Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for IKT og realfag

Kleive Bredeli, Ivar
Watt Gravdehaug, Steffen
Øveraas Karlstrøm, Vegard
Misund, Kenneth

Web-løsning med produkt-konfigurator

Konfigurasjon av en Pleat-kjøler

Bacheloroppgave i ingeniørfag, Data
Veileder: Kjell Inge Tomren
Mai 2023

Hydroniq
PLEAT CONFIGURATOR

NTNU
Kunnskap for en bedre verden

Kleive Bredeli, Ivar
Watt Gravdehaug, Steffen
Øveraas Karlstrøm, Vegard
Misund, Kenneth

Web-løsning med produkt-konfigurator

Konfigurasjon av en Pleat-kjøler

Bacheloroppgave i ingeniørfag, Data
Veileder: Kjell Inge Tomren
Mai 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for IKT og realfag

NTNU

Kunnskap for en bedre verden

Sammendrag

Hydroniq Coolers har vært på utkikk etter en ny løsning som skal kunne effektivisere den interne løsningen selskapet bruker for å konfigurere varmevekslerproduktet Pleat. De har uttrykt et ønske om å få utvikle en avansert konfigurator, en som ikke bare vil gi muligheten til å skape de mest hensiktsmessige produktet utfra innskrevne inputer, men som også skal kunne tilby kunder ytterligere informasjoner som GA-tegninger, omfattende kalkulasjoner og muligheten til å ta kontakt med selskap om kjøp av produktet.

Planen er at denne løsningen skal integreres i deres nettsideside, og har som mål å betydelig redusere arbeidsmengden til salgsavdelingen, samtidig som den er en god løsning for kunder å skaffe seg produktet de søker. Dette innebærer nye muligheter som vil muliggjøre at brukere skal få kunne konfigurere varmevekslere selv direkte fra nettsiden. Vi valgte å søke om denne bacheloroppgaven fordi oppgaven i seg selv er både krevende og spennende, men også fordi denne muligheten gir oss verdifull kunnskap og erfaringer fra et arbeidsfelt vi har lite kunnskap i fra før.

En varmeveksler, eller mer bestemt en «Pleat varmeveksler», er et kjølesystem som kvitter seg med overskuddsvarme fra systemer slik at systemer som båtmotorer og kraftproduksjon reduserer varmeenergien som produseres. Hvordan det gjøres er ved å overføre varmen fra et medium til et annet uten at de to kommer i direkte kontakt. Hovedmålet her er å forhindre overoppheting.

For å få mer informasjon rundt problemstillingen, tok gruppen kontakt med oppdragsgiver. Det ble diskutert mer om krav og forventinger rundt funksjonalitet til konfiguratoren og hvordan den best kunne utvikles etter deres visjon. Dialogen med bedrift har vært viktig for å sikre at arbeid er i tråd med deres forventninger og behov for prosjektet.

Det ble bestemt at konfiguratoren skal brukes som et webgrensesnitt som kommuniserer med et Excel-fil for datahåndtering. I Excel-filen vil avanserte kuldetekniske kalkulasjoner foregå. Konfiguratoren skal hente ut passende produkter basert på innskrevne brukerinputer, og resultatet skal vises i form av en tabell som viser de 10 beste produktene. Produktene vil være sortert etter areal (fra minst til størst), som åpenbart vil være mest rimelig for kunde. Produktkonfiguratoren vil være tilgjengelig på Hydroniq sine nettsider, og teamets løsning vil bli satt opp på en server som vil drives av bedriftens samarbeidspartnere.

Det har blitt utviklet to forskjellige måter å bruke konfiguratoren på, hvor den første løsningen er minstekravet for å få ut en Pleat-kjøler. Bruker må fylle ut tre av fire obligatoriske inputfelt for å kunne få muligheten til å kalkulere et resultat. Den andre mer avanserte løsningen gir brukeren muligheten til å fylle inn syv ekstra valgfrie inputfelt utover minstekravet.

Som resultat av alt arbeid og testing er både oppdragsgiver og gruppen godt fornøyde med applikasjonen. Det er planen at applikasjonen skal tas i bruk kommersielt på deres nettsider, så derfor kan det medføre noe arbeid etter levert bacheloroppgave for å sørge for at driften av applikasjonen holder god stand. Det har blitt gjort en avtale med firma om å ha en kontaktperson som selskapet kan kontakte dersom problemer oppstår med applikasjonen. Det vil bli signert under på en avtale om rettigheter til bruk av program.

Abstract

Hydroniq Coolers has been looking for a new solution that will be able to streamline the internal system used to configure the Pleat heat exchanger product. They have expressed a desire to be able to develop an advanced configurator, which will not only provide the opportunity to generate the most appropriate products based on collected inputs, but also be able to offer customers additional information such as GA-drawings (General Arrangement), comprehensive calculations and the possibility of contacting the company about purchasing an item.

The plan is for this solution to be integrated into their website and aims to reduce the workload of the sales department while being a good option for customers to get the product they need. This includes bringing new opportunities to be able to configure the heat exchanger yourself directly from the website. We chose to apply for this bachelor's thesis because the thesis itself is both demanding and exciting, but also because this opportunity gives us valuable knowledge and experience from a field of work in which we previously had little knowledge.

A heat exchanger or more precisely a "pleat heat exchanger" is a cooling system that gets rid of excess heat from systems so that systems such as HAVC, boat engines, cars and power generation reduce the heat energy produced. How it is done is by transferring the heat from one medium to another without the two media coming into direct contact. The main purpose of a heat exchanger is to prevent overheating.

To get more information about the issue, we had open communication with the company. More was discussed about requirements and expectations around the functionality of the configurator and how it could best be developed according to their vision. The dialogue with the company has been important to ensure that the group’s work is in line with their expectations and needs for the project.

Together with the company, it was eventually decided that the configurator will be used as a web interface that communicates with an Excel sheet for data handlin. In his file, advanced cold engineering calculations and data will be handled. The configurator must retrieve suitable products based on entered user inputs, and the result must be displayed in the form of a table showing the 10 best products. The products will be sorted by area (from smallest to largest), where the smallest area will be the least expensive pleat product for customers. The product configurator will be available on Hydroniq's website, and the team's solution will be distributed on a server that one of the client’s cooperation partners will host.

Two different ways of using the configurator have been developed, with the first solution being the minimum requirement to be able to calculate a Pleat cooler. The user must fill in three out of four mandatory inputs to get a result. The second solution gives the user the opportunity to fill an additional seven advanced inputs beyond the minimum requirement.

As a result of all the work and testing, both Hydroniq and ourselves are very satisfied with the application. The plan is for the application to be used on their website and therefore it may involve some work after the bachelor's task to ensure that the operation of the website goes well. An agreement has been made with the company to have a contact person that the company can contact if the application encounters any problems. An agreement on rights to use the program will be signed between all parties.

Forord

Denne rapporten fungerer som en detaljert beskrivelse av utviklingsprosessen til et konfigurasjonsverktøy. Denne utviklingsprosessen har involvert en rekke faser og et flertall av samarbeidspartnere involvert i prosessen for å nå sluttproduktet. Dette kommer av at Hydroniq har benyttet seg av forskjellige tredjepartsselskaper. Et selskap har hatt ansvaret for å vedlikehold og drifte websiden deres, mens et annet firma har hjulpet til med å sette sammen Excel-filen deres.

Prosjektet ble valgt på grunn av dets kompleksitet og muligheten om en betydelig faglig utfordring. SCRUM ble tatt i brukt for prosjektledelse, der hver av sprintene har hatt fokus på forskjellige aspekter og krav til prosjektet. De første tre sprintene ble lagt av til å identifisere de mest hensiktsmessige rammeverkene, bibliotekene og programvaren som skal tas i brukes i prosjektet. Skisser ble opprettet gjennom designverktøyet Figma for å gi en mer visuell fremstilling av nettsiden, men også for å kunne foreslå et eventuelt design. Her ble det det gått gjennom flere utkast før endelig konseptskisse ble bestemt.

I sprint tre og fire ble mye av tiden brukt til å integrere Excel-fil inn i applikasjonen. Etter å ha konsultert med faglærer, ble Apache POI-biblioteket identifisert som et verktøy som var best egnet for å håndtere Microsoft Excel. Apache POI er åpen kildekode, og har et bredt utvalg av funksjoner som også gjorde biblioteket mer attraktivt å ta i bruk.

Biblioteket tilbyr sterk funksjonalitet, som inkluderer muligheten for å lese, skrive, kjøre celler og regne ut formler. I tillegg gir det støtte for tverrkompatibilitet noe som gjør at både gamle og nye Excel formater kan brukes. Oppsettingen av Apache POI la grunnlaget for videre utvikling av applikasjonen, noe som gjorde at disse to sprintene ble et betydelig skritt fremover i prosjektet.

Sprint fem involverte arbeid med back-end-kommunikasjon og samarbeid med en Excel-fil for å lese og skrive data. Sprint seks gikk mer inn på implementering av resterende funksjonaliteter, dokumentasjon av kode, og feilretting. Sprint sju og åtte inneholdt for det meste testing, ferdigstilling og oppsett av applikasjon på server.

Resultatet er en fungerende konfigurator som kan produsere en resultatliste med produkter som matcher inputene som blir gitt i front-end. Den har en resultattabell som sorterer produktene etter minste areal.

Samarbeidspartnere inkluderer Hydroniq Coolers AS, som har stått for selve oppgaven. Ledelse og salgsavdeling har deltatt aktivt med å gi tilbakemeldinger og bidra til å utbedringer av konfiguratoren. Videre, har gruppen hatt jevnlig kontakt med Otto Godeset i Pleat AS, som er ansvarlig for oppdatering og vedlikehold av Excel-filen. Havnevik AS er ansvarlige for Hydroniqs nettside, og har jobbet sammen med gruppen for å få applikasjonen distribuert på Hydroniq sin nettside. Disse partnere har alle vært en enormt viktig del av bachelorarbeidet.

Prosjektet ble en lærerik og produktiv erfaring, og gruppen sender en enorm takk til alle involverte parter for deres engasjement og bistand.

Innholdsfortegnelse

Kapittel 1. Introduksjon og relevans

1.1 Bakgrunn

Hydroniq Coolers AS er et norsk firma som holder til på Ellingsøya, som spesialiserer seg på utvikling, design og produksjon av høyeffektive kjølesystemer og platevarmevekslere. Selskapets hovedfokus er på å utforme løsninger som er både effektive og enklere å vedlikeholde for brukere.

I 2012 ble firmaet etabler, Hydroniq Coolers er en del av SMV Invest, som er holdingselskapet til Sperre Mek Verksted AS. Tidligere var selskapet en del av Sperre-konsernet, en organisasjon kjent for å produsere startluftkompressorer gjennom Sperre Industri. Sperre Industri har en lang historie som går helt tilbake til tidlig 1970-tallet, hvor de produserte varmevekslere for marinefartøy og landbasert industri.

Selskapet byttet navn fra «Sperre Coolers» til «Hydroniq Coolers» i 2019, etter at merkevaren «Sperre» og Sperre Industri AS ble solgt til et privat selskap.

Varene Hydroniq leverer i dag består av rack-kjølere, pleat-kjølere og «shell and tube» varmevekslere. Produksjonene av disse produktene skjer i deres lokale på Ellingsøya i Ålesund kommune. Tall fra proff.no viser at Hydroniq hadde en omsetning på 63.5 millioner norske kroner i 2022.

Markedene som bedriften primært leverer produkter til er marinefartøy, industrielle bedrifter og kraftverk. Hydroniq Coolers har et lidenskapelig team bestående av 15 ansatte spesialisert i å utvikle innovative og bærekraftige kjøleteknologier som gir stor verdi til kundene.

Det har blitt levert 3000 elementer av Pleat til et stort antall av skipsfartøy, hvor over 150 båter tar i bruk Pleat for kjøling. Pleat benyttes også til en rekke applikasjoner utenfor maritim bransje. For produktet Rack har det blitt levert til mer enn 100-fartøy og over 1000 rack-elementer har blitt installert.

