Request values in two tables, the Request table and the TFC table. Because of this the search file has 4 states that need to be changed when Request data is updated, these are visibleRequestColumns, requestColumnNames, visibleTfcColumns and tfcColumnNames. These all contain Request data and need to be in the same order as the relevant data in the backend code. These states also exist in other files, and the same changes must be made to them, the other files are written about below.

Path:

SRC → pages → Search.js

```
const [visibleRequestColumns, setVisibleRequestColumns] = useState({
  Request_no: true,
  Request_ver_no: true,
  Request_date: true,
  Due_date: false,
  Sketch_no: true,
  Sketch_rev: false,
  Requestor_name: true,
  Email_requestor: false,
  Product_controller: false,
  Customer: true,
  Project_name: true,
  Project_phase: false,
  Design_type: false,
  Part_category: true,
  Part_description: false,
  SOP: false,
  Place_of_delivery: false,
  Project_SAP: false,
  Project_no: false,
  Average_volume_y: false,
```

```
const [visibleTfcColumns, setVisibleTfcColumns] = useState({
  Request_no: true,
  Request_ver_no: true,
  Request_date: true,
  Sketch: false,
  TFC_date: false,
  TFC_no: true,
  TFC_ver_no: true,
  Profile_type: false,
  Profile_weight_kg_pr_m: false,
  Weight_per_profile_kg: false,
  Profile_length_mm: false,
  Suggested_cutting_length_2_mm: false,
  Suggested_cutting_length_mm: false,
  Profile_area: false,
  Chamber_area_mm2: false,
  Circumscribed_circle_mm: false,
  No_of_chambers: false,
  Profile_heigth_mm: false,
```

To add a new parameter to visibleRequestColumns and visibleTfcColumn all you have to do is to follow the same format as the other parameters, the correct parameter name (the name in the database) and a colon followed by a true or false value depending on whether you want the parameter to be a default parameter in the table or not. To delete a parameter you just find the parameter in the list and delete that line.

```
const requestColumnNames = ({
  Request_no: 'Request Number',
  Request_ver_no: 'Request Version Number',
  Request_date: 'Date',
  Due_date: "Due Date",
  Sketch_no: 'Sketch Number',
  Sketch_rev: 'Sketch revision',
  Requestor_name: 'Request Name',
  Email_requestor: 'Requestor email',
  Product_controller: 'PC Email',
  Customer: 'Customer',
  Project_name: 'Project Name',
  Project_phase: 'Project Phase',
  Design_type: 'Design Type',
  Part_category: 'Part Category',
  Part_description: 'Part Description',
  SOP: 'SOP',
  Place_of_delivery: 'Place of Delivery',
  Project_SAP: 'Project Number SAP',
  Project_no: 'project number',
  Average_volume_y: 'Average Volume',
})

const tfcColumnNames = ({
  Request_no: 'Request Number', //All variables in request
  Request_ver_no: "Request version number",
  Request_date: "Request Date",
  Sketch: 'Sketch',
  TFC_date: "TFC date",
  TFC_no: "TFC number",
  TFC_ver_no: "TFC version number",
  Profile_type: "Profile type",
  Profile_weight_kg_pr_m: "Profile weighth",
  Weight_per_profile_kg: "Weight per profile",
  Profile_length_mm: "Profile length",
  Suggested_cutting_length_2_mm: "Suggested cutting length",
  Suggested_cutting_length_mm: "Suggested cutting length",
  Profile_area: "Profile area",
  Chamber_area_mm2: "Chamber area",
  Circumscribed_circle_mm: "Circumscribed circle",
  No_of_chambers: "Number of chambers",
  Profile_heighth_mm: "Profile heighth",
  Profile_width_mm: "Profile width",
})
```

To add a new parameter to the requestColumnNames and tfcColumnNames you just have to write the correct name and a colon and then the name you want the parameter to have in the table in quotation marks. Make sure that the place of the new parameter is in the same place in the visibleColumns states and also in the corresponding backend code.

## Request

The request pages consist of a search page as well as the form. The search page file “searchRequest.js” contain the states visibleColumns and columnNames which should be updated the same way as visibleRequestColumns and requestColumnNames in the general search file.

The searchRequest.js file also has three more states to modify, the confirmEmpty, permanentEmpty and optionalEmpty state. These states should contain all Request parameters, which means that if you add a new parameter you also have to add it to these and the same goes for any other type of modification. The order of these forms is not important. The confirmEmpty state should have the name of the parameter, and the value after should always be false. The same goes for permanentEmpty, but the value can be true if the value of it is permanent/uneditable. With the optionalEmpty state the value following the name should state whether or not the values should be required or not in the forms.

```
const [confirmEmpty] = useState({
  Request_no: false,
  Request_date: false,
  Requestor_name: false,
  Email_requestor: false,
  Product_controller: false,
  Customer: false,
  Project_SAP: false,
  Project_no: false,
  SOP: false,
  Place_of_delivery: false,
  Average_volume_y: false,
  Peak_volume_y: false,
  Volume_total: false,
  Part_description: false,
  Part_category: false,
  Part_length: false,
  Parts_pr_vehicle: false,
  SAP_CAD_model: false,
  Sketch_no: false,
})

const [permanentEmpty] = useState({
  Request_no: false,
  Request_date: false,
  Requestor_name: false,
  Email_requestor: false,
  Product_controller: false,
  Customer: false,
  Project_SAP: false,
  Project_no: false,
  SOP: false,
  Place_of_delivery: false,
  Average_volume_y: false,
  Peak_volume_y: false,
  Volume_total: false,
  Part_description: false,
  Part_category: false,
  Part_length: false,
  Parts_pr_vehicle: false,
  SAP_CAD_model: false,
  Sketch_no: false,
})

const [optionalEmpty] = useState({
  Request_no: false,
  Request_date: false,
  Requestor_name: false,
  Email_requestor: false,
  Product_controller: false,
  Customer: false,
  Project_SAP: false,
  Project_no: true,
  SOP: false,
  Place_of_delivery: false,
  Average_volume_y: false,
  Peak_volume_y: false,
  Volume_total: false,
  Part_description: false,
  Part_category: false,
})
```

## Requestform files:

The next step is to navigate to requestForm.js which is located in the same folder as searchRequest.js. The class in this file has a state that consists of all the parameters in the request form. It is important to also add the new parameter to this state. Under the render method we also have a const that pulls out every value from the state so here you also have to add this new parameter. Under this we have a new const that sets all the parameters to the name "values". Here you also need to add the new parameter.

1. Next step is to add a textfield corresponding to this parameter. See the section about How to add, change or delete a simple textfield in the forms for that.

## TFC

In the TFC search page Request-data is shown in two tables just like the general search page. There are 5 states in the TFC.js file that need to be updated when the Request database is updated.

These are requestValuesEmpty, which is a state containing all the request parameters and an empty string following it, this state is used to send request values to the form. The next states are visibleTfcColumns and columnTfcNames, these contain all TFC parameters and Request parameters together and are the parameters shown in the TFC table. These states need to be the same as the general search page and be in the same order as the backend code. The last state is visibleRequestColumns and columnRequestNames which contain all the parameters shown in the Request table, this also needs to be in the same order as the states in the general search file and in the corresponding backend code.

## CostCalc

The CostCalc page contains another search page, this time only with the TFC table. Similarly to the TFC-table in the search and TFC page, the table contains both TFC and Request values. Therefore you must change the states visibleTfcColumns and tfcColumnNames here too.

## Order

There is a Request-table in the order search page, which means the states visibleRequestColumns and columnRequestNames must be changed too whenever the Request database is changed. These should be changed the same way as in the search.js , searchRequest.js and TFC.js files earlier.

## TFC

### Database

For changing the database for the TFC, we have to write [dbo].[TFC] in the name of the table to modify. See the section about adding, modifying and deleting parameters in the database for the rest of the steps.

## Backend

Follow the steps for adding, modifying and deleting steps in the backend above.

## Frontend

The TFC data exists in several parts of the frontend code. In the application there is TFC-data in the Search page, TFC page and Cost Calc page, this means that if the data is changed it should also be changed in these different pages.

### Search

To make the TFC data in the Search page consistent with the database and backend, you have to make changes in the Search.js file. This file has two states that needs to be changed when TFC data is updated, these are visibleTfcColumns and tfcColumnNames. These both contain TFC-data and Request-data and need to be in the same order as the relevant data in the backend code. These states exist in every search page containing TFC data, which means they should be changed in everyone of those files (TFC.js, CostCalc.js).

Path:

SRC → pages → Search.js

```
const [visibleTfcColumns, setVisibleTfcColumns] = useState({
  Request_no: true,
  Request_ver_no: true,
  Request_date: true,
  Sketch: false,
  TFC_date: false,
  TFC_no: true,
  TFC_ver_no: true,
  Profile_type: false,
  Profile_weight_kg_pr_m: false,
  Weight_per_profile_kg: false,
  Profile_length_mm: false,
  Suggested_cutting_length_2_mm: false,
  Suggested_cutting_length_mm: false,
  Profile_area: false,
  Chamber_area_mm2: false,
  Circumscribed_circle_mm: false,
  No_of_chambers: false,
  Profile_heigth_mm: false,
  Profile_width_mm: false,
})
```

```
const tfcColumnNames = ({
  Request_no: 'Request Number', //All variables in request
  Request_ver_no: "Request version number",
  Request_date: "Request Date",
  Sketch: 'Sketch',
  TFC_date: "TFC date",
  TFC_no: "TFC number",
  TFC_ver_no: "TFC version number",
  Profile_type: "Profile type",
  Profile_weight_kg_pr_m: "Profile weighth",
  Weight_per_profile_kg: "Weight per profile",
  Profile_length_mm: "Profile length",
  Suggested_cutting_length_2_mm: "Suggested cutting length",
  Suggested_cutting_length_mm: "Suggested cutting length",
  Profile_area: "Profile area",
  Chamber_area_mm2: "Chamber area",
  Circumscribed_circle_mm: "Circumscribed circle",
  No_of_chambers: "Number of chambers",
  Profile_heigth_mm: "Profile heighth",
  Profile_width_mm: "Profile width",
  Wall_thickness_max_mm: "Maximum wall thickness",
  Wall_thickness_min_mm: "Maximum wall thickness",
  Corner_rad_inner_min: "Corner rad inner min",
})
```

### TFC

The TFC pages consists of a search page, as well as the TFC form. This means that in the same way as the general search page the TFC data in the table has to be consistent with the database and backend. This is done by changing the visibleTfcColumns and tfcColumnNames states in the TFC.js file, these states should be the same as in Search.js. In addition to visibleTfcColumns and tfcColumnNames, the TFC.js file has another state called tfcValuesEmpty. This state contains every TFC parameter (Not Request parameters) and is used to send data to the forms. When updating the TFC database and backend code it is also important to update this state, with the correct name and an empty string following it.

Path:

SRC → pages → TFC → TFC.js

```
const [tfcValuesEmpty] = useState ({
  Request_no: "",
  Request_ver_no: "",
  Request_date: "",
  Sketch: "",
  TFC_date: "",
  TFC_no: "",
  TFC_ver_no: "",
  Profile_type: "",
  Profile_weight_kg_pr_m: "",
  Weight_per_profile_kg: "",
  Profile_length_mm: "",
  Suggested_cutting_length_2_mm: "",
  Suggested_cutting_length_mm: "",
  Profile_area: "",
  Chamber_area_cm2: "",
  Circumscribed_circle_mm: "",
  No_of_chambers: "",
  Profile_height_mm: "",
  Profile_width_mm: "",
  Wall_thickness_max_mm: "",
  Wall_thickness_min_mm: "",
  Corner_rad_inner_min: "",
  Corner_rad_outer_min: "",
  Alloy: "",
  Temper: "",
  Press: "",
  Accel_time_s: "",
  Reduction_ratio: "",
  Billet_diameter_mm: "",
  Container_area: ""
})
```

## Cost Calc

Cost calc also has a search page with TFC data that needs to be updated if the TFC data is changed. The file to make these updates is the CostCalc.js file, and it contains the states visibleTfcColumns and tfcColumnNames just like the other search pages with TFC data, and needs to be updated in the same way.

Path:

SRC → pages → CostCalc.js

## Order

### Database

For changing the database for the Order table, we have to write [dbo].[Order] in the name of the table to modify. See the section about adding, modifying and deleting parameters in the database for the rest of the steps.

### Frontend

2. For adding parameters to the frontend of order we have to first navigate to:

src → pages → Order → SearchOrder.js

Here we have the visibleOrderColumns and columnOrderNames that have to be in the same order as the parameters are in the backend. So if you have added the new parameter as the last one in the backend you have to do that also in visibleOrderColumns and columnOrderNames like this:

```

Other_tolerances: false,
Tensile_test: false,
Structure_test: false,
Split_test: false,
Certificate: false,
BTM: false,
Profile_cutting_length: false,
Place_of_delivery_2: false,
Week_of_delivery: false,
Profile_quantity: false,
WBS_number_profile: false,
WBS_number_die: false,
Profit_center: false,
Order_comment: false,
Crush_test: false,
Sketch_no: false,
Profile_length_mm: false,
New_parameter: false
})

Other_tolerances: 'Other tolerances',
Tensile_test: 'Tensile test',
Structure_test: 'Structure test',
Split_test: 'Split test',
Certificate: 'Certificate',
BTM: 'BTM',
Profile_cutting_length: 'Profile cutting length',
Place_of_delivery_2: 'Place of delivery 2',
Week_of_delivery: 'Week of delivery',
Profile_quantity: 'Profile quantity',
WBS_number_profile: 'WBS number profile',
WBS_number_die: 'WBS number die',
Profit_center: 'Profit center',
Order_comment: 'Order comment',
Crush_test: 'Crush test',
Sketch_no: 'Sketch number',
Profile_length_mm: 'Profile length mm',
New_parameter: "New parameter",
})

```

The columnOrderNames is the name of the columns, so here you have to write the name that you want to call the column that will be displayed in the frontend. This searchOrder file also has a enabledFields const that has the task to set which fields should be editable or not in the form. If the new parameter you are adding is not going to be editable if for example it is sent from a request you can add the new parameter in this list, and set it to true which means that the parameter is editable. We also have to add this new parameter to the async function enablefields when we are adding the parameter to the enabledFields const. If you do not want the parameter to be editable when it is coming from a request you have to add this code to the async function enablefields:

```

Profile_quantity: true,
WBS_number_profile: selectedData.WBS_number_profile ? false : true,
WBS_number_die: selectedData.WBS_number_die ? false : true,
Profit_center: selectedData.Profit_center ? false : true,
Order_comment: true,
Crush_test: selectedData.Crush_test ? false : true,
Sketch_no: selectedData.Sketch_no ? false : true,
Profile_length mm: selectedData.Profile_length mm ? false : true,
New_parameter: selectedData.New_parameter ? false : true,

```

3. The changes made to the visibleOrderColumns and columnOrderNames does also have to be done in the search.js file.
4. The next step is to navigate to orderform.js which is located in the same folder. The class in this file has a state that consists of all the parameters in the order form. It is important to also add the new parameter to this state. Under the render method we also have a const that pulls out every value from the state so here you also have to add this new parameter. Under this we have a new const that sets all the parameters to the name "values". Here you also need to add the new parameter.
5. Next step is to add a textfield corresponding to this parameter. See the section about How to add, change or delete a simple textfield in the forms for that.

When modifying a textfield you just have to change the name of the parameters in all the places we have been through here. When deleting a parameter you simply have to remove the line of code with the specific parameter in all the places we have been through here.

## Backend

Follow the steps for adding, modifying and deleting steps in the backend above.

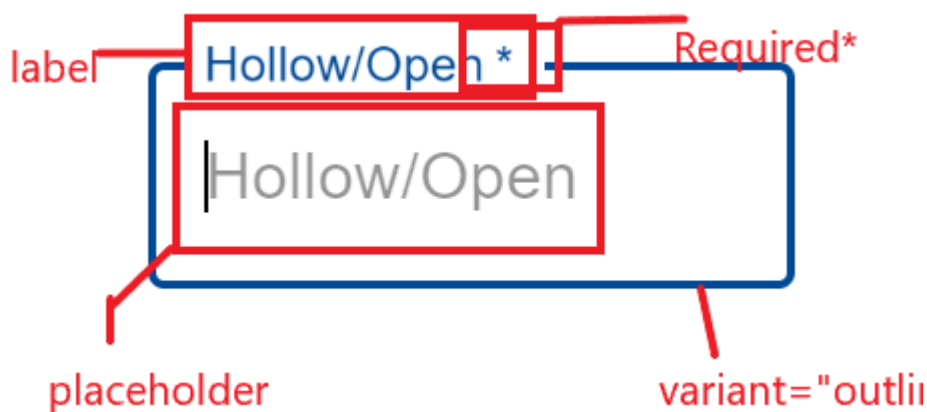
## How to add, change or delete a simple textfield in the forms.

To explain how to add, change and delete textfields, we need to explain the textfield a bit first. The textfields used in the request, order and tfc are a Material-ui component (<https://material-ui.com/api/text-field/>).

```

<TextField
  required                //This bool makes the label displayed as required.
  id="Profile_type"      //The id of the textfield. Every component should have a unique id.
  name= "Profile_type"  //The name of the component. Used in the handleChange-function. Should have the same name as the variable in the database.
  placeholder="Hollow/Open" //If the textfield is empty, this string is inside the textfield. Has no other functionality.
  label="Hollow/Open"   //The label of the textfield. Is shown on top of the textfield.
  margin="normal"      //
  variant="outlined"   //Style of the textfield.
  defaultValue={tfcValues.Profile_type} //The default value. Is changeable.
  onChange={handleChange} //The function called in the onChange is called everytime the value in the textfield is changed.
                        //Can also use onBlur witch runs the function inside when the textfield loose focus.
  inputProps={{        //Attributes applied to the input-element.
    maxLength:1,      //Here the max possible length of the text in the textfield is set to 1.
    style: {
      width: "15ch",  //Set the width of the textfield. "15ch" means 15 characters width.
    },
  }}
  className={classes.textField2} //ClassName classes used to override the default styles applied to the component.
/>

```



This is some of the props that can be used in a textfield.

Things to think of when **adding** a textfield:

- Does the value exist in a state? To save the coming value in a state, the value has to be defined somewhere.



- Has the textfield been given an id, a correct name, a label and a defaultValue/value? This is the most important props of the textfield. If the textfield is only used for output you can use value, else use defaultValue.
- Does the textfield have an onChange/onBlur function? This is only needed if the value is supposed to be changeable.

When **deleting** a textfield, the only thing you really need to do is remove the textfield code. But it would be a good idea to clear up the rest of the frontend and backend, if the value is no longer in use.

When **updating** a textfield, you can use the material ui documentation to see what can be done with the textfields.

# How to re-deploy the application/update the application.

Project need to be cloned first from this repository using “Git Bash”

<https://gitlab.stud.iie.ntnu.no/kritveit/extrusion-planner/-/tree/master/extrusionAPI/src/main/java/com/benteler/extrusionAPI>

**In Git Bash:** Navigate to a desired folder then use the command “git clone” following the HTTP clone request link found in the repository, it is possible that your username must be permitted to access the repository so contact us if it is necessary.

Required software that must be installed:

**Node.JS** - (<https://nodejs.org/en/>)

Node is a JavaScript runtime that is used to compile the react application and build the application for production.

**Java** - (<https://www.oracle.com/java/technologies/javase-downloads.html>)

Java JDK is needed for the backend API and the code editor IntelliJ. Install the newest version of the Java JDK, make sure it is the JDK and **not** JRE!.

**Visual Studio Code** - (<https://code.visualstudio.com/>)

Visual Studio Code is a code editor we used to develop the React frontend (Web interface).

**IntelliJ Community Edition-**

(<https://www.jetbrains.com/idea/download/#section=windows>)

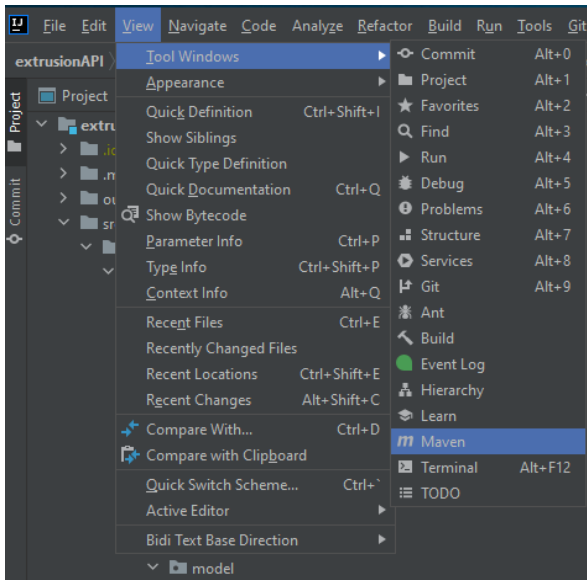
IntelliJ Community Edition is the free version of IntelliJ used by us to develop the Java Backend API.

**Git Bash** - (<https://gitforwindows.org/>)

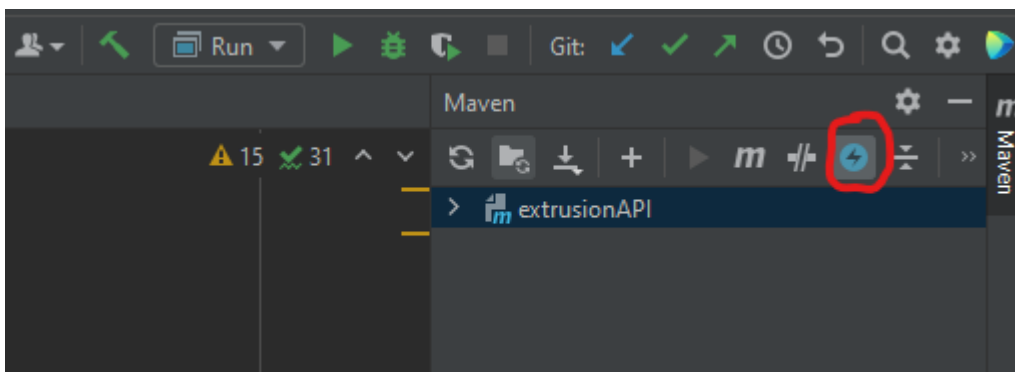
Code revision tool for downloading from repositories and pushing code updates to the repository.

## Procedure (Backend Java API)

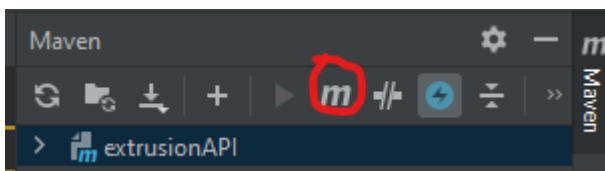
1. Save changes, make sure that it compiles correctly by running it:  
<https://www.jetbrains.com/help/idea/running-applications.html#rerun>
2. **Note** that the database connection will return an error if the application is not run on the deployment server.
3. To build the project into a JAR, open the Maven tool window:



4. Then you will get a Maven tool window on the right.
5. Press the blue lightning icon to skip tests if this is done outside of the deployment server.



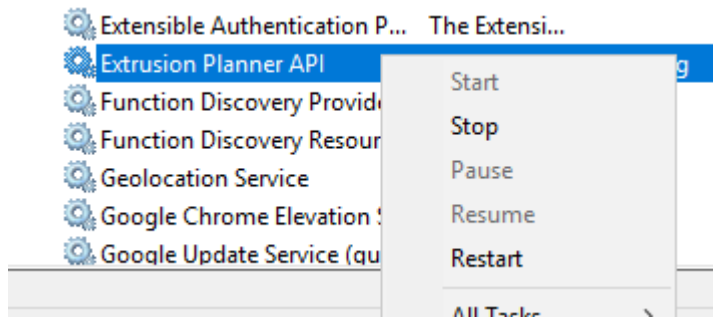
6. Then press "Execute Maven Goal"



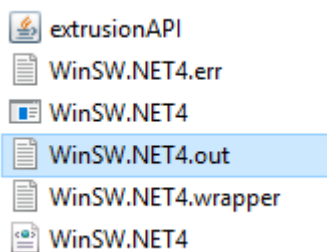
7. Select/write "maven package", and the JAR will be created in the target folder (extrusion-planner/extrusionAPI/target)

### ON THE WEB SERVER/DEPLOYMENT SERVER:

8. Open "Local Services" in Windows by searching in the Windows taskbar.
9. Find the "Extrusion Planner API" service, right click and press "Stop"



10. Navigate to the folder containing the “WinSW” service installer/the JAR, currently “C:\ExtrusionPlanner-APIService”, and replace the JAR file with the newly packaged one. (Must be the same name)
11. Right click the “Extrusion Planner API” service again and press “Start”
12. The backend is now updated! If you need to see the program output, see the bottom of the “WinSW.NET4.out” file, it is updated each time it's opened (so not live).



## Procedure (Frontend React web interface)

(In the terminal window in Visual Studio, run “npm install react-scripts” to install the necessary scripts for the following steps.)

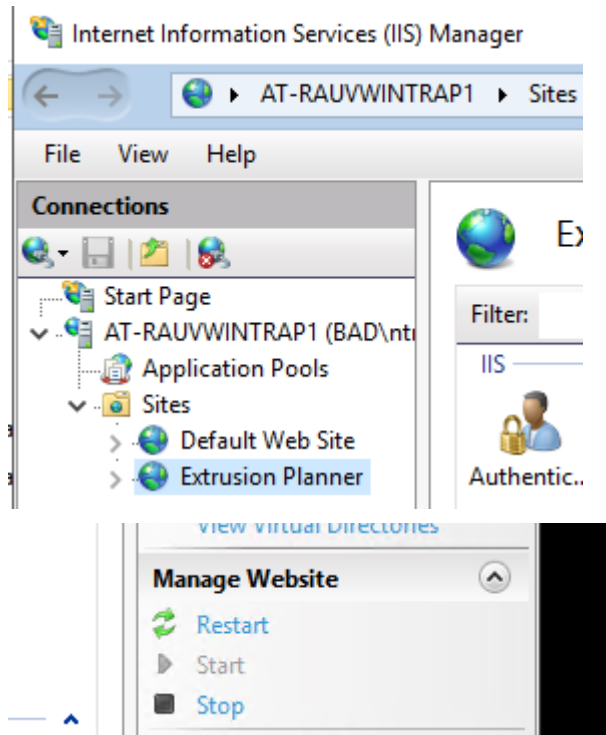
1. Save changes and make sure the program works as expected by running it using the command “npm start” in the terminal window in the bottom of Visual Studio Code, is it not open? See the top toolbar and press Terminal > New Terminal. (Must be connected like you would normally to reach the backend API to ensure that the application works during testing)
2. To build the application, make sure you are navigated in the root project folder for the frontend in the Terminal in Visual Studio Code. (Can be done without VS Code in the normal command prompt too as long as you are in the right folder.)

```
h\Documents\extrusion-planner\frontend> 
```

3. Run the command “npm run-script build”
4. Look for the “build” folder in the root frontend directory “frontend”, now replace the “build” folder on the deployment server (currently in documents), **NB: DO NOT DELETE “web.config”, this will break the web server configuration for IIS.**

Name	
asset-manifest.json	4
index	4
web.config	4
bentelerlogo	2
favicon	2
logo192	2
logo512	2
manifest.json	2
robots	2
static	4

5. Go into IIS (Internet Information Services) and navigate to the web page “Extrusion Planner” and restart it from the right tool window.



6. The frontend is now updated!

## How to add or delete an Alloy in the ranked list on dropdowns.

The ranked list of the most popular Alloys exists in the dropdown menu in 8 places in the code, so to update this list you must update it in all 8 places. The code is very similar for all the 8 places, so once you change it in one file you just have to repeat it on the other files. If you want the Alloy dropdown field to be similar in all forms, you have to make sure that the list in the code for each form is the same.

```

<FormControl variant="outlined" className={classes.select}>
  <InputLabel>Alloy</InputLabel>
  <Select
    value={values.Alloy || ""}
    onChange={handleChange("Alloy")}
    label="Alloy"
    disabled={!enabledFields.Alloy}
    MenuProps={{autoFocus: false}}
    >
    <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
    <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
    <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
    <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
    <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
    <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
    <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
    <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
    <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
    <Divider />
    {this.props.params.alloys.map(alloy =>
      <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
    )}
  </Select>
</FormControl>

```

This is the code for all Alloy field in the application. And to add or delete an Alloy to the ranked list in the dropdown area, you just have to change the code within the red circle on the picture. If you want to add a new Alloy to the list, you have to first make sure that the Alloy exists in the list of Alloys in the application (you can check this in the Edit Database page).

All you need to do to add a new Alloy to the list is to add another "MenuItem" to the list with the relevant key and value. For example if you want to add Alloy 6060.85 to the list.

```
<MenuItem key="6060.85" value="6060.85">6060.85</MenuItem>
```

You just add this line wherever you want it in the list, if you want it on top of the list you place the line first. Make sure to place the line after the > tag and before the <Divider /> tag.

If you want to delete an Alloy from the list you just have to find the MenuItem with that specific Alloy and remove that line.

**In request form:**

In request form the Alloy dropdown field exists in two places, in page two "Profile" and in the confirmPage. The files "Request2.js" and "confirmRequest.js" must be edited to change the dropdown list in these places.

Path:

SRC → pages → Request → Request2.js

SRC → pages → Request → confirmRequest.js

### Request2.js:

```

173 <FormControl variant="outlined" className={classes.select}>
174   <InputLabel>Alloy</InputLabel>
175   <Select
176     value={values.Alloy || ""}
177     onChange={this.props.confirm.Alloy ? (this.submit("Alloy", values.Alloy)) : handleChange("Alloy")}
178     label="Alloy"
179     disabled={this.props.permanent.Alloy}
180     MenuProps={{autoFocus: false}}
181   >
182     <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
183     <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
184     <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
185     <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
186     <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
187     <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
188     <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
189     <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
190     <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
191     <Divider />
192     {this.props.params.alloy.map(alloy =>
193       <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
194     )}
195   </Select>
196 </FormControl>

```

### confirmRequest.js

```

928 <FormControl variant="outlined" className={classes.select}>
929   <InputLabel>Alloy</InputLabel>
930   <Select
931     value={values.Alloy || ""}
932     onChange={this.props.confirm.Alloy ? (this.submit("Alloy", values.Alloy)) : handleChange("Alloy")}
933     label="Alloy"
934     disabled={this.props.permanent.Alloy}
935     MenuProps={{autoFocus: false}}
936     error={optional.Alloy ? null : (!values.Alloy)} //Display error if empty
937   >
938     <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
939     <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
940     <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
941     <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
942     <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
943     <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
944     <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
945     <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
946     <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
947     <Divider />
948     {this.props.params.alloy.map(alloy =>
949       <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
950     )}
951   </Select>
952   <FormHelperText className={classes.helperText}>{optional.Alloy ? null : (
953     //Display helptext if empty
954     !values.Alloy ? "This field is required!" : " "
955   )}</FormHelperText>
956 </FormControl>

```

In TFC form:

In TFC form the Alloy dropdown field exists in two places, in page two "Profile" and in the confirm page. The TFC form code is divided in two, one part for the metrics system and another one for imperial units. This means you must change 4 files to add or delete Alloys in the ranked list, this is the files "TFCForm2.js", "TFCCConfirmForm.js", "TFCUSForm2.js" and "TFCUSConfirmForm.js"

The code in the TFC form2 and in the confirm forms is similar to the ones in Request, and you can follow the same steps to add or delete an Alloy on the list.