Bacheloroppgaven vi skal utføre i samarbeid med Hydroniq går ut på å lage en konfigurator for disse Pleat-kjølerne. De ønsker å tilby kunder muligheten til å konfigurere en Pleat-kjøler etter spesifikke krav på deres webside. Dette løses ved å starte utvikling av en webapplikasjon som på best mulig vis kan møte deres behov. En produktkonfigurator i dette tilfellet vil være et grensesnitt med mulighet for kunder å sende inn spesifikke inputverdier som til sammen vil resultere i de produktene som vil passe kundes krav best.

En Pleat, mer spesifikt en platevarmeveksler, er en type kjøler (varmeveksler) som benytter en stabel av plater for å overføre varme mellom væsker. Målet til en Pleat er å tillate to væsker til å utveksle varme mellom dem uten direkte kontakt med hverandre. Dette gjøres for å bli kvitt overskuddsvarme fra systemer. Denne type kjøler bruker væske til å overføre varme fra en væske til en annen. Mer om prosessen til en Pleat kan du lese i teoridelen (se 2.1.2). Et slikt system finner vi ofte i maritime områder som bruker motorer og maskiner i skipsfartøy.

En produktkonfigurator for en Pleat vil gi brukere (skipsingeniører osv.) en mer personlig tilnærming til produktet, og gi bedre kontroll over hvilket produkt som kjøpes med spesifikasjoner i henhold til ønsker. Dette kan føre til bedre merkevare, økt salg og

positive tilbakemeldinger rundt om i bransjen for bedriften. De kjøler-variantene som skal kunne konfigureres av en Pleat er en basic parallell kjøler, og en basic seriekoblet-kjøler.

1.2 Problemstilling

Hovedutfordringen i denne oppgaven er å utvikle en webbasert applikasjon som kan kommunisere feilfritt med Hydroniq sine kalkulasjoner i en Excel-fil, for så å vise de best passende produktene i henhold til kundes spesifikasjoner. Dette innebærer at frontend- og- backend-delene av applikasjonen må integrere Excel-filen på en måte som gjør at brukerne kan konfigurere produktene i sanntid, mens inputverdiene som blir sendt inn blir automatisk lagret i Excel-filen. Dette vil kreve grundig undersøkning av ulike fremgangsmåter, som kan gi oss god forståelse om hvordan vi kan implementere funksjonaliteter som trengs for å manipulere Excel-filen fra applikasjonen.

Årsaken til integrasjon av applikasjon og firmaets Excel-fil, er hovedsakelig på grunn av oppdragsgiver mangel på kompetanse for å videreutvikle og vedlikeholde en web-applikasjon. Derfor skal det legges til grunn at bedriften må kunne videreutvikle Excel-filen med nye kalkulasjoner og produkter, uten at det skal ha noe å si for applikasjonen.

Det må også tas hensyn til samtidighet og validering i applikasjonen. Inputverdier som brukere sender inn har en minimal sjanse for å bli lagret i Excel-filen samtidig, noe som kan føre til konflikter og uønskede resultater hvis dette ikke blir håndtert på en hensiktsmessig måte. En mulig løsning på dette kan være å bruke synkronisering for å sikre at kun en kunde har tilgang til filen samtidig. Validering er også avgjørende for å sikre at produktkonfiguratoren tar i bruk den riktige Excel-filen og at plasseringen til filen er i riktig mappe.

Til slutt, er det viktig å sørge for at produktkonfiguratoren er mest mulig brukervennlig, slik at brukerne enkelt og effektivt vil kunne navigere seg gjennom applikasjonen samt få konfigurere produktene de ønsker.

1.3 Mål

Vi ser for oss følgende mål for bacheloroppgaven:

- Designe og utvikle en web-basert produktkonfigurator som gir kunder muligheten til å velge et produkt basert på deres inputverdier og ønsker.
- Å utvikle muligheten for applikasjonen til å lese og skrive til Excel-fila.
- Å sørge for at applikasjonen er brukervennlig, slik at konfiguratoren vil gi en god representasjon av mulige produkter ved bruk av funksjonalitetene vi har implementert.
- Sørge for at applikasjonen er enkel å vedlikeholde og utvide etter at bacheloroppgaven er avsluttet, slik at firma kan gjøre videre arbeid på applikasjonen senere.

1.4 Begrensninger

I utgangspunktet er det et par begrensninger i prosjektet som det er viktig å være klar over. For det første vil det ikke benyttes en database for datalagring, noe som kan påvirke sikkerhet, dataintegritet, sikkerhetskopiering og samtidighet. Her må også gruppen sammen med bedrift bli enige om hvordan samhandling med Excel-fil skal foregå, da uønskede konflikter kan føre til en rekke feil i applikasjon.

Det vil forsøkes å minimere mulige problemer ved å implementere alternative løsninger, som for eksempel "file-locking" for å sikre synkronisert tilgang til Excel-filen og regelmessig sikkerhetskopiering av filen for å unngå tap av data.

En One-drive vil også bli satt opp for Excel-filen som kun utviklere, bedrift og samarbeidspartnere skal ha tilgang til slik at kun noen få har mulighet til å endre innholdet i Excel-filen. Det må også lages grundige guidelines slik at det som kan og ikke kan endres spesifiseres.

Det kan være spesifikke Excel-verdier som må manipuleres fra applikasjon. Dette gjør at dersom nye data og parametere legges til av bedrift, kan det potensielt være nødvendig med endringer i applikasjonen for å kunne hente inn dette. Dette kan gjelde flere deler av filen, hvilket betyr at gruppen må prøve å forhindre dette på best mulig vis for å skape en god samhandling.

En annen begrensning ved prosjektet er at det kan være tidkrevende å manuelt laste opp eventuelle illustrasjoner og modelltegninger for hvert produkt om en ikke tar i bruk en database for dette. Det kan påvirke arbeidsmengden og effektiviteten for firmaet i senere tid, særlig når nye produkter legges til, da en må inn i prosjektet manuelt for å legge til PDF-filer av tegninger. Det vil likevel bli forsøkt å finne en effektiv arbeidsflyt for å minimere den negative innvirkningen av denne begrensningen.

Når vi går videre med utviklingen av konfiguratoren, må en huske på at det alltid kan oppstå utfordringer og begrensninger i løpet av prosessen, spesielt med tanke på at det vil bli tatt i bruk teknologi som gruppen ikke er kjent med fra før av.

Gruppen vil uansett forsøke å arbeide innenfor tidsrammen og ressursene som er tilgjengelige. Her er det en mulighet for at en ikke vil kunne utforske alle mulige løsninger eller teknologier i dybden, men medlemmene er fast bestemt på å levere et solid og funksjonelt produkt innenfor disse rammene.

1.5 Rapportstruktur

Rapporten er delt inn i flere kapittel, hvor hvert kapittel tar for seg et spesifikt aspekt av bacheloroppgaven.

Kapittel 1: Introduksjon og relevans

Dette kapittelet gir en introduksjon til oppgaven, beskriver bakgrunnen for prosjektet, og tar for seg problemstillingene og målene som skal utforskes og oppnås. Det vil også bli beskrevet hvorfor bedriften har et behov for denne applikasjonen.

Kapittel 2: Teoretisk grunnlag

Dette kapittelet representerer det teoretiske grunnlaget for oppgaven og skal gi en sammenheng mellom vårt produkt og eksisterende resultater. Vi må også ta for oss andres forslag til løsninger der vi kan sammenligne med deres løsninger eller forklare hvorfor vi har valgt enn annen. Viktig å bruke gode kilder til teorigrunnlaget som har ført oss til å velge vår fremgangsmåte, slik at vi kan argumentere og vurdere foreslått løsning.

Kapittel 3: Materialer og metoder

Dette kapittelet beskriver hvordan det har blitt planlagt å gå fram for å løse oppgaven og synliggjøre framgangsmåten. Målet er å beskrive alle sider av prosjektet. Kapittelet er 2-delt hvor den ene siden skal beskrive organiseringen av prosjektet. Ta gjerne med prosedyrer, planleggingsprosessen og litt om kvalitetssikring av arbeidet.

På den andre siden skal vi beskrive litt om verktøy og teknologier som er tatt i bruk for å realisere prosjektet. Beskriving av metodikker skal også forekomme her.

Kapittel 4: Resultater

Dette kapittelet presenterer resultatene av oppgaven, inkludert beskrivelse av grensesnittet og funksjonaliteten, og hvordan den oppfyller kravene som ble satt for prosjektet.

Kapittel 5: Drøfting

Dette kapittelet inneholder en kritisk analyse og drøfting av oppgaven, inkludert styrker og svakheter ved design og funksjonalitet, samt mulige forbedringer og utvidelser som kan gjøres.

Kapittel 6: Konklusjon og anbefalinger

Dette kapittelet oppsummerer resultatene og drøftingen av oppgaven, og gir en konklusjon om oppnåelsen av målene som ble satt. Det gir også anbefalinger for videreutvikling og vedlikehold av produktkonfiguratoren.

Sammfunnspåvirkning

I dette avsluttende punktet vil gruppen gå gjennom i hvilken grad bacheloroppgaven kan påvirke samfunnet positivt, og hvilke bærekraftige følger applikasjonen kan føre til.

1.6 Forkortelser og definisjoner

1.6.1 Forkortelser

API – Application Programming Interface (Programmeringsgrensesnitt)

REST – Representational State Transfer

MVC – Model View Controller (Modell-visning-kontroller)

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

VVS og HVAC – Varme-, ventilasjons- og sanitærteknikk

MVP – Minimum Viable Product (Minste brukbare produkt)

1.6.2 Definisjoner

Java – Et objektorientert programmeringsspråk brukt til utvikling av applikasjonen [22]

Front-end – Den visuelle siden av applikasjonen som bruker vil kommunisere med

Back-end – Baksiden av applikasjonen, som står for funksjonalitet og datahåndtering

JavaScript – Programmeringsspråk som sørger for dynamikk og kommunikasjon mellom elementene på nettsiden [12]

Spring Boot – Rammeverk som hjelper til med å konfigurere og sette opp applikasjonen [11]

Thymeleaf – Mal-motor

Pleat – Plate-varmeveksler produsert av Hydroniq Coolers AS

JUnit – Verktøy for å teste Java-programvare

IntelliJ – Program brukt for utvikling av programvare

Paradigm – en spesifisert metode for å løse et problem eller gjøre en oppgave [27]

Kapittel 2. Teoretisk grunnlag

2.1 Varmeveksler

Varmevekslere fungerer ved å overføre varme fra en væske til en annen. Ved å unngå direkte kontakt er det mulig for begge væskene å nå målet sitt. Det finnes mange forskjellige varianter av varmevekslere fra platevarmevekslere, og de blir tatt i bruk i en rekke industrielle og kommersielle sektorer som klimaanlegg, kjøleenheter eller til og med kraftverk. Utnyttelsen av ellers tapt varme gjør at varmevekslere bidrar til energieffektivitet og bærekraft. [1]

2.1.1 Hva er det?

En typisk varmeveksler bruker en rekke plater eller rør for å lage kanaler for en strøm som inneholder to væsker eller gasser. En vanlig metode for å overføre varme mellom de to kanalene, innebærer bruk av vegger laget av materialer med høy varmeledningsevne som aluminium eller rustfritt stål. En vekslers form varierer i forhold til ytelseskriteriene, størrelse og trykkfall. [1]

Noen eksempler på ulike typer varmevekslere kan være plate- eller rørformede. De har sine egne unike fordeler samtidig som de har ulemper. Kravet til en ideell varmeveksler avhenger av en rekke faktorer, som trykk- og temperaturkrav, væsketilstand og termisk ytelse.

2.1.2 Pleat

Hydroniq sitt Pleat-produkt er en platevarmeveksler spesielt utviklet for å tilby høy termisk effektivitet, lavt trykkfall og kompakt størrelse. Pleat-varmeveksleren bruker et innovativt foldedesign på platene som øker varmeoverføringsflaten og forbedrer turbulensen i væskestrømmene. Dette resulterer i en mer effektiv varmeoverføring sammenlignet med konvensjonelle platevarmevekslere og reduserer energiforbruket i systemet.