Path:

SRC → pages → TFC → TFCForm2.js

SRC → pages → TFC → TFCCConfirmForm.js

SRC → pages → TFC → TFCUS → TFCUSForm2.js

SRC → pages → TFC → TFCUS → TFCUSConfirmForm.js

### TFCForm2.js and TFCUSForm2.js

```

248 <FormControl variant="outlined" className={classes.select}>
249   <InputLabel>Alloy</InputLabel>
250   <Select
251     value={tfcValues.Alloy || ""}
252     name="Alloy"
253     onChange={(e) => {handleInputChange(e);}}
254     label="Alloy"
255     MenuProps={{autoFocus: false}}
256   >
257     <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
258     <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
259     <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
260     <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
261     <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
262     <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
263     <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
264     <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
265     <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
266   <Divider />
267   {params.alloy.map(alloy =>
268     <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
269   )}
270 </Select>
271 </FormControl>

```

### TFCCConfirmForm.js and TFCUSConfirmForm.js

```

1258 <FormControl variant="outlined" className={classes.select2}>
1259   <InputLabel>Alloy</InputLabel>
1260   <Select
1261     value={tfcValues.Alloy || ""}
1262     onChange={(e) => {
1263       handleInputChange(e);
1264     }}
1265     label="Alloy"
1266     name="Alloy"
1267     MenuProps={{autoFocus: false}}
1268     error={!tfcValues.Alloy} //Display error if empty
1269   >
1270     <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
1271     <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
1272     <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
1273     <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
1274     <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
1275     <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
1276     <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
1277     <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
1278     <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
1279   <Divider />
1280   {params.alloy.map(alloy =>
1281     <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
1282   )}
1283 </Select>
1284 <FormHelperText className={classes.helperText}>{
1285   //Display helptext if empty
1286   !tfcValues.Alloy ? "This field is required!" : " "
1287 }</FormHelperText>
1288 </FormControl>

```

In Order form:



In Order form the Alloy dropdown field also exists in two places, in page two “Profile” and in the confirm page. You must change the two files “Order2.js” and “confirmOrder.js” to change the ranked list. As you can see from the pictures below this code is the same as the other places, and all you have to do is to repeat what you did with the other files.

Path:

SRC → pages → Order → Order2.js

SRC → pages → Order → confirmOrder.js

### Order2.js

```
241 <FormControl variant="outlined" className={classes.select}>
242 <InputLabel>Alloy</InputLabel>
243 <Select
244   value={values.Alloy || ""}
245   onChange={handleChange("Alloy")}
246   label="Alloy"
247   disabled={!enabledFields.Alloy}
248   MenuProps={{autoFocus: false}}
249 >
250   <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
251   <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
252   <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
253   <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
254   <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
255   <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
256   <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
257   <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
258   <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
259   <Divider />
260   {this.props.params.alloys.map(alloy =>
261     <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
262   )}
263 </Select>
264 </FormControl>
```

### confirmOrder.js

```
743 <FormControl variant="outlined" className={classes.select}>
744 <InputLabel>Alloy</InputLabel>
745 <Select
746   value={values.Alloy || ""}
747   onChange={handleChange("Alloy")}
748   label="Alloy"
749   disabled={!enabledFields.Alloy}
750   MenuProps={{autoFocus: false}}
751 >
752   <MenuItem key="6060.35" value="6060.35">6060.35</MenuItem>
753   <MenuItem key="6063.85" value="6063.85">6063.85</MenuItem>
754   <MenuItem key="6005.40" value="6005.40">6005.40</MenuItem>
755   <MenuItem key="6005.63" value="6005.63">6005.63</MenuItem>
756   <MenuItem key="6082.26" value="6082.26">6082.26</MenuItem>
757   <MenuItem key="6082.27" value="6082.27">6082.27</MenuItem>
758   <MenuItem key="7003.30" value="7003.30">7003.30</MenuItem>
759   <MenuItem key="7003.62" value="7003.62">7003.62</MenuItem>
760   <MenuItem key="7108.50" value="7108.50">7108.50</MenuItem>
761   <Divider />
762   {this.props.params.alloys.map(alloy =>
763     <MenuItem key={alloy.Alloy_name} value={alloy.Alloy_name}>{alloy.Alloy_name}</MenuItem>
764   )}
765 </Select>
766 </FormControl>
```

## Suggested reading/viewing:

A starter guide for react:

<https://www.w3schools.com/react/>

React hooks (State hook and Effect hook most relevant):

<https://reactjs.org/docs/hooks-overview.html>

Spring boot, and how to test the backend using postman:

<https://www.youtube.com/watch?v=vtPkZShrvXQ> (00:20:00 -> 01:00:00 is the most relevant)

A small intro to Axios:

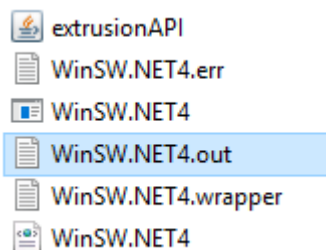
<https://www.youtube.com/watch?v=qM4G1Ai2ZpE>

## The server setup (if the application is to be moved to another server)

(This procedure does not include the database as it is not necessary to set up again)

Steps:

1. Enable IIS in Windows (Must be Windows 10) how to :  
<https://enterprise.arcgis.com/en/web-adaptor/latest/install/iis/enable-iis-10-components-server.htm>
2. Install CORS Module <https://www.iis.net/downloads/microsoft/iis-cors-module>
3. Install URL Rewrite Module <https://www.iis.net/downloads/microsoft/url-rewrite>
4. Deploy the frontend web interface with IIS using the existing build folder or a newly created one following the mentioned redeployment procedure in the other chapter:  
[https://www.youtube.com/watch?v=eA9p68rjuJI&ab\\_channel=SteveYu](https://www.youtube.com/watch?v=eA9p68rjuJI&ab_channel=SteveYu)
5. Make sure the “web.config” file from the build folder from the old server is copied into the new build folder on the new server, this ensures correct configuration of the URL Rewrite module and CORS module.
6. Copy the entire ExtrusionPlanner-APIService folder at “C:\ExtrusionPlanner-APIService” to a desired location on the new server.
7. Then simply run the WinSW.NET4.exe executable to install the service for the API.
8. Navigate to Local Services in Windows and find “Extrusion Planner API”, right click and select “Start” to run the service, voila the API and web interface is up and running!



## **Appendix C**

# **Feedback from testing phase with solutions and status per case**

List of OpenPoints

<b>Topic:</b> Extrusion Planner Application	<b>Language:</b> English
<b>Responsible:</b> GLN	<b>Creator:</b> GLN
<b>Last change:</b> 30/04/21	



No.	Date	Triggered by	Area	Open issue / Finding	Action and expected result and/or deliverable	Due date	Responsible	Supported by	Status
1	4/19/2021	GLN	Innlogging	Lukket vindu: Mulig å gå inn igjen uten passord etter å ha besøkt annen web side i samme tab, med "back" knapp i nettleser	Innlogging varer i en time med mindre nettleser lukkes eller bruker trykker "logg av". Denne timen blir oppdatert etter aktivitet, dvs at etter en time med inaktivitet logges brukeren ut.	21/04/21			Done
2	4/19/2021	GLN	Tilgangskontroll	bruker oversikt: viser ikke alle brukere	På grunn av glipp under oppdatering kom feil IP med i websiden som førte til at feil API/Database ble sett	21/04/21	Kristian		Done
3	4/19/2021	GLN	Innlogging	pålogging: ikke mulig å logge på hjemmefra selv med fullt navn i adressefeltet i stedet for ip adresse	Går ikke å logge på via intern nettverk, kun med VPN GLN: Sender forespørsel til Benteler Service Desk		Kristian		in work
4	4/20/2021	GLN	Request	Autofill: Databasen burde "lære" eller tilby autofill på navn av forespørere / e-post requestor / e-post product controller	Alternative nå er å komme med en liste vi kan legge inn i en dropdown meny. Eventuelt tilby et eget felt på database siden for å legge til innhold i dropdown listen. 27.04: Har implementert autoinnfylling av requestor name og email requestor. Også lagt til dropdown på product controller email. Man kan legge til flere product controllere på databasesiden 30.04: Liste med product controllere er sendt til Jesper		Jesper		Done
5	4/20/2021	GLN	General	Cost-Calc: Personen heter nå: Product controller --> Email cost calc burde hete e-mail Product Controller også i databasen som feltnavn. Cost calc som funksjon kan fortsatt beholdes tenker jeg	Løst	21/04/21			done
6	4/20/2021	GLN	Request	Alloy: Drop list her må sorteres som i forrige request skjema. Dvs de mest brukte legeringer overs i listen , deretter i stigende rekkefølge	Hent sortert liste fra tidligere program og sorter det i den rekkefølge i request skjema	23/04/21	Emma		Done
7	4/20/2021	GLN	TFC	Profile: Profile nummer kan bestå av både tall, bokstaver og spesialtegn, MA ikke fylles ut., kan autofylles fra database dersom kjent	Fikse sånn at det ikke er et nummer i feltet, og at det ikke er required. Blir hentet fra request om det er mulig.	21/04/21	Emma		Done
8	4/20/2021	GLN	FAQ	Request: Forklare at på Request skjema under PPart description = Part name		21/04/21	Kristian		Done
9	4/20/2021	GLN	Unit	Switch between units: Prøve bytte fra SI til US enheter midt i en TFC for å se hvor mange lb presserer var, da hoppet den ut av systemet og tilbake til hovedmeny. Her må vi kunne switche fra enhet til enhet og fortsatte på TFC utfyllingen underveis.	NT: Knappen er helt disabled under tfc, fra første verdien er endret. Den blir disabled til brukeren har gått ut av tfc gjennom Back-knapp eller tfc er sendt. NB: Hvis man går ut av TFC gjennom sidemenyen vil ikke switchen bli enablea igjen automatisk, da må man refreshe siden, jobber med saken.	23/04/21	Nils		in work
10	4/20/2021	GLN	TFC	Process: Last billet additional scrap må være et felt som kan fylles inn. Pluss litt mer utfyllende navn.	Endret navn til Last billet additional scrap, og gjør det feltet mulig å endre på	23/04/21	Emma		Done
11	4/20/2021	GLN	General	Desimalskilletegn: Her bør både komma og punktum aksepteres hvis mulig	Må finne hvilke felt det gjelder TFC - based on request "Profile Weight" på 2. Circumscribed Circle samme side, Chamber area (skjønt her bruker vi ikke desimaler) Non performance scrap, Safety margin og alle avkapp lengder front / rear, hastighet side 4, Konklusjon - dere må ganske enkelt teste alle felter i alle skjema og verifisere at det 28.04.21: Benteler sliter med bruk av komma, studentene har ikke problem med komma. Hva kan det skyldes?				Escalation
12	4/20/2021	GLN	TFC	Saw: Cavities stacked - MA fylles ut kun hvis: Press- No of Cavities > 1, skal IKKE fylles om No of Cavities = 1 så default verdi kan være ved siden av hverandre eller ingen verdi, det kommer litt an på hvordan denne verdien brukes videre	Gjøre "side-by-side" til default verdi, sånn at man kan bla forbi feltet uten å gjøre en endring. Om No of cavities er 1 skal ikke feltet gå an å endre på.	23/04/21	Emma		Done
13	4/20/2021	GLN	TFC	Chamber area - kriterie feil. Dette kan jeg sende en tabell på. Sjekkliste legering og presse --> utvalg av max chamber area Max 80 som dere har brukt skal brukes på feltet "Reduction Ratio" under Presse. Det må også kunne bekrefte og tillates > 80 men det skal markeres rødt og komme warning.	Endre målenhet på chamber area. Må ha mer informasjon om Reduction Ratio grensen.	21/04/21	Emma		Done
14	4/20/2021	GLN	TFC	Send: Fylte ut TFC og trykkte "send", fikk feilmeldingen "Failed to add to database" og alle data vekk. Dersom den ikke kan skrives til databasen, bør oppsummeringssiden stå og evt kunne printes så ikke alle data går tapt. Evt kunne eksporterers ved feil på skrivning til databasen? De bør ikke bare forsvinne i alle fall.	Sørge for å ikke resette siden ved feilet databaseinnsending, reset kun ved vellykket innsending. Ferdig på TFC, request og order	28/04/21	Jesper		Done
15	4/20/2021	GLN	General	Print av de ulike skjema gjerne fra "Confirm" siden hadde vært nyttig.	Knapp for å printe ut utfylling til PDF på siste side av hvert skjema, som backup. Velger å bruke excel istedet da dette fungerer bedre med mange parametre	28/04/21	Jesper		Done
16	4/20/2021	GLN	Search	Søke i request versjons nr mangler, min feil - spesifiserte sketch rev men ser at vi trenger også request revisjon i tillegg når vi skal søke opp forespørsler. Når det kommer til Sketch revisjon - så må bokstaven i Sketch nr flyttes ut av cellen og inn i sketch rev nr kolonnen.	Legge til request ver nr som default visningsparameter i de ulike tabellene. Også legge til sketch rev i skjemaet slik at brukerne kan legge inn bokstaven i skjema		Emma		Done
17	4/20/2021	GLN / KIA	Search	Mulighet for å endre rekkefølge på feltene som vises?	Vil ikke prioriteres, kommer til å ta mye tid.	01/08/21			not started
18	4/20/2021	GLN / KIA	Cost Estimation	Cost-Calc: ikke implementert enda? Her må en kunne søke og velge request nr og versjon, TFC nr og versjon som skal eksporteres til excel.	Cost calc funksjonen fungerer, dog ikke med det originale skjemaet da det ikke var mulig å redigere	30/04/21	NTNU	Gerda	In work
19	4/20/2021	KIA/GLN	Request	Skulle hatt funksjonen "View Request" Foreslår å legge til denne som egen funksjon i tillegg til "Correct request". View request blir da kun en "død" visning av en gitt request basert på nr og index		21/04/21	Kristian		Done
20	4/20/2021	KIA	Request	Endre rekkefølge på felter i Project. Øverst fra venstre: Customer - Proje name - SAP - PPA - Delivery - SOP Nederst fra venstre: Proj phase - Design - Avg volume - Peak volume - Total volume		21/04/21	Jesper		Done
21	4/20/2021	KIA	TFC	Process - "Gross extrusion length" burde kopieres inn slik at den også vises på "Press" da denne er gitt av input på "Billet length" og da bør vi se allerede her at det er for lang bolt	Legge Gross extrusion length både i Press og i Process.	21/04/21	Emma		Done
22	4/20/2021	KIA	TFC	Oversikten på venstre side burde også vise Request data input da dette er viktig å ha med seg i evalueringen	Innført	21/04/21	Jesper		Done
23	4/20/2021	KIA	TFC	Highlighte noen verdier av spesiell interesse i confirm siden: Presse, profilhastighet, net productivity, recovery, die cost pr kg		21/04/21	Jesper		Done
24	4/20/2021	KIA	Request	Change request: Skal ikke være lov å registrere en change request med mindre minimum CAD / SAP nr er endret.	Har endra så CAD / SAP må endres på change. Warning på textboxen og disablea send-knapp til den er endra. 26.04. GLN: Ser at vi har vært for raske på labben nok en gang. Det er visstnok mulig å endre slik at det oppfyller krav til "change request" uten nytt SAP / CAD nr, så regelen burde bli endret: Minst en av følgende verdier må være endret: SAP/CAD nr., alloy, temper, customer standard, rex, igc, ductility, geometric std	30/04/21	Nils		done
25	4/20/2021	KIA	Request	Change request: Skal ha varsel om "er du sikker på at du vil endre..." kun dersom verdien er endret, ikke når vi bare har klikket innom feltet	Har fikset feilen på de boksene det gjaldt. Endret funksjonen som sjekker om verdien er endret til å sammenligne uavhengig av number og String.	21/04/21	Nils		Done
26	4/20/2021	KIA	General	Farge hvit - grå foretrekkes	Ferdig med Order, Request og TFC	28/04/21	Kristian		done