Pleat-varmeveksleren har blitt designet med tanke på slitestyrke og pålitelighet, ved bruk av materialer som ikke ruster eller blir skadet i høye temperaturer. Det er enkelt å installere og vedlikeholde, takket være den kompakte utformingen. Systemkravene og applikasjonene kan lett tilpasses.

Hovedmålet med en Pleat-varmeveksler er å overføre varme mellom to medier uten at de er i direkte kontakt. Det gjøres for å kvitte seg med uønsket overskuddsvarme fra forskjellige systemer. Dette bidrar også til å forhindre overoppheting.

Ved hjelp av det innovative Pleat-designet kan man redusere energiforbruket og klimagassutslippene, noe som bidrar til overgangen til mer bærekraftige og miljøvennlige teknologier. [9]

2.2 Relevans til eksisterende resultater

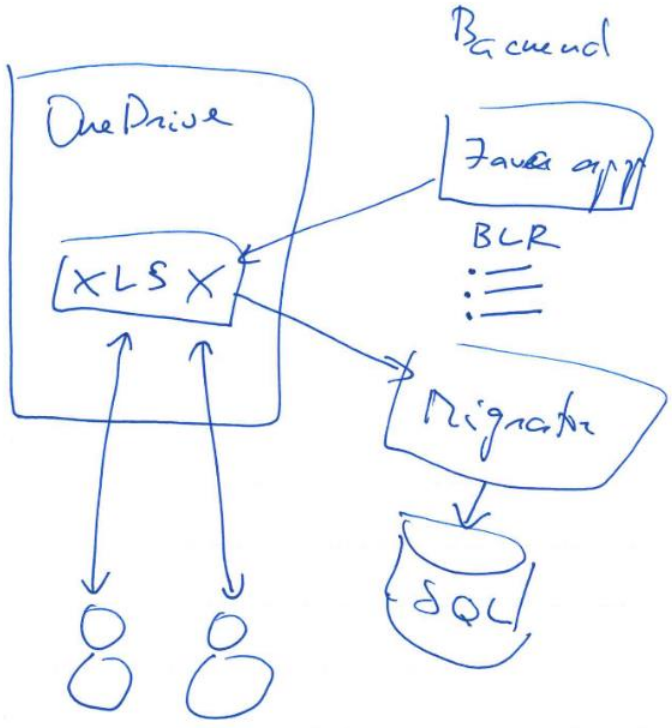
I denne delen skal det oppsummeres eksisterende resultater som er relevant for prosjektet, samt diskuteres ulike fremgangsmåter eller løsninger foreslått av andres arbeid i feltet. Målet er å få presentert en oversikt over den teoretiske bakgrunnen for bacheloroppgaven, og hvordan dette har påvirket applikasjonen.

2.2.1 Forslag til fremgangsmåter

Utvikling av en nettbasert produktkonfigurator krever at det gjøres sammenligninger av ulike teoretiske tilnærminger.

Det er ofte vanlig å konvertere data samlet i et Excel-ark inn i en database, det er en fremgangsmåte som brukes av utviklere ofte. Denne tilnærmingen forbedrer datahåndteringen betraktelig, men en stor barriere er at konvertering av et såpass stort regneark med avanserte kuldetekniske kalkulasjoner kan være tidkrevende, komplekst og tungvint. Dette kan gå utover den videre prosessen av oppgaven, og vil også gå utover videre arbeid da firmaet ikke har kompetanse i databehandling, utvikling eller IT-arbeid. [2]

En alternativ tilnærming som har blitt foreslått av fagperson er å bruke en skybasert lagringsløsning, mer spesifikt OneDrive, for å lagre og dele Excel-filen med firma. Dette kan forenkle samarbeidet mellom applikasjon og superbruker, og redusere behovet for en egen database. Denne løsningen kan gjøre det enklere å lese og skrive data direkte til Excel-filene dersom en kan finne et bibliotek som støtter denne type manipulasjon fra en applikasjon. Dette kan også begrense muligheten for samtidig redigering og øke risikoen for konflikter etter endringer (som f.eks. oppdatert versjon av regneark).



Figur 2.1 Forslag til løsning fra fagperson

2.2.2 Mulig tilnærming

Det finnes flere åpen-kilde biblioteker og verktøy som lar utviklere arbeide med Excel-filer og integrere dem med andre systemer. Eksempler på slike biblioteker inkluderer Apache POI og GcExcel for Java. Apache POI er et bibliotek som hjelper med å manipulere Microsoft Office-filer fra en java-applikasjon. Den åpner for muligheten til å sende data inn i spesifikke celler, lese data fra celler, evaluere formler og kalkulasjoner og mer. Det må undersøkes hvordan disse bibliotekene og verktøyene brukes i lignende løsninger, slik at en kan få innsikt i hvordan en kan løse problemstillingen med en slags løsning i backend. [26]

Valget av tilnærming og teknologi vil avhenge av prosjektets krav, ressurser og kompetanse, samt de spesifikke utfordringene og målene som er involvert i å utvikle en web-basert produktkonfigurator.

2.3 Designprinsipper og metodologi

2.3.1 Objektorientert programmering

Objektorientert programmering (OOP) er et paradigme (se definisjon i 1.6.2) som organiserer data eller kode som objekter. Objektene har attributter som inneholder data, og oppførsel som representeres av metoder som kan manipuleres av utvikleren. OOP baserer seg på prinsippet om innkapsling av data og atferd i et objekt. Det bruker en modulær tilnærming til programmering som gjør det enkelt å vedlikeholde, skalere og oppdatere programvare. [14]

Samhold

Konseptet samhold beskriver hvordan elementer i en modul hører sammen. Kode som har likheter eller er tett relatert bør holdes nær hverandre for å gjøre koden sammenhengende.

Ved fokus på høyt samhold får vi en kode som er enkel å teste, skrive og designe, dette på grunn av at koden er plassert og fungerer sammen. En slipper samtidig duplikasjon av informasjon.

En kode som utformes på lavt samhold utgjør at koden er spredt over hele kodebasen, noe som gjør koden uoversiktlig og vanskelig å navigere. Relasjonen til koden er også da lav. [3]

Kobling

Kobling beskriver hvordan modulene i et system avhenger av hverandre. En modul er en del av et program som inneholder funksjonalitet eller kode. Målet er at modulene bør være uavhengig fra andre moduler. Dette er for å unngå endringer i en modul ikke fører til påvirkning av en annen.

Høy kobling betyr at en modul vet for mye om den indre funksjonen til andre moduler. Når moduler har for mye innsikt i andre modulers indre funksjoner, kan det gjøre det utfordrende å håndtere endringer og svekke modulenes robusthet. Tenk deg at modul A og modul B er to medlemmer av et team. Hvis A alltid er avhengig av detaljert informasjon om hva B gjør, kan endringer i B sin arbeidsmetode forstyrre A sin ytelse og funksjon. Det er derfor viktig å begrense deres avhengighet av hverandre.

Lav kobling er altså det motsatte. Da vil moduler i liten grad være avhengig av hverandre. Fordelen med dette er at effektiviteten og fleksibiliteten av programmet vil være opprettholdt i stor grad. [3]

2.3.2 Designmønster

Designmønster er en form for løsning som brukes for å bearbeide forekommende problemer innen programvaredesign. Mønsteret fungerer som en mal eller beskrivelse av hvordan slike problemer kan løses i mange forskjellige situasjoner. [5]

Singleton-mønster

Singleton er et kreativt designermønster som passer på at bare et objekt av sitt slag kan bli opprettet. Den gir også bare et enkelt tilgangspunkt for et hvilket som helst annet punkt i koden. Dette blir gjort for å kontrollere tilgangen til en delt ressurs. [5]

Builder-mønster

Builder beskriver hvordan et kreativt designmønster brukes til å konstruere komplekse objekter trinn for trinn. Denne typen designmønster lar oss produsere forskjellige representasjoner av et objekt og bruker samme konstruksjonskode. Det kan også nevnes en "builder" er uavhengig av andre objekter. Mønsteret brukes for å løse problemet med å konstruere objekter som har mange forskjellige attributter og kan håndtere valgfrie attributter. [4]

Dependency Injection

Dependency Injection (DI) er en prosess i objektorientert programmering der kode eller en klasse er avhengig av en eller flere ressurser for å fullføre sin oppgave. Disse ressursene er ofte en del av programmet i form av klasser. Objektet som er avhengig av disse ressursene må kunne lokalisere og kommunisere med dem. DI gjør det lettere å utvikle programvare med "lav kobling", dette er fordi det øker gjenbrukbarheten til klassene i koden og hjelper til med å unngå å skrive om kode flere plasser når man gjør endringer i en klasse. [8]

2.3.3 Arkitekturmønster

Model-View-Controller

MVC (Model-View-Controller) er et designmønster som ofte brukes i programvareutvikling. MVC deler så applikasjonen i tre separate komponenter: modell (data), visning (brukergrensesnitt) og kontroller (logikken). Dette designermønsteret bidrar til å skape en mer strukturert og oversiktlig kodebase som gjør applikasjonen enklere å vedlikeholde, teste og skalere. [7]

2.3.4 Arkitektonisk stil

REST API

REST (Representational State Transfer) API er en arkitektur som følger et sett med regler og begrensninger for å utvikle en webtjeneste. REST API beskriver hvordan brukere kan kommunisere med en server gjennom HTTP-protokollen for å sende eller motta data. REST API bruker (CRUD)-operasjoner til å alterere data. [6]

2.3.5 Utvikling

Agil utvikling

Agil utvikling er en repeterende tilnærming til programvareutvikling som legger vekt på fleksibilitet, samarbeid, kundefokus og kontinuerlig forbedringer. Agil utvikling vil si at et

team leverer arbeid i små deler om gangen. Dette tillater raskere tilbakemeldinger, mer nøyaktig planlegging, og evnen til å tilpasse seg endringer raskt og effektivt. [17]

SCRUM

SCRUM er smidig arbeidsmetodikk, der målet er å utvikle og levere nye iterasjoner (sprint) av et produkt eller programvare. Lengden på en sprint varierer fra prosjekt til prosjekt, men generelt sett så ligger det mellom en til to uker. I starten av hver sprint, blir det laget en oversikt over de ulike arbeidsoppgavene som må utføres for den gjeldende sprinten. Deretter arrangerer gruppeleder et møte, hvor det blir det gått gjennom status av prosjektets arbeidsoppgaver og fordelt oppgaver til teamet. I løpet av møtet, vil hvert medlem gi status på sin oppgave og diskutere det i felleskap. Ved slutten av et møte, skal gruppeleder og medlemmene være oppdatert på prosjektets retning og fremgang. Når sprinten er ferdig, vil teamet ha en ny iterasjon av produktet som de kan vise frem til kunde eller eieren av produktet. Teamet vil dermed holde et lite møte med som ser på sprinten retrospektivt, der det diskuteres resultatet av sprinten, og hvordan teamet skal gå videre inn i neste sprint. [16]

Git

Git er et versjonskontrollsystem som hjelper utviklere med å håndtere ulike typer prosjekt gjennom effektivitet og struktur. Det er en gratis og åpen kildekode, og er designet for å håndtere både små og store prosjekter. Ved bruk av Git, er det enkelt å holde en oversikt over endringer og fremgang i arbeidet, noe som vil være vesentlig i en bacheloroppgave. [18]

2.3.6 Kvalitetssikring

Kvalitetssikring er som ryggraden i alle prosjekter innen programvareutvikling. Det er en stor del som sikrer at det ferdige produktet er i tråd med alle forventninger og oppfyller alle krav. Kvalitetssikring inneholder et bredt spekter, hvor det handler om alt fra å planlegge for kvalitet til å sikre og forbedre kvaliteten.

Kvalitetssikring inneholder jevnlige revisjoner, tester og evaluering av resultatene underveis. Målet med disse tiltakene er å avdekke eventuelle feil eller mangler som krever korrigering. Dette er for å forsikre at sluttproduktet skal være mer enn bare funksjonelt; det skal være pålitelig, effektivt og i samsvar med klientens visjon.