54	4/26/2021	GLN	General	Omregning TFC Die info, °C skal regnes om til F - er ikke implementert, die cost pr kg har enhet NOK/kg, kan vises i EUR/kg om currency er EUR(ingen omregning), eller regnes om og vises i blå for \$/lb om det jobbes i US unit	Lagt til currency i databasen, tempratur omregning er ok, tittelen på die cost er endret til die cost [currency/(kg eller lb)]		Nils		done
55	4/29/2021	GLN	Request	Når en prøver å endre kappelengde profil, eller part length så får en opp varselboks når første siffer skrives inn, denne må vente til enter eller tab eller ny celle er trykket	Fikset med onblur istedet for onchange 30.04: Fungerer ikke på request update som er der jeg har rukket å teste så langt. 01.05: Skal funke nå siden benteler serveren ble oppdatert idag		Jesper		Done
56	4/29/2021	GLN	Request	Surface quality legges til under Profile info	30.04: Lagt inn på order men ikke på request såvidt jeg kan se (GLN)		Nils		in work
57	4/29/2021	GLN	Order	Surface quality tas med i skjema - kan endres ved bestilling	30.04: ok - finnes i skjem		Jesper		Done
58	4/29/2021	GLN	TFC	Run-out length feilmelding selv om lengden er innenfor spec, ref e-post - varsel/feilmelding forsvant om jeg gikk til neste side og så tilbake Run-out length - ingen feilmelding selv om lengden er utenfor spec, ref e-post - varselet kom om jeg gikk til neste side og så tilbake....					not started
59	4/29/2021	GLN	TFC	TFC based on old TFC with new request, endret profilvekt men da blir ikke run-out length endret - den ser ut til å bli endret først etter at jeg har vært innom side 4 og så går tilbake til side 3. Endrer jeg andre verdier på side 3 (bille length eller butt end) så beregnes ny gross length med en gang, men dette gjelder ikke om jeg endrer verdier på sidene før eller etter....					not started
60	4/29/2021	GLN	Users	Ser at vi ikke har holdt tunga rett i munnen ved gjennomgang usecase fra dere tidlig i prosjektet og har derfor ikke riktige rettigheter lagt ut. user: request, order superuser: request, TFC, cost-calc og order Product Controller: Cost-Calc - hente ut bridge fil Admin: se og editere alt, parametre og brukere	ES: kommer med neste oppdatering		Emma		Done
61	4/29/2021	GLN	Users	Cost-Calc user har ikke tilgang til noe, tom meny fane	30.04: fortsatt tomme faner når jeg logger inn som costcalc user ES: kommer med neste oppdatering		Emma		Done
62	4/29/2021	GLN	TFC	Reduction ratio - upreis formulering av meg, warning skal si: This value should be between 15-and...			Emma		not started
63	4/30/2021	GLN	General	Automatisk e-post: når databasen er ferdig testet må vi kunne slå på automatisk e-postvarsler. Vi må sende dere innhold - standardoppsett for disse om dere ikke finner de fra gammel vb rutine tror de står der.. 1) Request utfyllt og sendt --> mail to "requestor", "Profile planner" = hara.extrusion@benteler.com 2) TFC utført og sendt til db --> mail to "requestor", "Product controller" med beskjed om at TFC ferdig og at bridge fil kan hentes 3) Order utfyllt og sendt --> mail to "profile planner"	Skal endre IP til Benteler sin interne SMTP server, og legge til mail på order og TFC. Ønsker gjerne å få tilsendt format på E-post. Automatisk mail etter ferdig TFC og Order er lagt til, men alle sender mail til en fast mail adresse for testing foreløpig for å unngå spam.		Kristian		in work
64	4/30/2021	GLN	General	FAQ side info Bilde til førstesiden / Layout			Kristian Inge	Gerda	not started

## Appendix D

# CostCalc document code

```

1 public static byte[] getBridge(int requestNo, Connection con) {
2     try {
3         List<Costcalc> costcalcs = getCostCalcPerRequest(requestNo,con); //Gets all
4         costcalc relevant data for each TFC per a request number
5
6         //Retrieves the file
7         Path temp = Files.createTempFile("BridgeFile", ".xlsx");
8         Files.copy(CostcalcService.class.getClassLoader().getResourceAsStream("
9         BridgeTemplate.xlsx"), temp, StandardCopyOption.REPLACE_EXISTING);
10        FileInputStream inputStream = new FileInputStream(temp.toFile());
11        Workbook workbook = WorkbookFactory.create(inputStream);
12
13        Sheet sheet = workbook.getSheetAt(0);
14
15        Costcalc costcalc; //Buffer
16
17        //Prepares the area of the Excel document for manipulation
18        initializeSheet(costcalcs,sheet);
19        int offset = 1; //Offset in the excel sheet (Where the first row starts)
20        for(int i = 2; i<=costcalcs.size()+1; i++) {
21            costcalc = costcalcs.get(i-2); //Indexing starts at 0
22
23            String suffix = "";
24            int press = Integer.parseInt(costcalc.getPress().substring(1));
25            if(press == 22 || press == 40 || press == 55)
26                suffix = " (NO)";
27            if(press == 16)
28                suffix = " (FR)";
29            if(press ==27 || press == 35)
30                suffix = " (US)";
31
32            sheet.getRow(0+offset).getCell(i).setCellValue(costcalc.getRequest_no());
33            sheet.getRow(1+offset).getCell(i).setCellValue(costcalc.getSketch());
34            sheet.getRow(2+offset).getCell(i).setCellValue(costcalc.getProfile_no());
35            sheet.getRow(3+offset).getCell(i).setCellValue(costcalc.getTFC_no());
36            sheet.getRow(4+offset).getCell(i).setCellValue(costcalc.getCustomer());
37            sheet.getRow(5+offset).getCell(i).setCellValue(costcalc.getProject());
38            sheet.getRow(6+offset).getCell(i).setCellValue(costcalc.getProduct());
39            sheet.getRow(7+offset).getCell(i).setCellValue(costcalc.getAlloy());
40            sheet.getRow(8+offset).getCell(i).setCellValue(costcalc.getTemper());

```



```

39         sheet.getRow(9+offset).getCell(i).setCellValue(costcalc.
getReq_part_length());
40         sheet.getRow(10+offset).getCell(i).setCellValue(costcalc.
getParts_pr_vehicle());
41         sheet.getRow(11+offset).getCell(i).setCellValue(costcalc.
getAvg_annual_volume());
42         sheet.getRow(13+offset).getCell(i).setCellValue(costcalc.getPress()+
suffix);
43         sheet.getRow(19+offset).getCell(i).setCellValue(costcalc.
getProfile_weight());
44         sheet.getRow(20+offset).getCell(i).setCellValue(costcalc.getBillet_length
());
45         sheet.getRow(21+offset).getCell(i).setCellValue(costcalc.getButt_end());
46         sheet.getRow(22+offset).getCell(i).setCellValue(costcalc.
getDelivered_profile_length());
47         sheet.getRow(23+offset).getCell(i).setCellValue(costcalc.
getExtrusion_speed());
48         sheet.getRow(24+offset).getCell(i).setCellValue(costcalc.
getDie_change_time());
49         sheet.getRow(25+offset).getCell(i).setCellValue(costcalc.
getBatch_size_billets());
50         sheet.getRow(27+offset).getCell(i).setCellValue(costcalc.getScrap_front()
);
51         sheet.getRow(28+offset).getCell(i).setCellValue(costcalc.getScrap_rear()
);
52         sheet.getRow(29+offset).getCell(i).setCellValue(costcalc.
getScrap_first_billet());
53         sheet.getRow(30+offset).getCell(i).setCellValue(costcalc.
getScrap_last_billet());
54         sheet.getRow(31+offset).getCell(i).setCellValue(costcalc.
getNon_performance_scrap());
55         sheet.getRow(33+offset).getCell(i).setCellValue(costcalc.
getSurcharge_tooling_tol());
56         sheet.getRow(35+offset).getCell(i).setCellValue(costcalc.
getLengths_per_layer());
57         sheet.getRow(36+offset).getCell(i).setCellValue(costcalc.
getLayers_per_rack());
58         sheet.getRow(37+offset).getCell(i).setCellValue(costcalc.
getCycletime_packing_rack());
59         sheet.getRow(38+offset).getCell(i).setCellValue(costcalc.
getAdditional_invest_extrspeed());
60         sheet.getRow(39+offset).getCell(i).setCellValue(costcalc.
getInvest_comment());
61
62         sheet.getRow(96+offset).getCell(i).setCellValue(costcalc.getDie_price());
63         sheet.getRow(97+offset).getCell(i).setCellValue(costcalc.getBolster_price
());
64         sheet.getRow(98+offset).getCell(i).setCellValue(costcalc.getNo_cavities()
);
65         sheet.getRow(99+offset).getCell(i).setCellValue(costcalc.getDie_life());
66
67     }
68
69     inputStream.close();
70
71     ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
72     workbook.write(outputStream);
73     workbook.close();
74
75     return outputStream.toByteArray();

```

```

76         //outputStream.close(); //Not needed for a byteArrayStream
77     } catch (FileNotFoundException e) {
78         e.printStackTrace();
79     } catch (IOException e) {
80         e.printStackTrace();
81     }
82     return null;
83 }
84
85
86
87
88 public static void initializeSheet(List<Costcalc> costcalcs, Sheet sheet) {
89     for(int i = 2; i<=costcalcs.size()+2; i++) {
90         for (int j = 0; j <= 101; j++) {
91             if (sheet.getRow(j) == null) {
92                 System.out.println("Creating row");
93                 sheet.createRow(j);
94             }
95             if (sheet.getRow(j).getCell(i) == null) {
96                 System.out.println("Creating cell");
97
98                 sheet.getRow(j).createCell(i);
99             } else {
100                 sheet.getRow(j).getCell(i);
101             }
102         }
103     }
104 }
105
106
107
108
109 public static List<Costcalc> getCostCalcPerRequest(int requestNo, Connection con) {
110     List<Costcalc> costcalcs = new ArrayList<Costcalc>();
111
112     Costcalc buffer = null;
113     try{
114         ResultSet rs; //TFC query
115         ResultSet rs2; //Press query
116         ResultSet rs3; //Request query
117
118         //Getting TFC data
119         PreparedStatement st = (PreparedStatement) con.prepareStatement("Select *
120 FROM TFC WHERE Request_no = ?");
121         st.setInt(1,requestNo);
122         rs = st.executeQuery();
123
124         while (rs.next()) {
125
126             //Getting press data
127             PreparedStatement st2 = (PreparedStatement) con.prepareStatement("SELECT
128 Die_change_time FROM Press WHERE Press_name = ?");
129             st2.setString(1,rs.getString("Press"));
130             rs2 = st2.executeQuery();
131             while (rs2.next()) {
132
133                 //Getting request data
134                 PreparedStatement st3 = (PreparedStatement) con.prepareStatement("
135 SELECT * FROM Request WHERE Request_no = ? AND Request_ver_no = ?");

```

```

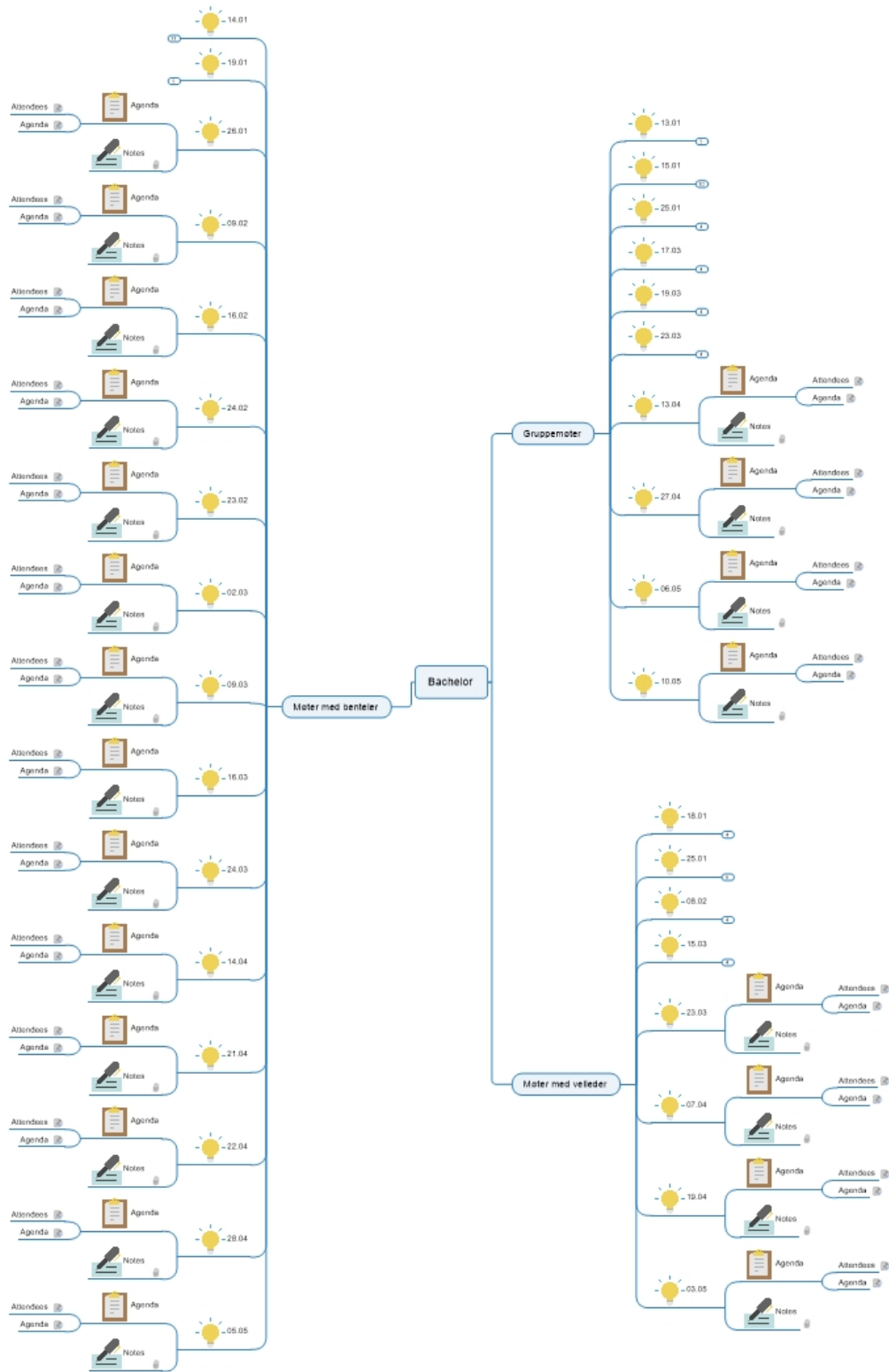
133     st3.setInt(1, rs.getInt("Request_no"));
134     st3.setInt(2, rs.getInt("Request_ver_no"));
135     rs3 = st3.executeQuery();
136     while (rs3.next()) {
137         String sketchRev = "";
138         if(rs3.getString("Sketch_rev") != null)
139             sketchRev = rs3.getString("Sketch_rev");
140
141         buffer = new Costcalc(
142             rs.getInt("Request_no")+"-"+rs.getInt("Request_ver_no"),
143             rs3.getString("Sketch_profileno"),
144             rs.getString("Sketch") + sketchRev,
145             rs.getInt("TFC_no"),
146             rs3.getString("Customer"),
147             rs3.getString("Project_name"),
148             rs3.getString("Part_description"),
149             rs.getString("Alloy"),
150             rs.getString("Temper"),
151             rs3.getFloat("Part_length"),
152             rs3.getInt("Parts_pr_vehicle"),
153             rs3.getInt("Average_volume_y"),
154             rs.getString("Press"),
155             rs.getFloat("Profile_weight_kg_pr_m"),
156             rs.getFloat("Billet_length_mm"),
157             rs.getFloat("Butt_end_mm"),
158             rs.getFloat("Profile_length_mm"),
159             rs.getFloat("Extrusion_speed_m_min"),
160             rs2.getFloat("Die_change_time"), //From inner query
161             rs.getInt("No_of_billets_batch"),
162             rs.getFloat("Front_scrap_m"),
163             rs.getFloat("Rear_scrap_m"),
164             rs.getFloat("First_billet_additional_scrap_m"),
165             rs.getFloat("Last_billet_additional_scrap_m"),
166             rs.getFloat("Non_perf_scrap_pct"),
167             rs.getFloat("Safety_margin"),
168             rs.getInt("Profiles_per_layer"),
169             rs.getInt("No_of_layers"),
170             0, //Is calculated in schema
171             rs.getFloat("Investment_req"),
172             rs.getString("Investment_description"),
173             rs.getInt("Die_cost_kg"),
174             rs.getInt("Bolster_cost"),
175             rs.getInt("No_of_cavities"),
176             rs.getInt("Die_life")
177         );
178     }
179     costcalcs.add(buffer);
180 }
181
182 }catch(SQLException ee){
183     ee.printStackTrace();
184 }
185 return costcalcs;
186 }
187