Automatisert testing er også en avgjørende del av kvalitetssikring. Dette innebærer bruk av ehetstesting ved hjelp av rammeverk som JUnit i utviklingsprosessen. Enhetstesting betyr at det blir testet ulike deler av koden for å sikre at ting fungerer etter forventningene. Dette sikrer at applikasjonen fungerer som den skal uten å forårsake uforutsette feil på grunn av eventuelle endringer eller oppdateringer i fremtiden. [23]

Ikke bare under testing er det fokus på kvalitetssikring, men også både i design- og implementeringsfasen. Her inngår bruk **Git** (se 2.3.5) som versjonskontrollsystem, som er med på å bidra til at applikasjoner inneholder en stabil kodebase.

Kapittel 3: Materialer og Metode

3.1 Planlegging og dokumentasjonsprosess

3.1.1 Organisering og planlegging

I oppstartsfasen ble det fokusert mest på dokumentasjon og prosjektarbeid. GitHub-repository ble satt opp, readme-fil ble opprettet, og forprosjektplanen ble startet arbeid på (se vedlegg **E**). Gruppen måtte signere samarbeidskontrakt, og det måtte signeres trepartsavtale med NTNU, vår gruppe og oppdragsgiver.

Planlegging og dokumentasjon ble støttet av en rekke teknologiske verktøy, inkludert GitHub, Discord, Teams og e-post. Discord tillot effektiv intern kommunikasjon, mens Teams fungerte som en møteplattform for både interne og eksterne møter. Dokumentasjon ble lagret i en felles mappe i Teams, og GitHub ble brukt til organisering av sprinter og tildeling av ukentlige oppgaver.

3.1.2 Kommunikasjon

Arbeid i gruppen

Gruppen har jobbet sammen jevnlig, og organisert arbeidsuken med faste oppmøtetidspunkt. Det ble daglig tildelt arbeidsoppgaver blant gruppemedlemmene. Dersom fravær har vært nødvendig, ble oppgaver enten utsatt eller omfordelt blant gruppemedlemmene. Det er også mulighet for medlemmene å jobbe overtid dersom en føler for det.

IntelliJ sin "Code With Me"-funksjon ble benyttet for samarbeid om enkelte oppgaver, noe som viste seg å være en effektiv metode for å løse komplekse problemer som et team.

Samhandling med arbeidsgiver

Det har vært sikret god kommunikasjon gjennom jevnlige møter med firma. Ved slutten av hver sprint, møttes gruppen og firma for å diskutere oppgaver som har blitt fullført og hva som skal prioriteres videre av fremtidige arbeidsoppgaver. Det ble også diskutert tidsberegning og omfanget av arbeidsoppgavene. Dette var viktig for å få avklart hvilke krav som ble stilt til oppgaven, og for å finne en fornuftig løsning. Samtidig ga klient en beskrivelse av hva deres tanke bak oppgaven var og hvordan dette kunne bidra til å hjelpe dem å oppgradere deres eksisterende løsning. Dette ga gruppen et innblikk i hvordan og hvor klienten ønsket å bruke løsningen. Resultat, funn og kritikk fra brukertester var også et kontinuerlig samtaleemne gjennom prosessen.

Før sprintmøter, ble et kort referat med stikkord skrevet for å ta med til møtet. Et gruppemedlem blir tildelt ansvaret for å notere stikkord og referat fra møte, som deretter kunne bli brukt til interne diskusjoner om fremtidige arbeidsoppgaver og oppretting av nye problemstillinger på GitHub. Arbeidsoppgaver blir lagt til i GitHub sin problemtavle, og deretter tildelt gruppemedlemmene.

3.1.3 Undersøke valg av teknologi

Den største delen av planleggingsprosessen gikk ut på å finne ut hvordan kravene fra klient kunne dekkes best mulig. Det ble sett på løsninger for applikasjon, database og server. Gruppemøter ble brukt i undersøkelsesprosessen til å finne teknologier som passet gruppen best for utvikling. Siden klient ønsket en web-applikasjon som skulle kjøres på deres server, var det enighet om at Spring Boot var beste alternativet til back-end utvikling av applikasjonen.

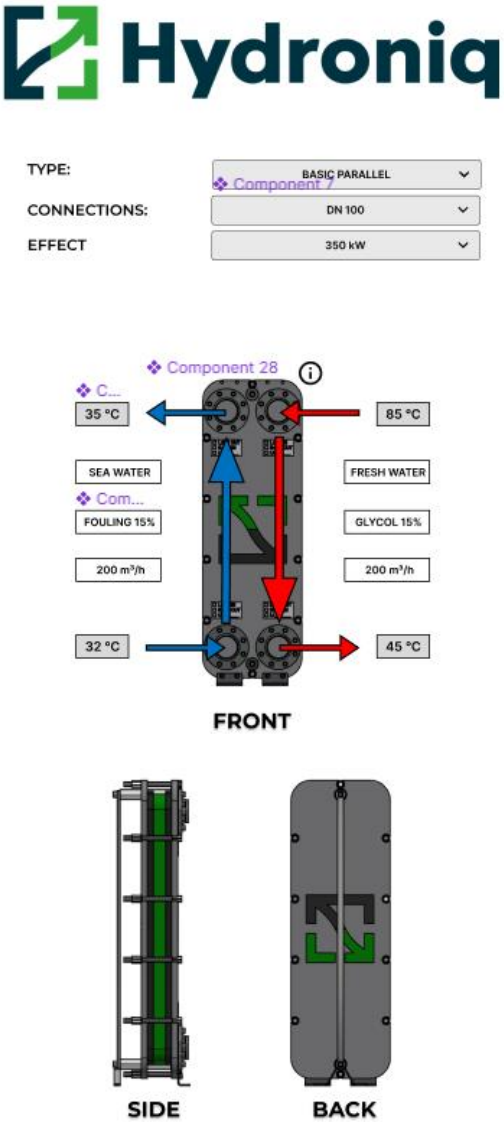
I tillegg, hadde klient et sterkt ønske om å få utviklet et web-interface for sin interne Excel-løsning. Som et resultat av dette, hadde gruppen et møte med faglærer fra NTNU for å diskutere mulige løsninger (se figur 2.1). Etter møtet, ble det gjennomført grundige undersøkelser for å komme frem til en fremgangsmåte basert på forslaget til faglærer. Til slutt, ble det konkludert med at en skulle prøve å ta i bruk et bibliotek som støtter manipulasjon av Excel-data. Biblioteket nevnt er kalt Apache POI, og er et godt kjent java-basert bibliotek som er mye brukt i feltet. Det tilbyr en rekke funksjonaliteter for å arbeide med Microsoft Office-dokumenter, inkludert Excel-filer. Apache POI er plattform-uavhengig, åpen kildekode og støtter de fleste Microsoft Office-dokumentformater, noe som gjør det til et godt valg for prosjektet. Se kapittel 2.2.1 for mer innsikt i teorien bak dette.

Forslaget fra faglærer om å bruke en "migrator" for å implementere SQL i prosjektet ble derimot skrotet, da konvertering av klientens kuldetekniske kalkulasjoner og data hadde tatt alt for lang tid å implementere i en database (grunnet ingen kompetanse i kuldeteknikk). Samtidig, har ikke de fagkyndige kontaktpersonene fra firmaet kjennskap til datahåndtering og databaser, så valget falt på bruk av et bibliotek som støtter integrering av excel-filer i Java.

3.1.4 Applikasjonsdesign

For å komme frem til et endelig design for applikasjonen, trengte gruppen en konseptskisse å følge. Det ble derfor startet arbeid sammen med oppdragsgiver for å sammen drøfte ulike idéer og tanker for å kunne danne grunnlag for en endelig konseptskisse. Oppdragsgiver ga oss en skisse av en tidligere idé til grensesnitt på vårt første møte. Denne skissen tok vi med oss inn i vurderingene våre, med målet om å utvikle et brukervennlig sluttresultat.

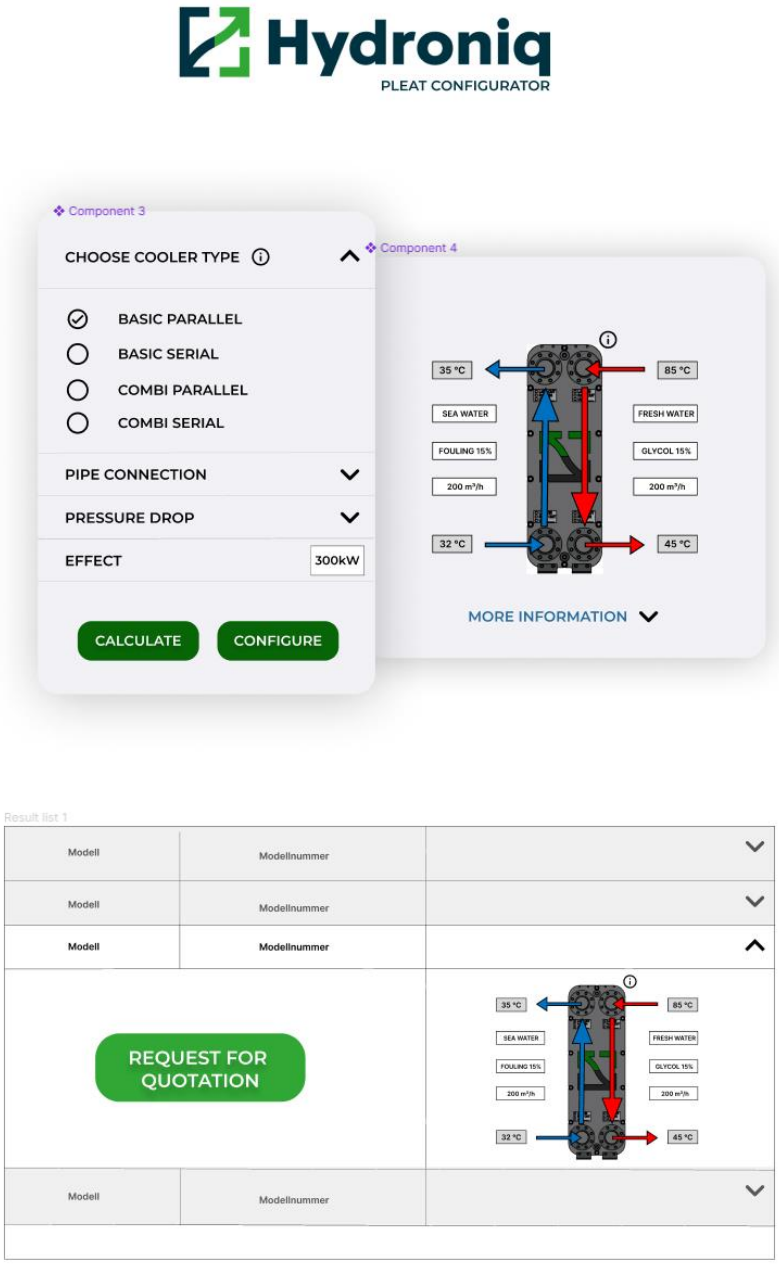
Ut ifra informasjonen vi fikk gjennom de par første møtene, ble det designet tre forskjellige skisser som ble vurdert og sammenlignet på kommende møter.



Figur 3.1 Første utkast

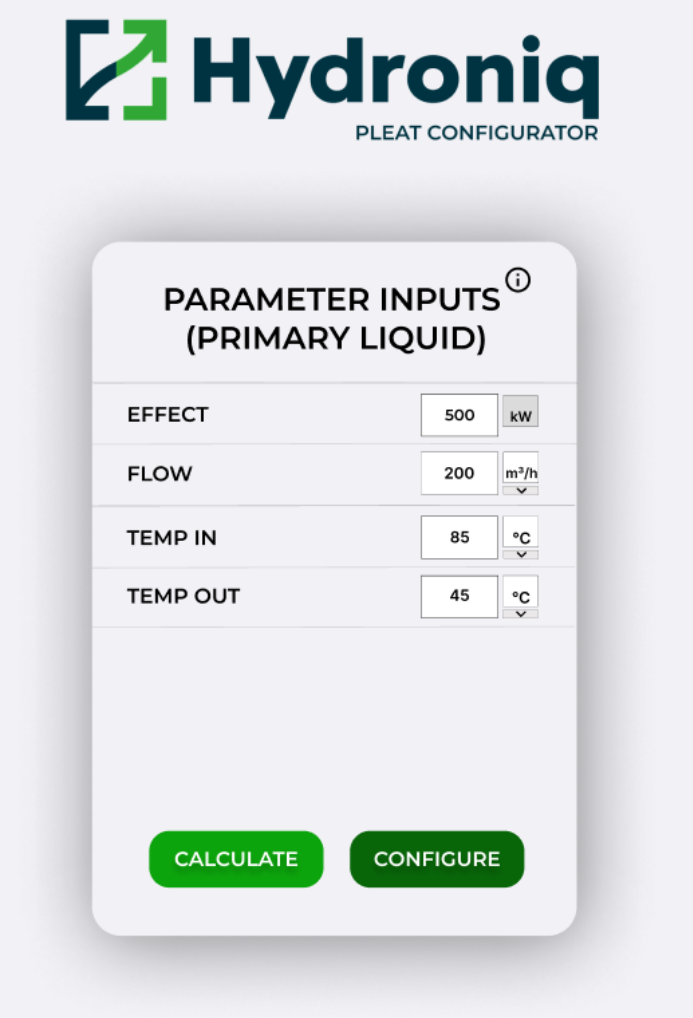
Første utkast (fig. 3.1) var høyst inspirert av forslaget til oppdragsgiver, som en kan si hadde et mer visuelt preg. Konklusjonen her ble at denne ideen ville bli for lite brukervennlig, og at den mangler en konkret struktur. Her var oppdragsgiver og gruppen samstemte gjennom hele prosessen, slik at en raskt kunne forbedre neste utkast.

Andre utkast (fig. 3.2) skapte mer interesse, da den var med på å forme den endelige skissen i stor grad. Her ble det skapt et fundament for videre bruk med tanke på farger og design. Resultattabellen ble også tatt med videre. Frem mot neste utkast, ble det enig om at selve inputfeltene og generelle funksjoner, både på den simple og avanserte siden av konfiguratoren, måtte modereres til å være mer brukervennlig og korrekt med tanke på funksjon.



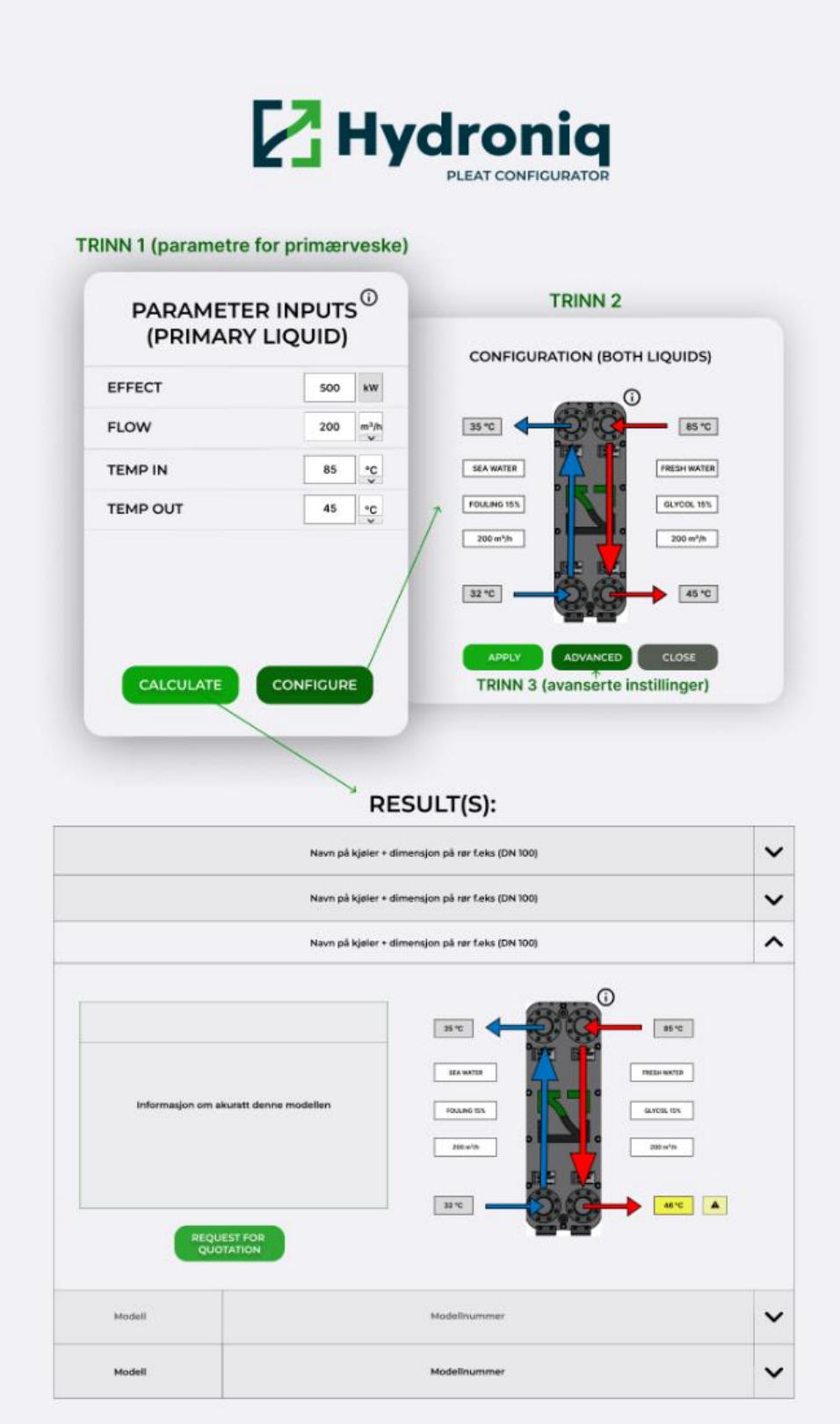
Figur 3.2 Andre utkast

Tredje utkast (fig. 3.3) gikk ut på å finne hvilke felt som faktisk måtte være med, og dette ble gjort sammen med bedrift. Her ble det prioritert å finne et godt resultat på den simple siden av konfiguratoren, da vi allerede hadde en god visjon for hvordan resultattabellen og dens innhold skulle ende opp.



Figur 3.3 Tredje utkast

Etter ulike tilbakemeldinger og konstruktiv kritikk, ble det endt opp med å lage en kombinasjon av skisse to og tre, hvor en kan se resultatet nedenfor i figur 3.4. Det ble etter hvert bestemt at den avanserte delens design ville bli for uoversiktlig og kaotisk, og derfor ble den prioritert bort. Gruppen foreslo internt at en skulle gå for en mer tekstbasert og simplistisk variant. Forslaget gikk ut på å utvide og tilpasse tilnærmingen brukt i figur 3.3 for den avanserte delen.



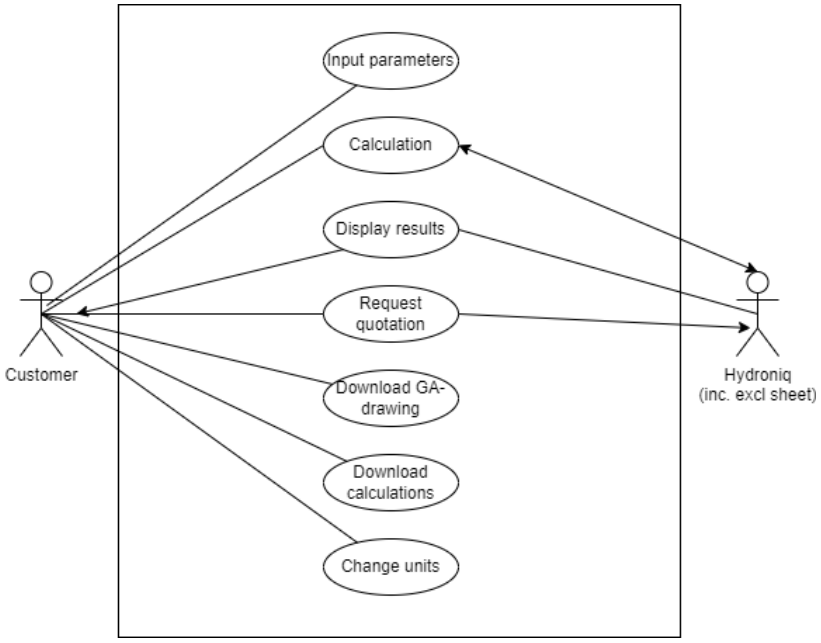
Figur 3.4 Endelig konseptskisse (Siste utkast)

3.1.5 Brukertester

I prosjektets innledende fase hadde vi ingen tilgjengelige brukere for testing. Etter hvert som prosjektet utviklet seg, ble det fremstilt et "Minimum Viable Product" (MVP) som vi gjorde tilgjengelig på en server tildelt av NTNU. Denne serveren gjorde det mulig for oppdragsgiver å teste programmet fremover. (Se vedlegg **A** for en lenke til serveren)

Vi benyttet ansatte i Hydroniq Coolers som testbrukere for programmet. Primært var det salgsavdelingen som testet programmet, men også administrerende direktør og CEO deltok i testingen. Her ble de tildelt link til serveren hvor applikasjonen var tilgjengelig, slik at de til enhver tid kunne gå inn å se på nye endringer og oppdatert versjon. Dette førte til at firma ble kjent med programvaren og kunne sammenligne programmets resultater med sitt eget Excel-ark. Dette la grunnlag for at oppdragsgiver enkelt kunne gi tilbakemelding over e-post, som førte til at gruppen enkelt og effektivt kunne ta dette med videre inn i sprint-arbeidet. Disse konstruktive tilbakemeldingene var avgjørende for videreutviklingen av programmet.

Ved å tillate oppdragsgiver å være testbrukere, ble det lettere å finne frem til problemer og feil. De innehar en kompetanse som vi ikke sitter med, slik at de vil kunne se kuldetekniske feilverdier dersom dette ville oppstå. Totalt sett la dette oppmerksomheten på produktets svakheter, som vi noterte og rettet opp i fremtidige sprinter. Denne prosessen med testing og tilbakemelding viste seg å være uvurderlig for prosjektet helt frem til innlevering.



Figur 3.1 Use Case-diagram

3.2 Tekniske aspekter

Gjennom prosjektet har vi brukt en smidig utviklingsmetode, noe som resulterte i arbeid i 2-ukers sprinter. I løpet av disse sprintene ble oppgaver identifisert og mål fastsatt for hver sprint ved hjelp av GitHub.

3.2.1 Programmeringsspråk, verktøy og teknologier

Back-end utvikling

Tidlig i prosjektet ble det besluttet at Java var det mest passende programmeringsspråket for oss som utviklere. Hovedårsaken til dette er at gruppen har opparbeidet seg betydelig erfaring innen Java gjennom tre års studier, noe som gjorde det til det språket som passet best for gruppens kompetanse. Java er et språk alle medlemmer er veldig komfortable med å bruke, og det er også det primære språket som Spring Boot er utviklet for.

Spring Boot ble tatt i bruk for å utvikle webapplikasjonen, da det forenkler utvikling av Java-applikasjoner og passet godt med tanke på gruppens erfaring med rammeverket fra tidligere studieemner.

Fra et teknisk perspektiv, er Java et godt valg for denne webapplikasjonen av flere grunner:

- For det første så forenkler det integrasjonen med andre webteknologier som HTML, CSS og JavaScript. Dette er en viktig egenskap for å skape brukervennlige webapplikasjoner.
- For det andre, kan server-side teknologier og rammeverk som Spring Boot benyttes for å akselerere utviklingsprosessen for applikasjonen. Dette gjør det enklere å vedlikeholde og skalere applikasjonen, noe som er avgjørende for en robust og langvarig løsning.