```

Listing D.1: Cost Calc main function for producing a bridge file with accessory functions

## **Appendix E**

# **Mindmanager map**



## **Appendix F**

# **Project agreement**

Norges teknisk-naturvitenskapelige universitet	
Fakultet for informasjonsteknologi og elektroteknikk	

## Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Benteler Automotive Raufoss AS

(oppdragsgiver), og

Jesper Trøan, NILS OLAV TUV

Kristian Tveiten,

Emma Sofie Søvik

(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 15/1 til 15/6 - 2021.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:

- Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
- Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle beståtte bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om

FP

publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.
6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.
7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.
8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.
9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.
10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.
11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.
12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): \_\_\_\_\_

Oppdragsgivers kontaktperson (navn):

FRODE PAULSEN  
frode.paulsen@benteler.com

Student(er) (signatur):

Jesper Trøan dato 18.01

Nil olav Tur dato 18.01

Kristian Tveiten dato 18.01

Emma S Søvik dato 18.01

FP



Oppdragsgiver (signatur):

Frode Paul

dato

15/1-21

*Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.*

*Godkjennes digitalt av instituttleder/faggruppeleder.*

*Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.*

Plass for evt sign:

Instituttleder/faggruppeleder (signatur):

dato

[Type here]

## **Appendix G**

### **Group rules**

# Gruppregler:

- Alle må delta i møte med veileder\*
- Alle må delta i møte med benteler\*
- Alle må delta i sprint møte (om vi velge scrum)\*
- Etter hvert møte skal alle ha klart for seg hva de skal jobbe med til neste gang.
- Hvis en person ikke har mulighet til å delta på et møte, så må det sies i fra om minst to timer i forveien.
- Ved lengre fravær skal situasjonen diskuteres med veileder.
- Antall timer som er forventet å arbeide i uken? 30, 6 per dag
- Ved uenigheter i gruppen blir det avstemning, om det er likt får gruppeleder ta avgjørelsen.
- Alle må loggføre timene de jobber med prosjektet

## **Appendix H**

# **Meeting minutes**

# Meeting minutes

14.01.2021

**Agenda:** Requirements and desires for the program. Project agreement

**Place:** Microsoft teams

**Duration:** 30 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Frode and Gerda*

**Outcome:**

Discussed what requirements Benteler had for our application, as well as the inner workings of the current application used, which lacked functionality in key places where our application will fill in. This includes the current way the CostCalc team receives process parameters by Excel documents, instead our application should directly send this data using some form of integration with the CostCalc program.

Security was also discussed but no clear opinion was formed, so the next meeting is planned with someone from IT at Benteler. For the next meeting we are also supposed to discuss what developing language is the most fitting, (web or java).

We were also promised a schema showing the current workings of the current program as well as some key requirements for the next meeting on Tuesday.

15.01.2021

**Agenda:** Project rules, routines, tools, Planning the start of next week. Plan things that have to be done

**Place:** Teams

**Duration:** 1 hour

**Participants:** Nils, Jesper, Kristian, Emma

**Outcome:**

Came up with some project rules that the group has to follow. We also found some time slots where the whole group can meet and work together. We came up with some tools that we want to use during this project. Things we have to do next were also discussed during the meeting.

18.01.2021 (11-12) - Supervisor meeting.

**Agenda:** Discuss development language in context of usage (little experience with web vs. alot of java experience), how we should consider Bentelers current program solution, the current received requirements.

**Place:** Teams

**Duration:** 30 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Sony and Hilda*

**Outcome:**

Look at previous courses in web languages

LinQ for SQL to visual basic

Split what we learn of the language, so we all learn different special stuff.

Ask for if they want to extract to a pdf for example? for a report

We discussed the dilemma of client versus web application, and got recommended to take a week (week #3) to learn about web languages, like react javascript etc, and after this week

we should make a decision if we believe we can finish the project using web languages. Looking at previous courses in web development was recommended. It was also recommended that we split non-basic web development research, so we each learn a specialized “field”. We should discuss if the database content will need to be extractable to for example PDF with Benteler. They also mentioned that we can use ASP.net for the web application, and visual studio for the development because visual studio connects both the frontend, backend and the database.

## 19.01.2021 (11:30-12:30) - Meeting with Benteler

**Agenda:** Get a walk through of the flow chart/functionality of the current program solution, and receive concrete functional requirements.

**Place:** Teams

**Duration:** 60 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Frode and Gerda*

**Outcome:**

A complete walkthrough of how Benteler handles requests, TFC, intercommunication with databases, teams and so on. Also got a general idea of privileges that certain user groups should have.

**Detailed meeting notes/use cases/workflows:**

- When a profile request is made, the project planner is notified (today with an email), this must happen.
- Then the internal TFC meeting is called, where the request is evaluated and most of the process parameters is decided/calculated/estimated from the meeting.
- The “admin” section in the current program describes access to certain stuff, today its done in excel
- “OrderTest” is the test database, which we will get access to in form of a copy that we have to set up ourselves.
- Number and index on a request is tied to a schema/drawing tied to a type of alloy, with parameters.
- The program should be able to search for matches based on all parameters (at least those available in the database).
- A user can start a totally new request, or utilize a previous request and its data.
- If a user select “Change” the purpose is that the changes to a existing request is big enough that it requires a new calculation by the CostCalc team, so its sent to them. Significant changes includes change of alloy or running parameters. Gets a new index number after this change.
- If a user selects “ReCalc” only certain parameters can be changed, and the program will automatically recalculate values in the request. The expert team will not have to meet after this. Calculations are based on volume or length.
- Fields that cannot be changed from the copied request are marked with **orange**, fields that HAS TO BE approved are marked with **red**, fields that needs to be filled are marked with **yellow**.
- Indexing/enumeration is done automatically by the program, when a profile/TFC is changed (enough?) it gets a new index.
- A user can also request an old profile again.
- The program should keep static parameters like alloys, presses etc in the database, today it’s kept in excel in a drop down menu.
- Profile drawing number is always in the request if the drawing exists.
- SAPID (the id of the profile drawing) is in the request, we could also include an image of the drawing or a link to the drawing in addition.

- All **purple** fields in the TFC document represent fields where values are automatically retrieved from the request the TFC is based on. These values come from the database, and the rest is filled in during the meeting.
- Some options/choices in the TFC have some linked parameters that's automatically filled in other places in the document like press capacity and so on.
- A user can also start with a previous TFC as a base, and change it.
- Some fields should include formulas and limits, so the user cannot input illegal values that are either too high or too low. The color of a field should also change if values are coming too close to an illegal value. This is done to alarm the TFC team.
- TFCs with illegal values and hence are not producible should not pass.
- There should be functionality to change units used (metric/imperial)
- Language should be English ONLY
- A profile drawing has the request number and version number attached to it.
- A request can have multiple TFCs, a TFC for a small press and a TFC for a large press.
- When a user presses update after finishing the change of the TFC/Request it is sent to the database.
- A user should be able to search for requests/TFCs with for example a cyclus time of more than X or less than Y or both. Which has a minimum wall thickness and number of chambers.
- A user should be able to delete requests/TFCs
- When a TFC is finished it is sent to the CostCalc team using a button, (today the most important values is sent with a excel document.)
- The CostCalc team has access to the BridgeCalc report which is sent.
- For a order to be made, it requires that something has been requested first, then a order is made based on that request.
- When a order is made, the request, TFC and what was ordered is sent to "base"?
- RND orders doesn't require a previous project/tfc/request, it is a open order.
- The program should provide functionality to adjust database data like static values and other values.(insert/delete/update etc).

#### **Permissions:**

- Reports of which requests that are made are open to everyone
- TFC evaluation is not open for everyone (certain people)
- Full report is really limited with all key profiles.
- Permissions could be ordered in such a way that the leftmost column in their existing program belongs to user group 1, and middle user group 2, and right user group 3.

## **25.01.2021 (09-10) - Internal Meeting**

**Agenda:** Status meeting

**Place:** Teams

**Duration:** 60 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils*

**Outcome:**

Discussed our progress in learning front- and backend languages/tools.

## 25.01.2021 (11:00-11:30) - Supervisor meeting.

**Agenda:** Discuss how comfortable we have become in terms of the web programming languages. Discuss the Use case diagram, and requirements.

**Place:** Teams

**Duration:** 30 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Sony and Hilda*

**Outcome:**

Discussed what can be added to the project plan, and that the plan looked good. Also discussed that we need to collect everything our application does better than the old application so that the sensors see that we not only have copied the old application.

## 26.01.2021 (11:30-12:30) - Meeting with Benteler

**Agenda:** Present use cases, who creates the accounts?, do you still need the testing functionality from the old software, filename format: [request number]:[TFC number]

**Place:** Teams

**Duration:** 60 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Frode and Gerda*

**Outcome:**

We got a run through of the use cases, as well as comments and improvements on what we made. Admin will be creating the accounts with necessary permissions as it is the safest. We will also attend a meeting where we will learn about the extrusion process to get some background knowledge when developing the application. We also discussed how we could improve the current methods. Like letting the program evaluate if a Recalc, Update etc is required. We also got approval of the idea of filling out the TFC section by section, but then we need to have certain parameters visible at all times (like extrusion length).

TFCs should still be sent to the database even though any values are outside given thresholds, as an expert needs to assess this and approve it for production before it can be sent as an order.

There was also discussed that they would want a report of multiple TFCs with only selected values (which is done in our application). This report should not be changed so it could be in PDF format.

Discussed also whether mail or any task program would be the best choice for notifying the project planner.

The threshold values for deciding whether a profile is legal or not should be in the database. Normal users can see whole requests. Also discussed that normal users should only be able to see requests and not delete. They also want us to find a suited format for printing of request reports, and also be able to choose which parameters to print out on the report.

## 08.02.2021 - Supervisor meeting.

**Agenda:** Discuss NDA, citing code usage

**Place:** Teams

**Duration:** 30 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Sony and Hilda*

**Outcome:**

Discussed the NDA and that we could sign it and send a mail with it describing what we need to show (?). Also discussed that the NDA was a quite standard NDA so it shouldn't be any problem, but Sony is going to check with Tom. Got input from the supervisors to convey in the report and presentation that we are working on the whole process from frontend to database as well as planning testing and input from external people. The A grade is in the



fine details! We should also research if there is something already existing that could be used for the purpose, and show it to Benteler to get a response if it could solve the case, or if it is lacking and what our application should be doing differently. (*I doubt it though as the case is very specific -Kristian*).

## 09.02.2021 (11:30-11:45) - Meeting with Benteler

**Agenda:** Discuss the NDA. Also discuss if the application can be tested during a period before the deployment phase.

**Place:** Teams

**Duration:** 15 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Frode and Gerda*

**Outcome:**

Discussed the NDA, and agreed on that we should have a meeting closer towards the end of the project to make sure we are not disclosing unwarranted information in the report or during the presentation. Unwarranted information is mostly production data regarding certain profiles for example. The programs workings, Request/TFC input schemas and such can be shared. Snippets of the code can be used in the report/presentation for demonstration purposes. A period of testing was also agreed upon further towards the end of the project.

## 16.02.2021 (11:30-12:10) - Meeting with Benteler

**Agenda:** Discuss the NDA. Also discuss if the application can be tested during a period before the deployment phase.

**Place:** Teams

**Duration:** 40 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils, Frode, Geir and Gerda*

**Outcome:**

Showed the progress, and got some feedback; in the search interface, only show the last requests per the last six months, and the search interface should be flexible to adjust the shown columns to a personal preference, perhaps create personal views.

Titles in the search interface should be translated to human readable text.

Export of selected parameters from the request into a PDF document.

Explore the possibility of manipulating columns in the database without needing to change the code/SQL Queries. Also discussed access to a webserver at Benteler. They are going to send us some papers that we have to sign, and then we will get the server.

## 23.02.2021 (12:30-13:00) - Meeting with Benteler

**Agenda:** Walkthrough for the cost-calc process

**Place:** Teams

**Duration:** 30 minutes

**Participants:**

*Kristian, Jesper, Nils, Frode and Markus*

**Outcome:**

We got a walkthrough by Markus from the CostCalc team on the process of calculating from TFC to order, where he explained that the calculation is done in multiple steps by multiple people possibly in different locations. Discussed how we could accomplish this in our program, that we could either:

1. Export a excel documents with required information
2. Export to database (SQL)
3. Integrate the whole process, by providing access for different people to calculate/input different values at each step in the costcalc process.

We also received the costcalc schema by mail.

Contact: [Markus.werz@benteler.com](mailto:Markus.werz@benteler.com) for further questions to CostCalc

## 24.02.2021 (14:00-15:20) - Meeting with Benteler

**Agenda:** Detailed walkthrough of the extrusion process

**Place:** Teams

**Duration:** 1 hour and 20 minutes

**Participants:**

*Kristian, Jesper, Nils and Gerda*

**Outcome:**

We got a detailed walkthrough of the extrusion process and explanation of the different process parameters used in the TFC schema. For example how you get metal scrap out of the extrusion process, what happens when a new bolt is inserted back to back with the old one. Temperature and speed limits for keeping certain properties of the metal and different types of alloys and how that affects the strength, extrusion speed and crushability.

We also got input to our request schema that we should add "Due date", "Extrusion Length/Delivery Length", and we could also combine multiple steps in the TFC/Request Scheme like TFC : (6,7,8) and format/group more fields together (we will get input on this from Gerda). In order main page the user should be able to search for request and TFC number to input in the order schema.

## 02.03.2021 (11:30-12:21) - Meeting with Benteler

**Agenda:**Show what we have done in this sprint. Also discuss if they want to export to excel or pdf or both. We should also ask for which parameters that can change in change/recalc etc. Also ask about the parameter intervals.

**Place:** Teams

**Duration:** 51 min

**Participants:**

*Kristian, Jesper, Emma, Nils and Gerda*

**Outcome:**

Showed development progress, got input on certain menu options to be improved as no request can be changed without recalculation. TFC also cannot be created from scratch, must be based on at least an earlier request. Choosing an old TFC and then picking the desired request afterwards was also discussed for implementation. Parameters and limits for certain fields were explained, as well as how certain values are calculated using for example press parameters.