På grunn av erfaringen og disse fordelene, valgte gruppen å benytte Java som hovedprogrammeringsspråk for prosjektet. [22]

Spring Boot ble valgt som rammeverk for utvikling av applikasjonens backend-server, fordi dette rammeverket har de mest omfattende funksjonaliteter og gjør det enkelt å ta i bruk REST API (2.3.4). Gruppen har også tidligere erfaring med dette, som gjorde det enklere å implementere. Spring Boot tilbyr flere nøkkelfunksjoner som innebygde servere og automatiserte konfigurasjoner som reduserer tiden det vil ta å sette opp og konfigurere en applikasjon. Videre støtter Spring Boot et bredt utvalg av biblioteker og moduler som medlemmene kan utnytte til å forbedre applikasjonens ytelse og funksjonalitet. [11]

Apache Tomcat-sever har blitt benyttet i prosjektet som en integrert webserver av Spring Boot-applikasjonen. Fordelen med dette er at Tomcat-serveren kan testes og kjøres lokalt under store deler av utviklingsprosessen. Applikasjonen blir pakket som en JAR-fil som deretter distribueres til serveren. En annen fordel med Tomcat er at serveren blir automatisk inkludert og konfigurert i prosjektet om "dependency" har blitt implementert i prosjektet. [21]

Excel har blitt brukt til lagring og organisering av data i dette prosjektet. Dette er ingen gunstig løsning på databehandling, men klient hadde et sterkt ønske om å beholde Excel-filen og benytte den for å hente ut produkter fra filen. Det ble luftet tanken av gruppen

om å konvertere om til en stor database for å forbedre databehandlingen, men grunnet firmaets mangel på utviklingskompetanse og mengden arbeid ville ikke dette la seg gjøre. Oppdragsgiver hadde også et ønske om en slags integrering av sin interne løsning, så dette var det mest henholdsmessige en kom frem til.

Excel-filen inneholder kuldetekniske kalkulasjoner, tabeller, lister, samt et grensesnitt utarbeidet av klienten. Dette inneholdt grensesnitt-arket hvor det var lagt opp til endring og manipulering av data. Ved å benytte dette som en løsning på databehandling, er ikke dette prosjektet like robust og sterkt når det gjelder sikkerhet, dataintegritet og skalerbarhet.

Selv om Excel er et kraftig verktøy for dataanalyse og kalkulering, tilbyr det ikke de nødvendige funksjonene som databaser gjør. Det er likevel verdt å merke seg at Excel-filen vil være lett for firmaet å videreutvikle i fremtiden, da det er det de har kompetanse i og kjennskap til fra før. Det blir forhindret at oppdateringer i Excel-filen vil lage konflikt for applikasjonen, ved at endringer bare gjøres i spesifikke regneark i filen. (Se vedlegg **F** for å se brukermanualen som ble opprettet for oppdragsgiver)

Front-end utvikling

Gruppen valgte å bruke programmeringsspråket JavaScript, da det kjører i nettleseren og gir interaktivitet og dynamikk til nettsiden. Det kan også nevnes at fordelen med JavaScript er at det kan skape en bedre brukeropplevelse. Det er enkelt å lage og vedlikeholde, det gir god fleksibilitet, og det skaper en økt interaktivitet som gjør nettsiden mer engasjerende og imøtekommende for brukere. [12]

Prosjektet inneholder også HTML (HyperText Markup Language), hvilket fungerer som grunnlaget for å strukturere og presentere innholdet på nettsiden. HTML er et standard markup-språk som brukes hyppig i websammenhenger, og som i dette tilfelle vil måtte håndtere elementer som tekst, bilder, lenker, tabeller, knapper, felt og PDF-filer. [13]

Det kan også nevnes at HTML fungerer som et bindeledd for å koble opp utvidelsesfiler for JavaScript og CSS, siden "extension-files" har blitt brukt gjennom hele prosjektet. Disse utvidelsesfilene bidrar til interaktivitet og stil på siden, og gir en mer brukervennlig og engasjerende opplevelse.

Ved å inkludere JavaScript og CSS-utvidelsesfiler i HTML-filen, kan prosjektet utnytte gjenbruk av kode, dette gir bedre organisering og vedlikehold for kodebasen.

Thymeleaf kan beskrives som en Java-basert mal-motor som tas i bruk for å lage dynamiske og fleksible webapplikasjoner. Thymeleaf fungere veldig bra sammen med Spring Boot, som er et rammeverk som forenkler utviklingen av Java-baserte webapplikasjoner. Det kan også nevnes at Thymeleaf og Spring Boot ofte integreres sammen for å lage server-side applikasjoner. Mal-motoren benytter seg av "th" attributter i HTML-filer for å binde og erstatte disse med data fra Java-objekter. Dette gjør innholdet på siden mer dynamisk og tilpasset, ettersom det endres basert på data som sendes fra serveren. [15]

CSS er det som gir mulighet til å endre på utseende og formatering på elementer i applikasjonen. Her inngår alt av farger, mellomrom, responsivitet, størrelser og mer. [19]

For wireframe-skisser og planlegging av filstrukturen, benyttet vi oss av Figma. Dette verktøyet er enkelt og oversiktlig å bruke for webdesign, diagrammer og tabeller. Vi har gjort endringer på Figma-skisser i store deler av prosessen, noe som har vært med på å forme og forbedre applikasjonens design og funksjonalitet, samt gitt oss muligheten til å gi visuelle forslag til oppdragsgiver. [20]

Det skal samarbeides med Hydroniq for å få applikasjonen distribuert på deres nettsider i tiden etter innlevert oppgave. Havnevik leverer i dag nettsidene til Hydroniq, og de har vi hatt kontakt med i sluttfasen for å planlegge arbeidet med en sømløs integrasjon av applikasjonen på deres server og nettside.

3.3 Grunnlag for kalkulasjoner og resultat

For at det skal kunne utføres noen som helst form for konfigurering i applikasjonen, må det være noe i bakgrunnen som kan kalkulere verdiene som kommer inn. Selve kalkuleringen og produsering av resultat, utføres eksternt i de ulike kalkulasjonsformlene i Excel-filen. Filen består av 11 forskjellige regneark:

- **Web interface** – Dette er regnearket som er lagt opp for å kunne ta inn input av brukers verdier. Her finnes 11 ulike inputfelt, hvorav fire er grønne ferskvannsparametre, fire oransje sjøvannsparametre, samt tre "konstante" avanserte verdier
- **Sluttberegning billigste –** her blir de 10 billigste kjølere som oppfyller kravene for kjøling av ferskvann hentet fra sortert svartabell. Tidligere er det ikke blitt tatt hensyn til maksimal temperatur for sjøvann ut, så det blir derfor her regnet på nytt med henhold til det.
- **Svartabell -** Her blir alle løsninger vurdert opp mot minimum- og maksisums-kriterier
- **Sortert svartabell –** Sorterer de godkjente kjølerne etter pris – uten hensyn til maks sjøvannstemperatur.
- **Kalkulasjon inn-verdier –** verdiene i inputfeltene fra Web-interface blir hentet inn her, og det siste og gjenværende inputfeltet blir utregnet.
- **Regneark DN65-80 –** regneark som gjelder for varmevekslerstørrelse DN65-80. Her blir aktuelle beregnede svar sendt videre til svartabell, som videre fortsetter kalkuleringer
- **Regneark DN100-125 –** samme som forrige
- **Regneark DN150-200 –** samme som forrige
- **Sjøvannsegenskaper -** tabeller med data om sjøvannsegenskaper (varmekapasitet, ledningsevne, densitet osv.)
- **Ferskvannsegenskaper -** tabeller med samme data som sjøvannsegenskapene, bare for ferskvann

10	Flow, Fresh water (hot fluid)	m^3/h
0	Heat transfer	kW
55	In-temperature, Fresh water (hot, before cooling)	°C
46	Out-temperature, Fresh water (hot, but cooled)	°C
32	In-temperature, Sea water (cold)	°C
0	Weight percent antifrogen (ethylene glycol)	%
0	Bio-fouling factor	%
3,5	Salinity, Sea water	%
1,5	Max pressure drop, Fresh water	bar
2	Max pressure drop, Sea water	bar
90	Max temp out, sea water	°C

Figur 3.1 Inputfeltene i "web-interface"-arket i Excel-fil

3.3.1 Kalkulasjonsprosess

Grønne inn-verdier fylles inn ved at data blir hentet fra webapplikasjonen. Tre av disse må fylles ut. Man kan ikke fylle ut alle fire, for da blir beregningen overbestemt. Gule inn-verdier er allerede foreslått, men ikke nødvendig. Oransje inn-verdier er også foreslått, men disse kan overskrives ved behov.

Her tenkes det å legge inn JavaScript-kode i applikasjonen som begrenser hvor mange felt som kan skrives inn i før kalkulering er mulig, for å forhindre feilberegning.

Når "Calculate"-kommandoen gis i applikasjon, skal verdiene som bruker sender inn videresendes inn i tilsvarende felt i de tre regnearkene for de tre hovedstørrelsene (DN65/80, DN100/125, DN150/200). Det er foreløpig bare DN100/125 som er tilgjengelig i nær framtid, men de andre er det mulig at kommer på et senere tidspunkt.

For at verdiene skal bli hentet fra web-applikasjonen inn i arket, tenkes det å bruke Apache POI for å manipulere de spesifikke cellene med verdier en kan se i figur 3.1, og deretter lese av resultatet som vises i tabellen (figur 3.2).

De tre innsendte grønne verdiene går via arket "Kalkulasjon inn-verdier", der den manglende verdien finnes. Dersom umulig kombinasjon er gitt av de 3 andre variablene, vil cellerute A1 vise "FEIL", og ytterligere informasjon om "koking" eller "frysing" gis i celle B1. Det finnes da ingen gyldige løsninger.

Regnearkene henter fysikalske data fra egne ark, og universelle data fra eget ark (Universelle data). Det beregnes deretter svar for samtlige elementkonfigurasjoner, 1-pass & 2-pass, via 2 iterasjoner, og svarene sendes til "Svartabell". Der vurderes alle løsningene mot satte min/maks-kriterier, og de løsninger som godtas, listes først på "Sortert svartabell", og deretter på Web ut-verdier. Herfra vil resultatene leses fra applikasjon.

Description	First calculation	Second calculation	Tørrvekt (kg)	Vekt operativ (kg)	Sice class	Passes	Elements per pass	FW flow (m^3/h)
DN65/80, 1-pass with 1 element. Recommended	OK	FEIL	165	175	1	1	1	10,0
DN65/80, 2 serial passes with 1 element each pa	OK	FEIL	188	208	1	2	1	10,0
DN65/80, 3 serial passes with 1 element each pa	OK	FEIL	211	241	1	3	1	10,0
DN65/80, 4 serial passes with 1 element each pa	OK	FEIL	234	274	1	4	1	10,0

Heat effect (kW)	FV temp inn (°C)	FV temp ut (°C)	K-value clean (W/m^2K)	K-value service (W/m^2K)	SW flow (m^3/h)	SV temp ut (°C)	ΔP FV (bar)	ΔP SV (bar)	Recommended port size	Titan area (m^2)
103	55,0	46,0	4501	4501	7,9	44	0,10	0,08	DN65	1,81
103	55,0	46,0	3806	3806	4,7	52	0,20	0,06	DN65	3,62
103	55,0	46,0	3666	3666	4,2	54	0,29	0,07	DN65	5,43
103	55,0	46,0	3623	3623	4,0	55	0,39	0,09	DN65	7,25

Figur 3.2 Resultattabell i Excel-fil

Pris vil bli luket ut i applikasjonen, da dette er konfidensiell informasjon.

3.3.2 Samhandling med applikasjon

En mulig konsekvens som ofte har blitt diskutert med tanke på bruk av Excel-filen i bakgrunnen av applikasjonen, er at det kan skape problemer i ettertid om bedrift vil legge til eller modifisere produkt eller kalkulasjoner.

For at dette skulle forhindres, har excel-filen blitt justert på den måte at regnearket "Web-interface" bare skal kunne brukes i sammenheng med applikasjonen. Derfor kan alle de resterende regnearkene endres og videreutvikles uten konflikt med applikasjonen. Dette avhenger også av at "SORTER"-funksjonen ikke blir brukt, da Apache POI ikke støtter denne funksjonen.

Kapittel 4: Resultater

I dette kapittelet, vil det presenteres resultater og implementasjon av applikasjonen (som er navngitt "Pleat Configurator") i henhold til problemstillingen som ble gitt. Innholdet i kapittelet vil bli knyttet sammen med bakgrunnen presentert i kapittel 2 og 3. For at dette skal kunne presenteres på best mulig måte, skal det tas en gjennomgang av hver enkelt komponent. Der vil det vises litt diverse bakgrunnsinformasjon, blant annet forklaringer av funksjonalitetene og begrunnelse for hver av valgene.