Admin should also be able to mark a request as invalid, instead of deleting which creates gaps. Our program should also account for "Die Change Time" in addition to "Dead Cycle Time" when calculating time usage. Press parameters were also desired to be translated into something more readable. When the user gets close or beyond limits in a field like "Reduction ratio" a confirm dialog could pop up to make the user confirm the input. But other values like "Gross extr length" could only be marked with red. (Will get further input on this later). Agreed on only "translating" the values to imperial if that option is chosen, meanwhile all saved data is metric.

Only a set of parameters should be visible to choose from in the search pages, input on this will come later from Gerda. We also got input that only the admin user should have an option

to “update request” on the request mainpage. This is for if normal users fill some wrong parameters.

## 09.03.2021 (11:30-12:12) - Meeting with Benteler

**Agenda:** Some questions about parameters and order-setup

**Place:** Teams

**Duration:** 42 min

**Participants:** Nils, Kristian, Jesper, Emma, Frode, Gerda and Kristian Inge

**Outcome:** Got input on the order mainpage that they want to search from requests and get the option to choose an tfc number based on that requestnumber they choose. Kristian Inge also came up with a suggestion to change the TFC forms so that they always see the input they have entered in the previous steps. The students are going to do some research and try to come up with a solution to this. The duplicated fields "Alloy" and "Temper" in the TFC form were discussed, input on this will come later. The dropdowns for alloy and temper should be sorted by the most used, but input on which is the most used is needed for this to happen. We also discussed testing. Benteler wants to be able to test the program's functionality early on. The students will discuss internally and give Benteler a response of when we believe it will be ready for testing.

From week 12 and onwards, meeting time will be Wednesday 11:30 to better suit all participants schedules

## 15.03.2021 - Supervisor meeting.

**Agenda:** Showing progress

**Place:** Teams

**Duration:** 40 minutes

**Participants:**

*Kristian, Jesper, Emma, Nils and Hilda*

**Outcome:**

Showed progress and discussed more about the wow factor. We need to finish main functionality and get it working correctly before figuring out the wow factor. Having a per project report was also discussed but it is not that applicable to the cause.

In the report we should also provide screenshots and comparisons to the old software and how we exceed that in which ways. If images are too large we could put them in the appendix last in the report.

We also discussed when testing could be done, and that we could request Benteler to give feedback on how our program helped their workflow in certain areas/sections in the program.

## 16.03.2021 - Benteler meeting

**Agenda:** Meeting end of sprint, showing the latest progress

**Place:** Teams

**Duration:** 28 minutes

**Participants:** Nils, Kristian, Jesper, Emma, Frode, Gerda and Kristian Inge

**Outcome:**

Everything in the TFC form is required besides packing, investments and comments, which can be left optional for now. We will be testing the TFCForm with all formulas, but if that becomes slow the amount of fields could possibly be cut down in the final schema. Having a dropdown on the fields “Email Product Controller” and “Email Requestor” should be added to

save time and ensure integrity. Apart from that they were pleased with how things are progressing.

## 17.03.2021 - Internal meeting

**Agenda:** Meeting end of sprint, showing the latest progress

**Place:** Teams

**Duration:** minutes

**Participants:** Nils, Kristian, Jesper, Emma

**Outcome:**

Discussed colors on backgrounds in forms, red color in invalid request is a bit too strong, disabled requests should not be viewable in request-selection popups, but viewable in main search page with the same red color.

Speak with Steven from the software security course - meeting next week or after easter vacation.

We agreed on backlog items/tasks. This sprint is mainly focused on security and design, with further improvements on the existing functionality/forms/pages etc.

## 19.03.2021 - Internal meeting

**Agenda:** Showing the latest progress

**Place:** Teams

**Duration:** minutes

**Participants:** Nils, Kristian, Jesper, Emma, Frode

**Outcome:** Discussed process. Came up with a solution to make the tfc form faster by making a own form that will be runned when US metrics are chosen. Also agreed to that everyone needs to research more on deployment before the next meeting.

## 23.03.2021 - Supervisor meeting

**Agenda:** Get information about deploying a website on a webserver like IIS

**Place:** Teams

**Duration:** 20 min

**Participants:** Kristian, Nils, Emma, Jesper, Sony and Doney

**Outcome:**

<https://stackoverflow.com/questions/41519198/how-can-i-host-a-spring-project-in-windows-iis-using-tomcat>

<https://www.youtube.com/watch?v=yVKiNAkhav8>

<https://www.codejava.net/servers/tomcat/how-to-deploy-a-java-web-application-on-tomcat>

<https://www.guru99.com/deploying-website-iis.html>

Tomcat can be used with IIS, by connecting in through that. HTTPS was a good idea for traffic encryption.

## 23.03.2021 - Internal meeting

**Agenda:** Showing the latest progress

**Place:** Teams

**Duration:** 40 minutes

**Participants:** Nils, Kristian, Jesper, Emma

**Outcome:** Discussed process.

## 24.03.2021 - Meeting with Benteler

**Agenda:** Discussing US/Metric conversion, show and discuss the overview over previous filled in parameters in TFC.

**Place:** Teams

**Duration:** 40 minutes

**Participants:** Nils, Kristian, Jesper, Emma, Frode, Gerda, Kristian inge.

**Outcome:** Got feedback that we don't have to prioritize implementing saving forms locally because internet shutdown is very rarely to happen. Showed also the latest process. We got feedback that they had envisioned a bit different solution of the summary in TFC, but they were willing to test it out first. Benteler also asked what would happen if two people were working with the same request/index and we agreed to implement a solution where the index number would change and notify the user on send, if someone already has sent in the same request/index combo.

## 07.04.2021 - Meeting with Steven (Security)

**Agenda:** Get input on what security and methods is appropriate for our application, discuss HTTPS/Authentication/Frameworks

**Place:** Teams

**Duration:** 35 minutes

**Participants:** Nils, Kristian, Jesper, Emma and Steven

**Outcome:**

We need to identify the most important and critical functions/parts of the application and implement relevant security or methods to protect. A important thing is to use the spring framework and surrounding modules as much as possible, like spring boot security to do authentication (we need to see if this is necessary).

Input validation is also a thing we need to assess, should perhaps be done in frontend, like email validation. For input validation we should use existing APIs or libraries. Numerical validation is already done by using material ui fields.

We should identify certain Owasp practices which is relevant and important for our application.

Concrete security criteria we try to assess/protect with security practices should be properly listed in the report, and the countermeasures should be compared to the OWASP Application Security Verification Standard 4.0.2.

## 13.04.2021 - Internal meeting

**Agenda:** Sprint meeting

**Place:** Teams

**Duration:** 1 hour and 10 minutes

**Participants:** Nils, Kristian, Jesper, Emma

**Outcome:** Discussed what has to be done in the next sprint.

## 14.04.2021 - Benteler meeting

**Agenda:** Sprint progress meeting

**Place:** Teams

**Duration:** 25 minutes

**Participants:** Kristian, Jesper, Emma, Kristian Inge, Gerda and Frode

**Outcome:** Showed progress and discussed when testing will be available. Requested to get feedback from testing which we will add in the report. Also requested feedback on how our application differ from the old solution. We also got feedback on the cost calc page, that exporting relevant costcalc data to Excel file format is sufficient for now. Content on the FAQ (Frequently Asked Questions) page will be evaluated to see what should be there.

## 19.04.2021 - Supervisor meeting about report

**Agenda:** Get feedback on the report

**Place:** Teams

**Duration:** 27 min

**Participants:** Kristian, Nils, Emma, Jesper, Sony and Hilda

**Outcome:**

- Text heavy in tech but short in introduction, balance it
- Put big/complex code stuff in the appendix, but smaller snippets or important snippets for illustrating a point directly in the report
- Use appendix for non-important content
- Have a diagram describing sprints and how it affects different part of the system
- Dont have chapter for terminology. Just normal section (Use star?)
- Dont mix group challenges and project challenges, have a separate section about group limitation, team members. Then have another section with project part.
- Add more diagrams above the text to easier explain certain concepts. Should add a bigger diagram explaining the system as a whole.
- Add our motivation behind the project and how we wanted to create something far superior to the old solution.
- Make a list with with how the old system was bad, and compare how our was much better.
- Inn the first chapter we should sell our project properly, also sell how we did security in comparison to the old solution

## 21.04.2021 - Benteler meeting

**Agenda:** Walk through the testing result feedback list

**Place:** Teams

**Duration:** 60 minutes

**Participants:** Kristian, Nils, Emma, Jesper, Kristian-Inge, Gerda and Frode

**Outcome:**

- On requestor name and email the username can be used to fill in both if a new empty request is created.
- Product Controller Email can have a modifiable list in the database page to modify the dropdown list.
- Alloy should be sorted like the dropdown in the old program.
- FAQ Questions can be formulated by Benteler.
- When filling out the TFC form, the field "Last billet additional scrap" should be able to be overridden.

- Side-by-side is made the default value for “Cavities stacked”, and lock this field if No of cavities is 1.
- Switch between units: Agreed to a small tooltip in each box with the conversion. Textsize h2 and bentler blue and light blue colors.
- If send to database fails, the user should not be sent to the menu and should have the option to download the data to a pdf. Option to print to pdf should be added to all confirm pages.
- Possibility to change order of search table column-order for users will not be implemented, but Benteler can send us changes they would like and we will change them.
- Highlighting of some specific fields should be done with a yellow border and only on the tfc confirmpage.
- Gerda will plan a meeting to discuss costcalc in the near future.

## 22.04.2021 - Benteler meeting

**Agenda:** Cost Calc discussion

**Place:** Teams

**Duration:** 42 minutes

**Participants:** Kristian, Nils, Emma, Jesper and Gerda

**Outcome:**

- Direct import of data into the existing CostCalc excel document would be the optimal solution, but if there are time constraints having the data be exported into an empty excel sheet can be a plan B solution. Macros can then be used to extract to the proper excel sheet.
- CostCalc relevant data should be exported for all TFCs per a request number (and version numbers).
- Research and testing on how to properly import the live database is going to be done to ensure that the data integrity is kept when the application goes live.

## 27.04.2021 - Internal meeting

**Agenda:** Go through the LOP action item list

**Place:** Teams

**Duration:** 1 hour and 30 minutes

**Participants:** Nils, Kristian, Jesper, Emma

**Outcome:** Went through the list with bugs and distributed the last items on the list. Planned a new meeting on thursday.

## 28.04.2021 - Benteler meeting

**Agenda:** Status meeting

**Place:** Teams

**Duration:** 1 hour

**Participants:** Kristian, Nils, Emma, Jesper, Frode and Gerda

**Outcome:**

- Comma doesn't work on some textfields in the application on the benteler servers. Have to test and find out why.
- We need to add Surface quality in request and orderform. Surface quality cannot be changed in TFC but is open to be changed in request and Order.
- Benteler is going to send us an CostCalc excel document with no password protection.

- Benteler also wanted the order page to be available for normal users.

## 03.05.2021 - Supervisor meeting about report

**Agenda:** Get feedback on the report

**Place:** Teams

**Duration:** 25 min

**Participants:** Kristian, Nils, Jesper and Sony

**Outcome:**

- Got feedback that it is smart to prepare a survey and send to benteler so we can get feedback about how satisfied they are. (Measure the quality of the application, ask if they feel the security is good, how easy is it to locate and use)
- Put LOP action items list in the appendix.
- Add a discussion part, where we discuss how things could be improved (like points with low rating from the survey).
- An idea is to make a screenrecord for the application that we can show during the presentation.
- Look at old bentelerreports, to see if theres anything we have missed.
- Important to cite all figures in the report.
- Also important to add at least one comment in the code listings (So the examiners see that we follow general coding standards).
- The survey should contain about 10 different questions, look after a standardized questionnaire for applications.
- We should also send a draft 15th of May or earlier for evaluation.
- In the following text after a figure, it should explain sufficiently so that the examiner does not have to look at the figure. (This should be fixed @ figure 6)
- The presentation can be a reflected version of the report or go in more detail, there should be slides where we have answers to anticipated questions (like security)

## 05.05.2021 - Benteler meeting

**Agenda:** Status meeting

**Place:** Teams

**Duration:** 10 minutes

**Participants:** Kristian, Nils, Emma, Jesper, and Gerda

**Outcome:**

- Agreed that they will answer our survey, with at least 10 people.

## 06.05.2021 - Internal meeting

**Agenda:** report discussion

**Place:** Teams

**Duration:** 30 minutes

**Participants:** Emma, Nils, Jesper and Kristian

**Outcome:**

- Discussed what should be focused on in the report during the next few days.

## 10.05.2021 - Internal meeting

**Agenda:** report discussion

**Place:** Teams



**Duration:** 2 hours

**Participants:** Emma, Nils, Jesper and Kristian

**Outcome:**

- Went through the project and discussed what we have to prioritize this week.
- Agreed to have a new meeting on wednesday.

## 11.05.2021 - Supervisor meeting

**Agenda:** report discussion

**Place:** Teams

**Duration:** 50 minutes

**Participants:** Emma, Nils, Jesper and Kristian, hilda, sony

**Outcome:**

- fix spacing (tabulating)
- Fix upper and lower case letters in listings
- Always add text before a figure in chapters.
- Figures always show up after the reference.
- Terms and abbreviations instead of dictionary.
- Change document type to report, and Use report -> section -> subsection
- Make a table/figure in technologies for pros and cons about different technologies.
- Abstract:
- Brag a bit, Dont use adjectives. Mention everything that we have done. Mention that we have learned a lot of platforms. Everything we have learned. What we adressed.
- Add why we chosed the testing methods.
- Add that we tested on different browsers and OS.
- Find which tests are most common to use in web applications.
- Change discussion to conclusion.
- Center references and appendix titles.
- Write about VPN and why it remedied some security concerns
- In the end of conclusion, have "Concluding Remarks" as a ending (where everything is tied together), learning outcome section can be used here.

## **Appendix I**

# **Full project time usage report**

Week #2	Kristian	Jesper	Emma	Nils
Project Plan Work	72	240	250	?
Internal Meeting	130	130	130	130
Supervisor Meeting	60	60	60	60
Meeting with Benteler	30	30	30	30
Research			220	
Other			120	
<b>SUM:</b>	<b>4,87</b>	<b>7,67</b>	<b>13,50</b>	<b>3,67</b>
Week #3	Kristian	Jesper	Emma	Nils
Project Plan Work	667	1016	867	870
Internal Meeting	60	0	0	0
Supervisor Meeting	30	0	30	30
Report Work	217	120	252	0
Meeting with Benteler	60	60	60	0
Research	255	370	861	701
Other	60	0	0	0
<b>SUM:</b>	<b>22,48</b>	<b>26,10</b>	<b>34,50</b>	<b>26,68</b>
Week #4	Kristian	Jesper	Emma	Nils
Development	328	0	0	0
Testing	0	0	0	0
Report Work	87	0	0	0
Project Plan Work	0	0	0	0
Research	1108	1659	1666	1542
Other	234	90	15	0
Internal Meeting	60	60	60	60
Supervisor Meeting	20	20	20	20
Meeting with Benteler	45	45	45	45
<b>SUM:</b>	<b>31,37</b>	<b>31,23</b>	<b>30,1</b>	<b>27,78</b>
Week #5	Kristian	Jesper	Emma	Nils
Development	786	938	1029	567
Testing	0	0	0	0
Report Work	112	0	634	88
Research	0	356	183	711
Other	0	0	0	0
Internal Meeting	341	145	319	309
Supervisor Meeting	0	0	0	0
Meeting with Benteler	0	0	0	0
<b>SUM:</b>	<b>20,7</b>	<b>24,0</b>	<b>36,1</b>	<b>27,9</b>
Week #6	Kristian	Jesper	Emma	Nils
Development	812	1399	1269	1610
Testing	0	0	0	0
Report Work	143	90	185	74
Research	529	120	155	0
Other	521	0	0	0
Internal Meeting	0	275	227	113
Supervisor Meeting	30	30	90	0
Meeting with Benteler	13	15	14	14
<b>SUM:</b>	<b>34,1</b>	<b>32,2</b>	<b>32,3</b>	<b>30,2</b>
Week #7	Kristian	Jesper	Emma	Nils
Development	1351	434	1597	1350
Testing	0	0	0	0
Report Work	0	60	132	0
Research	0	797	70	0
Other	27	0	91	0
Internal Meeting	131	60	54	54
Supervisor Meeting	0	0	0	0
Meeting with Benteler	40	40	40	40
<b>SUM:</b>	<b>25,8</b>	<b>23,2</b>	<b>33,1</b>	<b>24,1</b>
Week #8	Kristian	Jesper	Emma	Nils
Development	1826	1361	976	1752

Testing	0	0	0	0
Report Work	95	30	0	0
Research	0	165	0	0
Other	0	0	76	0
Internal Meeting	0	0	0	0
Supervisor Meeting	0	0	0	0
Meeting with Benteler	115	115	0	129
<b>SUM:</b>	<b>33,9</b>	<b>27,9</b>	<b>17,5</b>	<b>31,4</b>

Week #9	Kristian	Jesper	Emma	Nils
Development	1794	1173	1633	1512
Testing	0	40	0	0
Report Work	0	0	94	0
Research	0	185	0	44
Other	44	29	186	0
Internal Meeting	148	150	151	143
Supervisor Meeting	18	20	22	20
Meeting with Benteler	51	55	61	60
<b>SUM:</b>	<b>34,3</b>	<b>27,5</b>	<b>35,8</b>	<b>29,7</b>

Week #10	Kristian	Jesper	Emma	Nils
Development	1673	1530	1776	1658
Testing	0	0	0	0
Report Work	116	114	35	0
Research	78	0	0	0
Other	13	0	0	0
Internal Meeting	66	70	69	75
Supervisor Meeting	0	0	0	0
Meeting with Benteler	45	45	43	0
<b>SUM:</b>	<b>33,2</b>	<b>29,3</b>	<b>32,1</b>	<b>28,9</b>

Week #11	Kristian	Jesper	Emma	Nils
Development	1367	1717	1402	1559
Testing	0	0	0	0
Report Work	0	0	0	0
Research	38	0	196	0
Other	23	60	225	0
Internal Meeting	217	225	214	210
Supervisor Meeting	40	40	45	38
Meeting with Benteler	30	30	34	32
<b>SUM:</b>	<b>28,6</b>	<b>34,5</b>	<b>35,3</b>	<b>30,7</b>

Week #12	Kristian	Jesper	Emma	Nils
Development	845	1411	1748	1280
Testing	0	0	0	0
Report Work	0	0	0	0
Research	0	0	0	0
Other	31	111	0	0
Internal Meeting	55	0	0	43
Supervisor Meeting	15	0	13	23
Meeting with Benteler	30	45	47	37
<b>SUM:</b>	<b>16,3</b>	<b>26,1</b>	<b>30,1</b>	<b>23,1</b>

Week #13	Kristian	Jesper	Emma	Nils
Development	130	347	75	224
Testing	0	0	0	0
Report Work	0	104	90	0
Research	0	0	0	102
Other	0	0	0	0
Internal Meeting	0	0	0	0
Supervisor Meeting	0	0	0	0
Meeting with Benteler	0	0	0	0
<b>SUM:</b>	<b>2,2</b>	<b>7,5</b>	<b>2,8</b>	<b>5,4</b>

Week #14	Kristian	Jesper	Emma	Nils
Development	1080	1335	753	1470
Testing	0	0	0	0

Report Work	53	0	0	0
Research	219	119	0	0
Other	38	0	691	0
Internal Meeting	0	0	0	0
Supervisor Meeting	35	0	0	34
Meeting with Benteler	0	0	0	0
<b>SUM:</b>	<b>23,8</b>	<b>24,2</b>	<b>24,1</b>	<b>25,1</b>

Week #15	Kristian	Jesper	Emma	Nils
Development	1313	1163	530	1542
Testing	0	0	60	0
Report Work	515	470	878	19
Research	0	0	0	103
Other	18	48	219	0
Internal Meeting	67	75	81	64
Supervisor Meeting	15	15	13	16
Meeting with Benteler	0	35	31	0
<b>SUM:</b>	<b>32,1</b>	<b>30,1</b>	<b>30,2</b>	<b>29,1</b>

Week #16	Kristian	Jesper	Emma	Nils
Development	958	1712	1007	1448
Testing	0	0	158	0
Report Work	146	43	153	193
Research	0	0	32	0
Other	355	0	171	0
Internal Meeting	128	142	140	126
Supervisor Meeting	27	30	33	71
Meeting with Benteler	102	102	103	67
<b>SUM:</b>	<b>28,6</b>	<b>33,8</b>	<b>30,0</b>	<b>31,8</b>

Week #17	Kristian	Jesper	Emma	Nils
Development	1507	1094	950	1709
Testing	0	0	0	0
Report Work	0	65	0	0
Research	0	0	0	0
Other	323	612	819	0
Internal Meeting	56	208	242	110
Supervisor Meeting	0	0	0	0
Meeting with Benteler	67	60	63	63
<b>SUM:</b>	<b>32,6</b>	<b>34,0</b>	<b>34,6</b>	<b>31,4</b>

Week #18	Kristian	Jesper	Emma	Nils
Development	374	0	0	0
Testing	0	82	0	0
Report Work	1430	1805	2018	1678
Research	0	0	0	79
Other	15	0	0	0
Internal Meeting	0	0	81	49
Supervisor Meeting	25	30	0	24
Meeting with Benteler	0	10	0	0
<b>SUM:</b>	<b>30,7</b>	<b>32,1</b>	<b>35,0</b>	<b>30,5</b>

Week #19	Kristian	Jesper	Emma	Nils
Development	0	0	0	0
Testing	0	0	0	0
Report Work	1617	1881	1763	1499
Research	0	0	0	0
Other	0	0	0	0
Internal Meeting	289	227	138	300
Supervisor Meeting	49	0	51	52
Meeting with Benteler	0	0	0	0
<b>SUM:</b>	<b>32,6</b>	<b>35,1</b>	<b>32,5</b>	<b>30,9</b>

TOTAL MINUTES

Kristian	Jesper	Emma	Nils
28083	29194	31164	28075

TOTAL MIN	TOTAL HRS
116516	1941,93

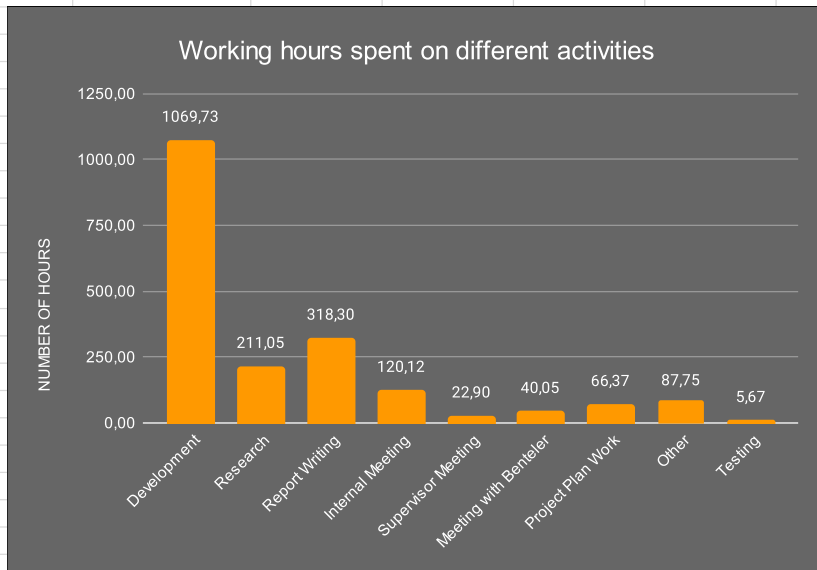
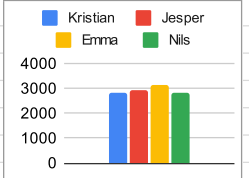
TOTAL MINUTES PER ACTIVITY

	Kristian	Jesper	Emma	Nils
Development	16144	15614	14745	17681
Research	2227	3771	3383	3282
Report Writing	4531	4782	6234	3551
Internal Meeting	1748	1767	1906	1786
Supervisor Meeting	364	245	377	388
Meeting with Benteler	628	687	571	517
Project Plan Work	739	1256	1117	870
Other	1702	950	2613	0
Testing	0	122	218	0

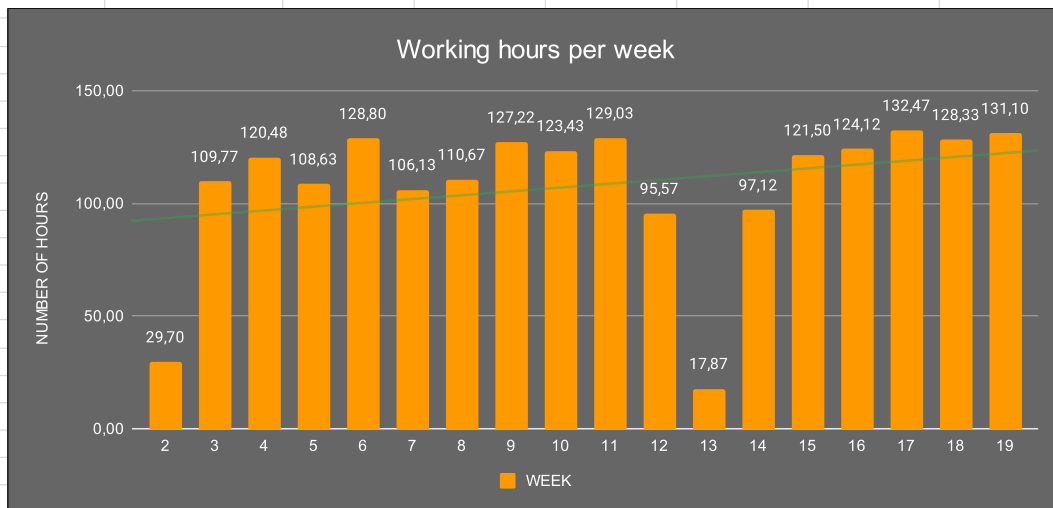
TOTAL MIN	TOTAL HRS
64184	1069,73
12663	211,05
19098	318,30
7207	120,12
1374	22,90
2403	40,05
3982	66,37
5265	87,75
340	5,67

Checksum: 1941,93

Note:  
Document manager (Kristian) should fill in time usage each Monday from Togg!



TOTAL HOURS PER WEEK	
2	29,70
3	109,77
4	120,48
5	108,6
6	128,8
7	106,1
8	110,7
9	127,2
10	123,4
11	129,0
12	95,6
13	17,9
14	97,1
15	121,5
16	124,1
17	132,5
18	128,3
19	131,1



## **Appendix J**

# **Feedback from Benteler**

# Benteler feedback

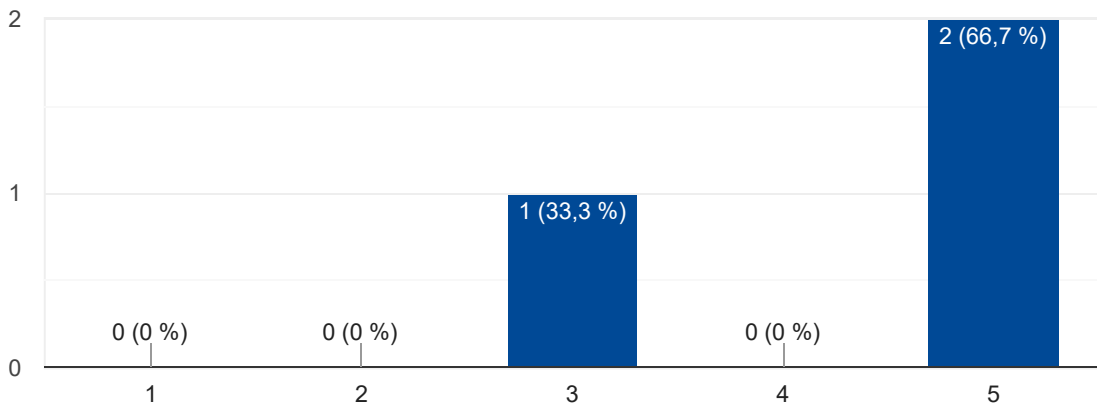
3 svar

[Publiser analytics](#)

Considering your complete experience with our software, how likely would you be to recommend replacing it with the old software?

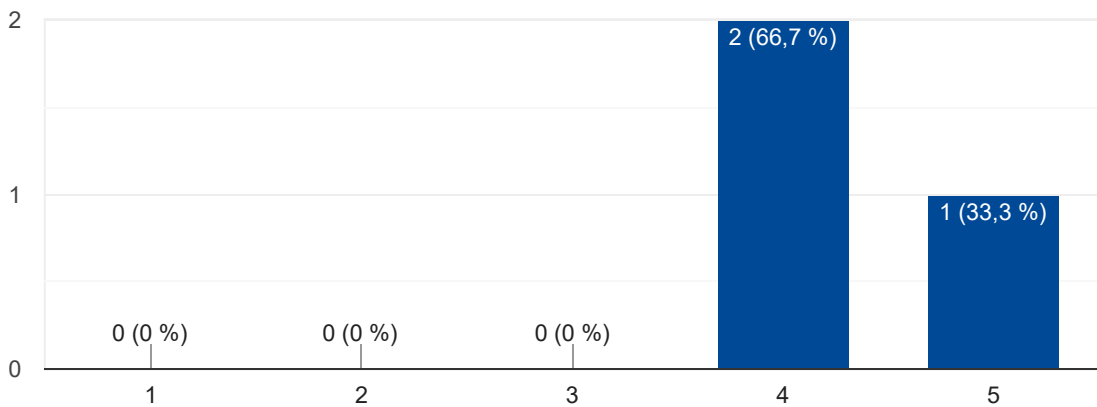


3 svar



In your opinion, where does the new application compare in relation to the old application?