4.1 Vellykka prosjekt

Når det kommer til selve resultatet av oppgaven, må det sies at samtlige gruppemedlemmer og oppdragsgivere ser seg selv fornøyde. Vi sitter igjen med en webapplikasjon som fyller de kravene som ble satt fra start, og den har potensiale til å bli en daglig brukt applikasjon av både oppdragsgivers salgsavdeling og potensielle kunder både på landsbasis og internasjonalt.

Gruppen har vurdert ulike alternativer, og mange forskjellige faktorer har vært i spill. Hovedfaktorer består av firmaet sine krav for prosjektet, gruppens tidligere erfaring og kompetanse, samt hva som er mest hensiktsmessig for oppgaven som er valgt. Det er blitt tatt viktige beslutninger, og sett i ettertid er valgene og sluttresultatet meget godkjent når en sammenligner det med planen i startsfasen.

4.1.1 Fornøyd oppdragsgiver

Det har blitt holdt et godt samarbeid med oppdragsgiver helt fra start. De har vært veldig tydelige på hvilke krav og ønsker de har til resultatet, og gruppen har jobbet hardt for å innfri på alle punkter. Oppdragsgiver har jevnlig gitt uttrykk for at de er fornøyde med arbeidet, og har konkludert med at gruppen har levert et godt stykke arbeid. Her har de spesifikt nevnt at de er imponert over både arbeidsinnsats og sluttprodukt, samt at de er glade for å ha mottatt tilnærmet det produktet de hadde sett for seg fra start. (Viser til attest fra bedrift i vedlegg **D)**

4.1.2 Verdi for brukere

Produktkonfiguratoren har en enorm verdi for brukerne. Den gir den brukerne en enkel måte å tilpasse og konfigurere produkter på. Løsningen vil være en ny funksjon på Hydroniq sin nettside, som etter hvert kan resultere i økt interesse på flere områder.

Konfiguratoren fjerner behovet for lengre konfigurasjonsprosesser og usikkerheter, både for salgsavdeling og kunder. Det vil spare enormt med tid, og til syvende og sist føre til en mer behagelig brukeropplevelse. Den har også fordeler for bedriftens salgsavdeling. Mange av de tidligere manuelle prosessene vil bli erstattet, slik at det blir lettere å lage tilbud. Dette gjør salgsprosessen mye raskere enn tidligere, som vil gi salgsavdelingen mer tid til å fokusere på andre ting som kundeoppfølging og salgsstrategi.

Kunde vil også få en slags visualisering på produktet de er interessert i, slik at en kan få en idé om hvor godt dette vil passe inn i deres situasjon. De vil også ha muligheten til å sette ulike muligheter opp imot hverandre, hvor de selv kan se hvilke av produktene som er mest relevant.

4.2 Applikasjonen

I dette delkapittelet skal det bli diskutert litt mer om selve applikasjonen – og hvordan den er oppbygd. Det vil bli tatt for seg litt mer om hovedkomponentene programmet består av, altså de løsningene som til sammen får alt til å fungere. Disse er alle sammen brukt for at det til slutt skulle ende opp med ønsket resultat. Mye av teknologien som er

blitt brukt her er allerede blitt nevnt i tidligere kapitler, så det vil her bli tatt en gjennomgang som er mer resultatorientert.

4.2.1 Modellklasser

Pleat

Pleat-modellklassen er en klasse som representerer varmevekslerproduktet med de ulike egenskapene til en kjøler. Her finner vi 18 forskjellige attributter som brukes til å beskrive kjøleren. Dette er parameterne som skal vises for hvert Pleat-produkt i resultattabellen. Dette består av blant annet:

- **Beskrivelse** – en generell beskrivelse av produktmodellen
- **Vekt** – vekt i kg av produktet
- **Størrelse** - Areal (m²) av selve produktet
- **Element** – Antall element produktet vil bestå av
- **Effekt** – Varmeeffekten til kjøleren
- **Temperaturer** – Temperaturverdier inn og ut på både ferskvanns –og sjøvannssiden av kjøleren, samt makstemperatur for sjøvann.
- **Trykkfall** – trykkfallet i både sjøvann og ferskvann, og maksimumskriterier.

Det er også lagt opp «getters» og «setters» for alle parametere, samt en konstruktør som initialiserer klassen. Dette blir også laget et sett med settere for "Builder", som er et krav for at "builder"-mønsteret skal fungere. Til slutt, har klassen noen metoder som sørger for at verdiene ikke er negative, at de ikke tomme, og at de ikke er NULL.

Eksempelkode:

Denne metoden sjekker om parameterne til Pleat-klassen ikke har negative verdier

```
private static void checkIfNumberNotNegative(int object, String errorMessage) {
    if (object < 0) {
        throw new IllegalArgumentException("The " + errorMessage + "Cannot be negative values");
    }
}
```

Vi har fulgt Builder-mønsteret (se delkapittel 2.3.2) med denne klassen slik at skapelse av et nytt objekt av denne typen kan bruke "builder"-metoder. Hovedgrunnen til bruk av Builder-mønsteret for denne modellen er på grunn av at Pleat-modellen er en klasse med mange felt, og det vil gjøre koden mer lesbar og ryddig. Se eksempler i de to kodesnuttene nedenfor.

```
public Builder setDescription(String description) {
    checkString(description, "description");
    this.description = description;
    return this;
}

public Builder setDryWeight(int dryWeight) {
    checkIfNumberNotNegative(dryWeight, "dry weight");
    this.dryWeight = dryWeight;
    return this;
}

public Builder setOperativeWeight(int operativeWeight) {
    checkIfNumberNotNegative(operativeWeight, "operative weight");
    this.operativeWeight = operativeWeight;
    return this;
}
```

```
public Pleat build() {
    Pleat pleat = new Pleat();
    pleat.setDescription(description);
    pleat.setDryWeight(dryWeight);
    pleat.setOperativeWeight(operativeWeight);
    pleat.setSizeClass(sizeClass);
    pleat.setPasses(passes);
    pleat.setTotalNumberOfElements(totalNumberOfElements);
    pleat.setFwFlow(fwFlow);
    pleat.setHeatEffect(heatEffect);
    pleat.setFwTempIn(fwTempIn);
    pleat.setFwTempOut(fwTempOut);
    pleat.setkValueClean(kValueClean);
    pleat.setkValueService(kValueService);
    pleat.setSwFlow(swFlow);
    pleat.setSwTempOut(swTempOut);
    pleat.setFwDeltaP(fwDeltaP);
    pleat.setSwDeltaP(swDeltaP);
    pleat.setRecommendedPortSize(recommendedPortSize);
    pleat.setTitanAreaM2(titanAreaM2);
    pleat.setUnits(units);
    return pleat;
}
```

ExcelInput

ExcelInput-modellklassen er en klasse som representerer selve inputfeltene inne i Excel-filen. Disse parameternes verdier er de som skal bli kalkulert, og som etter hvert resulterer i en tabell av produkter. Klassen inneholder parametere som f.eks:

- **Effekt –** Representerer den nødvendige kjøleeffekten for en varmeveksler.
- **Flow –** Representerer strømningshastigheten til vann inni rørsystemer.
- **Temperatur inn –** Representerer sjøvann eller ferskvann som kommer inn i kjøleren fra enheten som skal bli kvitt overskuddsvarme. For eksempel en motor eller et HAVC system.
- **Temperatur ut –** Ferskvannslinje som går ut fra varmeveksler, er ei lukket linje som ligger nært inn linjen til kjøleren for å ta imot varmen som skal overføres. Er ei linje som alltids er lavere enn temperatur inn for å få kjøleeffekten.
- **Glyokolprosent –** Har med hvor stor konsentrasjonen av glykol, som brukes i kjølesystemer som varmeoverføringsvæske.
- **Bio-fouling prosent –** Refererer til uønskede materialer på overflaten av kjølerens plater, noe som kan redusere effektiviteten. Disse materialene som legger seg på overflaten er ofte planter, alger eller små dyr på våte flater.
- **Saltinnhold –** Saltinnhold, også kjent som salinitet, er en viktig faktor for platekjølere og andre varmevekslere som bruker sjøvann. Dette skyldes faktorer som Korrosjon, varmeledningsevne og fouling.
- **Units –** Dette feltet inneholder et Units-objekt som er måleenheten for dataene i skjemaet, brukt til å endre enheter. Kan for eksempel endre fra liter i sekund (l/s) til kubikkmeter i sekund (m^3/h), eller lignende for temperatur.

Klassen bruker "builder"-mønster (se forrige punkt) til å lage en instans av ExcelInput. Det ble brukt dette designermønsteret siden denne klassen er kompleks og tar inn mange parametere som skal være valgfri, derfor bygger vi objektet med en "builder".

Hovedmålet med denne klassen er å legge inn parameterne for en varmeveksler på en god og strukturert måte. Disse parameterne brukes i beregninger eller simuleringer for å gi oppførselen til et varmevekslersystem.

PleatForm

PleatForm-modellklassen er en klasse som representerer verdiene som blir fylt inn av bruker i konfiguratoren. Disse parameternes verdier blir sendt til back-end for videre kalkulering. Klassen inneholder identiske parametere som **ExcelInput**, da de representerer de samme egenskapene.

Klassen har vanlige "getter"- og- "setter"-metoder for hver av feltene, dette gjøre det mulig for innkapsling av data og overholder prinsippet for OOP (se 2.3.1)

Denne klassen har en interessant metode som heter "**convertToExcelInput**", som samler inn data fra feltene i klassen for å konvertere skjemaet til et ExcelInput-objekt. Dette brukes til å legge data inn i et Excel-ark for videre behandling.

Til slutt er det lagt til en **toString**-metode som gir en strengrepresentasjon av PleatForm-objekter, noe som har blitt brukt for å logge og feilsøking av prosjektet.

Units

Units-modellklassen er en klasse som representerer enhetene som blir brukt for temperatur- og flow-verdiene i konfiguratoren. De inneholder den valgte enheten for hver bruker. Klassen inneholder parameterne **tempUnit** og **flowUnit**, som er henholdsvis for temperatur og mengde. Temperatur er enten celsius eller fahrenheit. 'Mengde' er enten m^3/h eller l/s.

4.2.2 Kontrollere

Kontrollerklassene er klasser som håndterer forespørsler slik at programmer kommuniserer på ønsket vis. De kontrollerer hvordan data blir sendt mellom de ulike delene av applikasjonen, og vil ha mulighet til å sette opp kommunikasjon mellom back-end og front-end. Det er enkelt forklart som en mellommann som utfører oppgavene som trengs for at du skal få tilsendt ønsket resultat.

IndexController

I applikasjonen er det en hovedkontroller kalt '**IndexController**' som er ansvarlig for å behandle de viktigste forespørsler. Dette fungerer ved at den sender brukers input inn i Excel-fila, henter oppdatert data fra Excel-fila ved hjelp av '**ExcelService**' og "**ExcelServiceImplementation**"-klassene (som vi tar en mer detaljert titt på senere), og sender resultatene tilbake til brukergrensesnittet.

De viktigste metodene av **IndexController** er følgende:

- **getIndex**(): Denne metoden håndterer GET-forespørsler til path ("/") og returnerer index.html-filen. Den legger også til et nytt **PleatForm**-objekt til modellen, som vil bli brukt til å samle inn data fra brukeren gjennom feltene på nettsiden.
- **handleFormSubmission**(): Denne metoden håndterer POST-forespørsler når brukeren sender inn skjemaet med **Pleat**-data, enten fra ferskvanns-konfigurator eller avansert-konfigurator. Den mottar en rekke parametere, inkludert en boolean som angir om skjemaet ble sendt inn fra "advanced config", et **PleatForm**-objekt som inneholder Pleat-dataene som ble sendt inn i skjemaet, og et Spring Model-objekt for å legge attributter til grensesnitt.