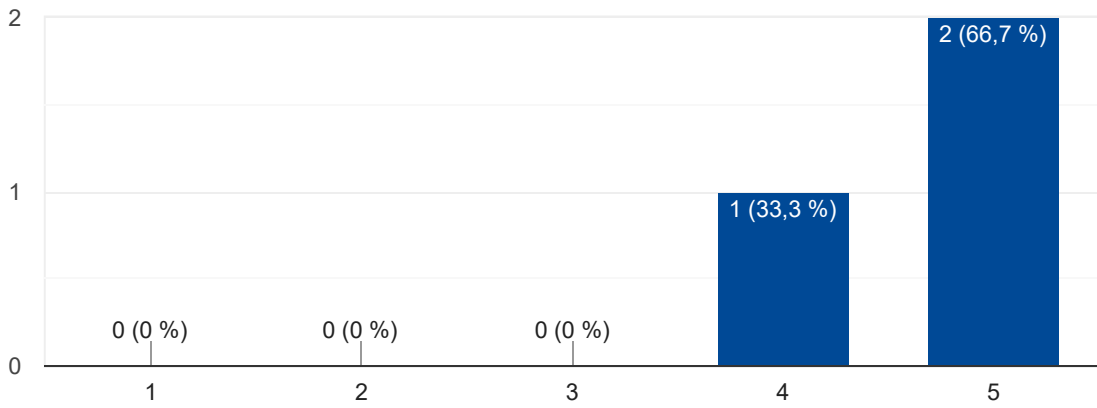
3 svar





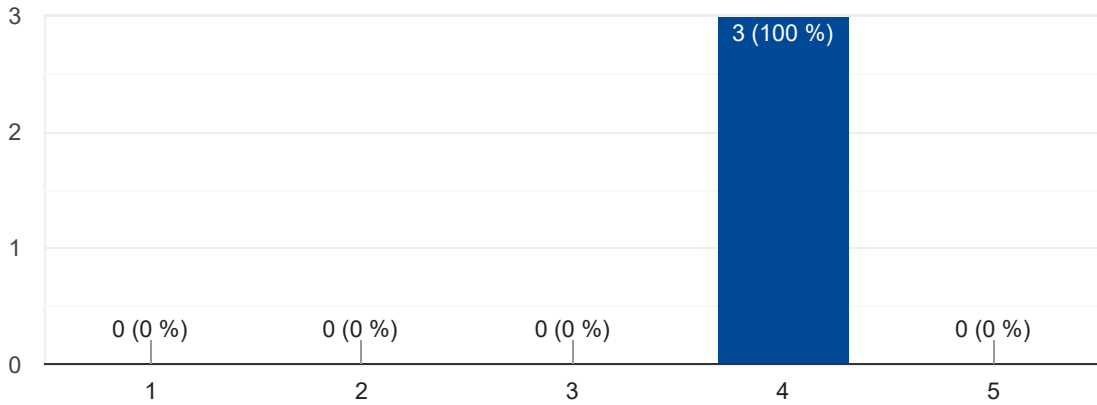
How easy is it to navigate the software? (Are there any areas that cause confusion etc)

3 svar



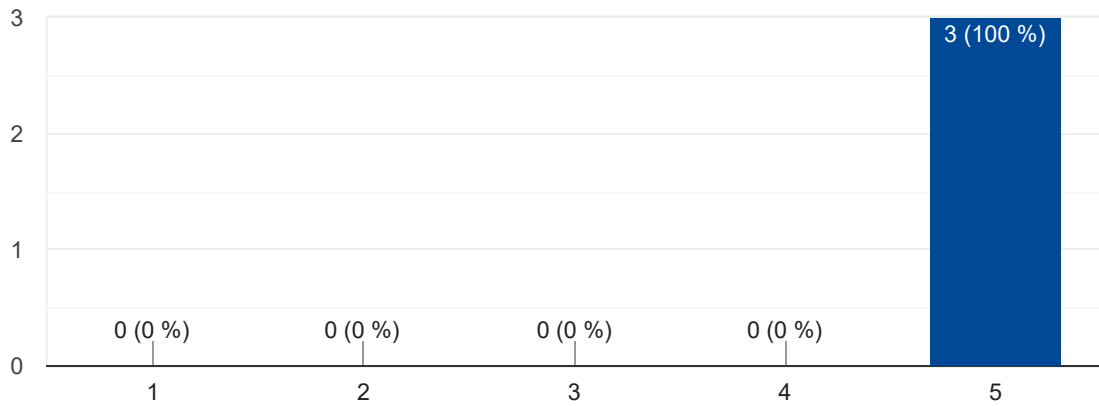
How secure do you feel the application is?

3 svar



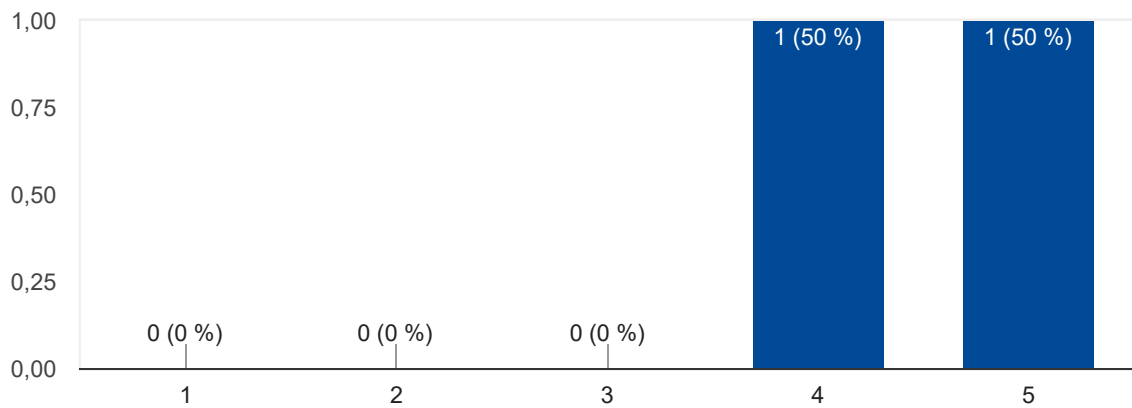
How satisfied are you with the new search functionality and having the ability to search within multiple parameters?

3 svar



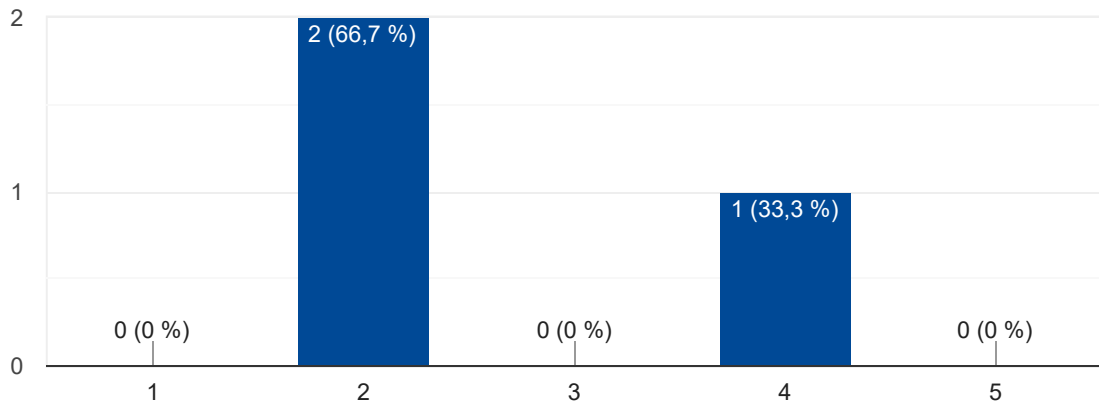
How easy do you feel the database page is in use compared to your old solution? (Add, update, delete parameters, presses etc. Only available for admin.)

2 svar



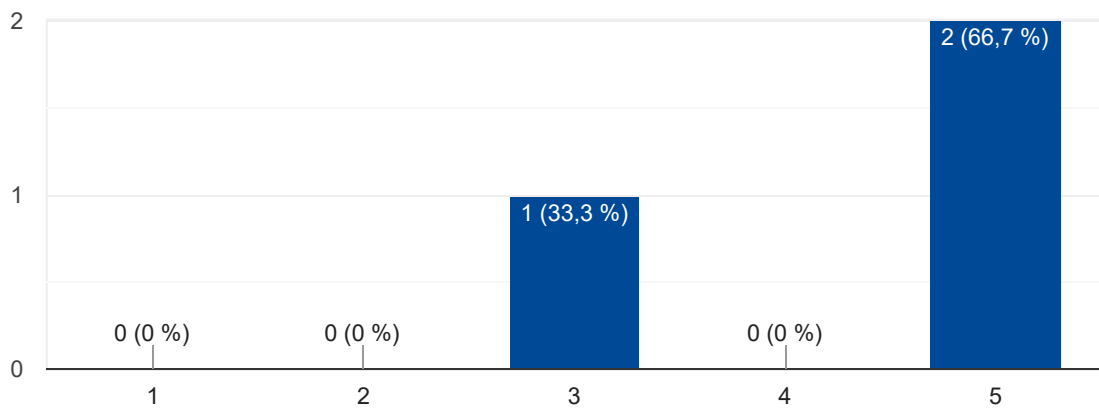
How satisfied are you with the solution of exporting data to a bridge file on the CostCalc page? (Only available for CostCalc, Superusers and Admin)

3 svar



How satisfied are you with the new forms compared to the old solution? (With multiple steps, and progressbar)

3 svar



Which feature of our web-application are you most pleased with?

3 svar

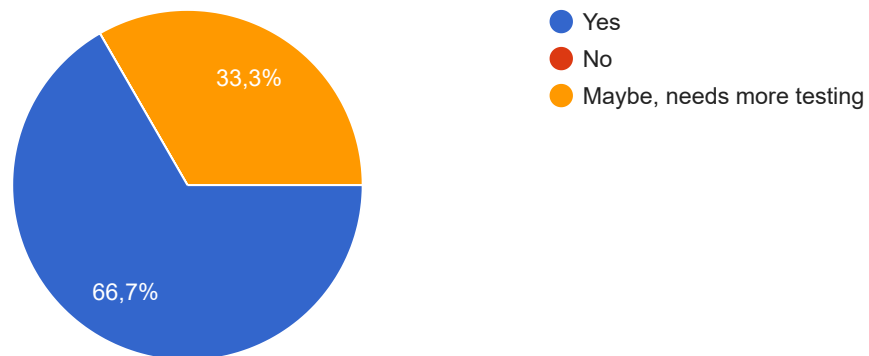
The GUI looks very nice

TFC

Enkelt brukergrensesnitt.

Do you want to continue using the application?

3 svar



Please elaborate about your answers or other things the questions did not cover about the software.

1 svar

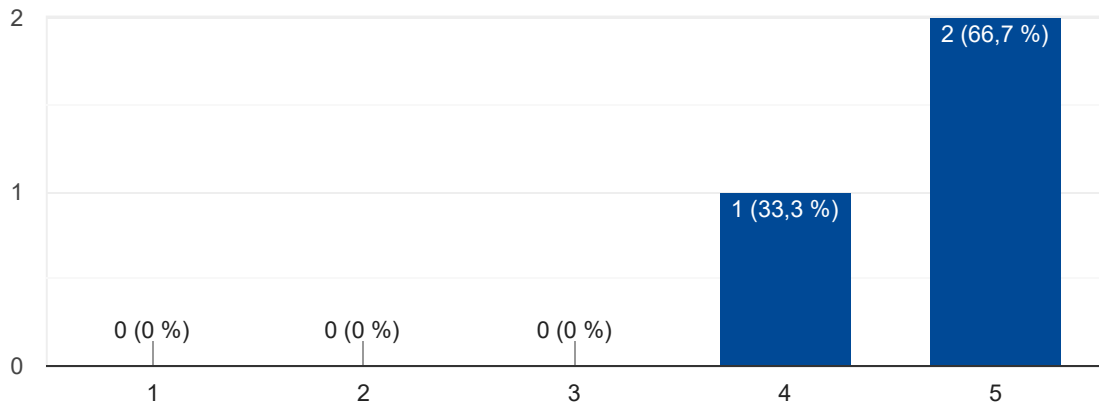
The GUI appearance really want you to use the application

Process



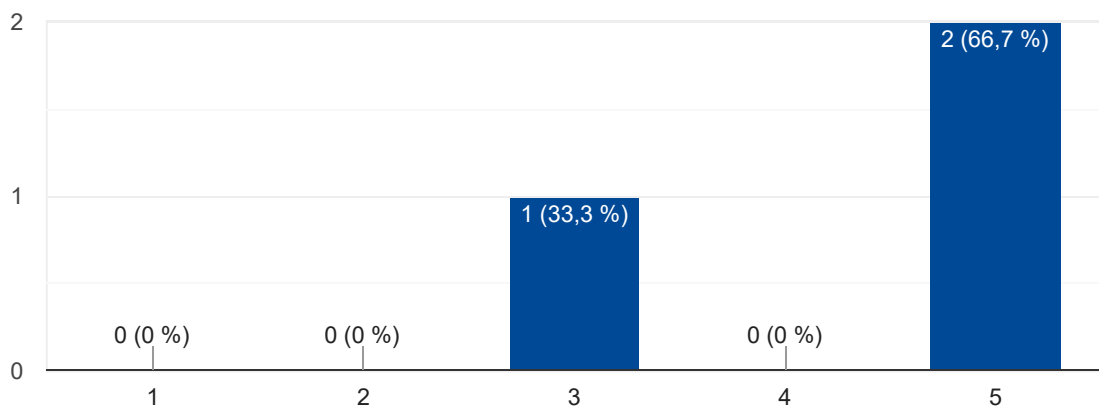
To what grade has the development team shown initiative and came with their own suggestions during the development?

3 svar



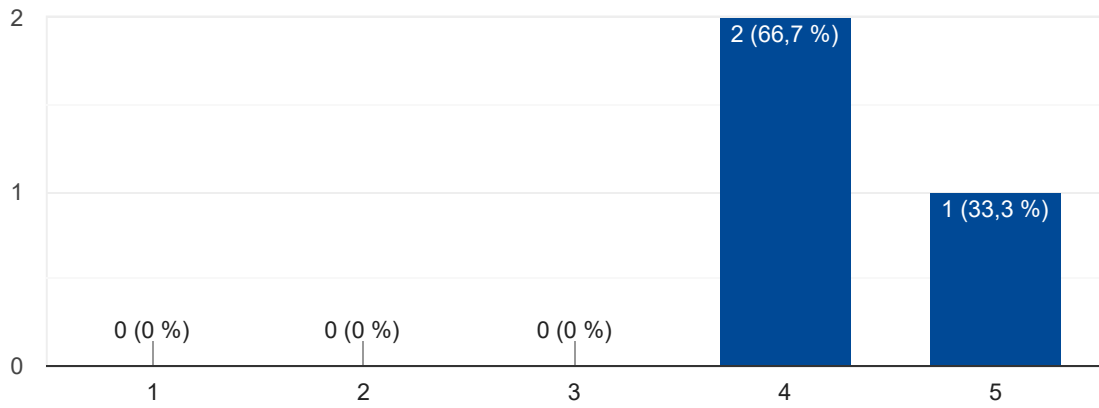
How satisfied are you with the development teams time management?

3 svar



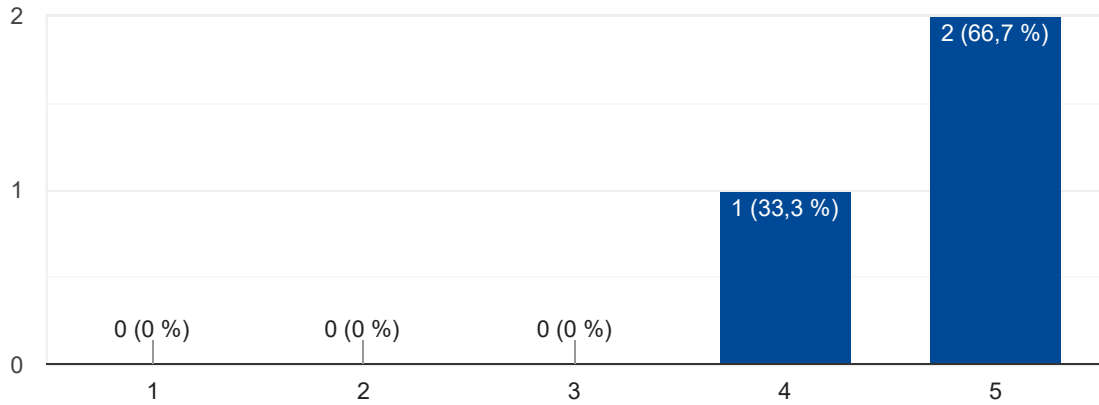
How satisfied are you with the development teams professionalism during this project?

3 svar



In total, how satisfied are you with the development teams effort?

3 svar



Please elaborate about your answers or other things the questions did not cover about the process.

2 svar

The team has worked structured and well organized. Changes and suggestions have been implemented directly upon request. The communication with and within the team has worked well. All-in-all a good project!

Still some bug fixing required, smaller design adjustments required.

Other lack of functions caused by Benteler late or missing input.

A more frequent cross check on planned solutions from students vs Benteler expectations before putting too much work into solving task could have increased the completion level a bit to date.

Dette innholdet er ikke laget eller godkjent av Google. [Rapportér misbruk](#) - [Vilkår for bruk](#) - [Retningslinjer for personvern](#)

Google Skjemaer