Kodeeksempel:

```
@PostMapping(value = "/")
public String handleFormSubmission(@RequestParam(value = "isAdvanced") boolean
isAdvanced,
                           @ModelAttribute PleatForm pleatForm,
                           Model model) {

    // Handle the form submission here
    Map<String, Object> result = new HashMap<>();

    log.info("isAdvanced = {}", isAdvanced);

    if (isAdvanced) {
        result.put("message", "Advanced form submitted successfully!");
    } else {
        result.put("message", "Basic form submitted successfully!");
    }

    // Adding the list of pleats to the response
    List<Pleat> pleats = new ArrayList<>();

    // Retrieving table list
    try {
        pleats = excelService.getResultList(pleatForm.convertToExcelInput());
    } catch (NumberFormatException numberFormatException) {
        //numberFormatException happens when we try to read an unexpected value from
excel
        log.warn("NumberFormatException {}", numberFormatException.getMessage());
    }

    // Adding the list of pleats to the response
    result.put("pleats", pleats);
    result.put("numberOfProducts", pleats.size());
    // We resend the pleatForm so input values is kept
    model.addAttribute("pleatForm", pleatForm);
    model.addAttribute("result", result);
    model.addAttribute("displayAdvanced", isAdvanced);
    return "index";
}
```

Når denne metoden kalles, vil den først logge om bruker brukte advanced-konfiguratoren eller ikke. Deretter vil den legge til en melding i resultatet basert på hvilken konfigurator som ble brukt. Metoden vil også hente resultatlisten med Pleats ved å kalle excelService.**getResultList**(...), og legge den til i resultatet. Til slutt vil metoden legge til resultatet og andre attributter i modellen, og returnere visningen "index" for å vise responsen.

Gjennom denne prosessen kan **IndexController** effektivt håndtere forespørsler, utføre nødvendige beregninger ved hjelp av **ExcelService** og **ExcelServiceImplementation**,

og vise resultatene til brukeren på en strukturert og forståelig måte. Denne klassen er derfor en kritisk komponent i vår web-applikasjon, og den demonstrerer effektiv bruk av kontrollere for å håndtere interaksjoner mellom back-end og front-end

PDFController

PDFController-klassen er en annen viktig komponent i vår web-applikasjon, som håndterer forespørsler relatert til PDF-filer. Denne klassen bruker Spring-annotasjonen, og er ansvarlig for å håndtere PDF-forespørsler og returnere de riktige PDF-filene. PDF-filene består av alle de ulike GA-tegningene til de forskjellige Pleat-produktene.

Kodeeksempel

```
@GetMapping("/{pdfName}")
public ResponseEntity<?> downloadPdf(@PathVariable String pdfName) {

    //Retrieving the pdf file
    String pdfPath = oneDrivePathServer + File.separator + "pdf" + File.separator + "GA-tegninger" + File.separator + pdfName + ".pdf";
    File file = new File(pdfPath);

    if (!file.exists()) {
        // creating a new file using system property
        file = new File(systemOneDrivePath + File.separator + "pdf" + File.separator + "GA-tegninger" + File.separator + pdfName + ".pdf");

        if (!file.exists()) {
            // Still not found the file then respond with notfound page
            return ResponseEntity.notFound().build();
        } else {

            // The file dose not exits, try with system property
            return getFileResponseEntity(pdfName, file);
        }
    } else {
        return getFileResponseEntity(pdfName, file);
    }
}
```

Metoden er merket med @**GetMapping**-annotasjonen, noe som indikerer at den håndterer HTTP GET-forespørsler. I dette tilfellet vil den håndtere alle forespørsler til "/pdf/{pdfName}", hvor "{pdfName}" er en variabel som representerer navnet på PDF-filen.

Inne i metoden blir PDF-filen hentet ved hjelp av **ResourceLoader**. Deretter prøver metoden å få URL-en til PDF-filen. Hvis det oppstår en **IOException**, vil metoden returnere en **ResponseEntity** med en HTTP-statuskode **"Not Found" (404)**. Dette kan skje hvis PDF-filen ikke finnes på den angitte plasseringen. Til slutt setter metoden opp nødvendige HTTP-headers for å returnere PDF-filen, og sender den tilbake som respons.

CalculationsController

CalculationsController er en sentral klasse i programmet, designet som en Spring **MVC**-kontroller (se kapittel 2.3.3) for å håndtere HTTP POST-forespørsler til "/calculations". Denne kontrolleren har ansvaret for å generere en PDF som inneholder beregningsresultater basert på data mottatt i forespørselen.

Når en forespørsel ankommer, bruker CalculationsController et **JSONObject** som en forespørselsparameter. Dette JSON-objektet er forventet å inneholde data som representerer en Pleat-instans. Ved hjelp av **ObjectMapper** fra **Jackson**-biblioteket, forsøker kontrolleren å konvertere JSON-strengen til en Pleat-instans. Dersom det oppstår en feil under konverteringsprosessen, for eksempel hvis JSON-objektet mangler nødvendige data, logges feilen og kontrolleren returnerer en HTTP 500 (Internal Server Error) respons.

Ved vellykket konvertering legger CalculationsController Pleat-instansen til modellen under attributtnavnet "calculations". Denne instansen benyttes deretter av en **TemplateEngine** for å generere HTML-innhold basert på en Thymeleaf-mal kalt "calculated-model". Denne malen er designet for å bruke dataene i modellen for å generere en HTML-representasjon av beregningsresultatene. Hvis det oppstår en feil under denne prosessen, blir feilen logget og en HTTP 500 respons returneres.

HTML-innholdet som er generert av Thymeleaf-malen, blir deretter konvertert til en PDF ved hjelp av **Flying Saucer**-biblioteket. PDF-resultatet lagres i en **ByteArrayOutputStream**. Hvis det oppstår en feil under denne prosessen, logges feilen og en HTTP 500 respons returneres.

Til slutt setter CalculationsController respons-headere for å indikere at responsen er en PDF. PDF-innholdet blir sendt som responskroppen. På denne måten tar CalculationsController inn data om en Pleat-instans og returnerer en PDF med beregningsresultatene basert på den mottatte dataen fra bruker.

4.2.3 Sikkerhet

Nettapplikasjonen krever ikke brukerautorisasjon på grunn av virkemåte som en offentlig nettside med tilhørende endepunkter. Siden nettsiden er laget for offentligheten, er det ingen krav til brukerspesifikk autorisasjon.

Feilhåndtering er tatt i brukt i for å få en mer robust og stabil applikasjon for å unngå potensielle sikkerhetssårbarheter. Med håndtering av en feil på riktig måte kan vi forhindre sensitiv informasjon til å bli eksponert, redusere risikoen for angrep og gi en bedre brukeropplevelse. Under er et eksempel hvor applikasjonen håndterer at en fil ikke ble funnet med en "Ikke funnet"-nettside og den riktige HTTP status-koden 404.

```
if (!file.exists()) {
    // Still not found the file then respond with notfound page
    return ResponseEntity.notFound().build();
}
```

Eksempel under er feilhåndtering som sikrer at applikasjonen kan håndtere uventede scenarier som er med å redusere risikoen for krasj eller sikkerhetssårbarhet.

```
try {
    pleats = excelService.getResultList(pleatForm.convertToExcelInput());
} catch (NumberFormatException numberFormatException) {
    //numberFormatException happens when we try to read an
    //unexpected value from Excel
    log.warn("NumberFormatException{}",numberFormatException.getMessage());
}
```

4.3 Integrasjon med Excel-fil

For å håndtere interaksjonen mellom applikasjonen og Excel-filen, er det implementert flere klasser og grensesnitt som sammen oppnår ønsket effekt. Klassene nedenfor inneholder kode som fungerer sømløst sammen med det formål å lese/skrive data i Excel-filer samt produsere resultater i henhold til spesifiserte kriterier.

4.3.1 Klasser og verktøy

ExcelService

Her defineres metoden getResultList, som tar modellklassen ExcelInput som parameter og returnerer en liste av Pleat-objekter (se 4.1.1). ExcelService er implementert i ExcelServiceImplementation-klassen.

ExcelServiceImplementation

Dette er en klasse som er en del av tjenestelaget i Spring-applikasjonen og er designet for å ta seg av operasjoner relatert til Excel-filer. Klassen implementerer grensesnittet **ExcelService,** og inneholder metoder for å lese, skrive og oppdatere data fra en Excel-fil. Denne klassen har også flere metoder som konverterer temperatur- og strømningsverdier tilbake til deres opprinnelige enheter.

Metoden **readExcelWeb** leser spesifikk celledata fra «Web Interface» arket i Excel-filen, konverterer det til Pleat-objekter og returnerer en liste av Pleat-objekter. Denne metoden tar hånd om ulike Excel-celletyper og evaluerer formler før den henter ut verdien. Pleat-objektet som returneres og genereres blir sortert etter attributtet "titanAreaM2", som vil si arealet av materialet som er brukt til å produsere varmeveksleren.

Metoden **writeExcel** skriver data fra en ExcelInput-objekt til en Excel-fil. Det blir brukt for å oppdatere og legge til nye data i Excel-filen. Denne metoden kaster IOException hvis det oppstår en I/O-feil mens en leser eller skriver til filen.

Datavalidering er en instans av "**ExcelValidator**" som er brukt til å validere Excel-filen slik at filen kun tas i bruk om den er godtatt før noen operasjoner utføres på filen.

ExcelValidator

Denne klassen inneholder metoder for å validere Excel-filens eksistens og tilgjengelighet, og brukes i ExcelServiceImplementation-klassen for å sjekke om filen er tilgjengelig før den prøver å lese eller skrive data fra den.

Klassen har to sentrale metoder:

- **isExcelFileValid**(String filePath) - Denne metoden tar filplasseringen som input og sjekker om Excel-filen eksisterer på den angitte plasseringen. Dette er gjort ved å lage en ny File-instans, for så kaller exists()-metoden. Dette vil returnere true hvis filen eksisterer, og false hvis ikke. Hvis programmet ikke finner frem til Excel-filen, oppretter metoden en tekstfil ("*feedback.txt*") i samme plassering som den manglende Excel-filen skulle ha vært. Denne filen vil inneholde en feilmelding som informerer brukeren om at den angitte Excel-filen ikke ble funnet. Hvis det oppstår en IO-feil mens tilbakemeldingsfilen opprettes, logges en advarsel.

- **getPath**(Resource excelResource) - Denne metoden tar en Resource-instans som representerer Excel-filen som input, og prøver å hente filbanen til denne ressursen ved å kalle getURL().getPath(). Hvis det oppstår en IO-feil i prosessen, logges en advarsel, og metoden returnerer en streng som sier "Path not found".

ConversionUtils

ConversionUtils er en verktøy-klasse som inneholder statiske metoder for å konvertere forskjellige måleenheter som er relevante for programmet. Disse metodene er designet for å være enkle å bruke, da de gjenbrukes i flere deler av programmet. Alle metodene er statiske, noe som betyr at de kan kalles uten å opprette en instans av ConversionUtils-klassen.

Klassen består av fire metoder som er brukt i delene av applikasjonen hvor det trengs enhetskonvertering:

lsToM3h(double ls): Denne metoden konverterer flythastighet fra liter per sekund (ls) til kubikkmeter per time (m^3/h). Dette gjøres ved å multiplisere inngangsverdien med 1000 (for å konvertere liter til kubikkmeter) og deretter dividere med 3600 (for å konvertere sekunder til timer).

m3hToLs(double flowRateM3h): Denne metoden konverterer verdien i Flow-variabelen fra kubikkmeter per time (m^3/h) til liter per sekund (l/s). Her blir inngangsverdien multiplisert med 3.6, noe som reflekterer det motsatte av lsToM3h-metoden.

fahrenheitToCelsius(double fahrenheit): Denne metoden konverterer temperatur fra Fahrenheit til Celsius ved hjelp av formelen **(Fahrenheit - 32) * 5/9**.

celsiusToFahrenheit(double celsius): Tilsvarende konverterer denne metoden temperatur fra Celsius til Fahrenheit ved hjelp av formelen **Celsius * 9/5 + 32**.

Bruk av Apache POI-verktøy

Apache POI-biblioteket brukes for å lese, skrive og oppdatere Excel-filen i denne koden. Det brukes i ExcelServiceImplementation-klassen for å håndtere interaksjonen med Excel-filen.

Fra dette biblioteket er det flere viktige verktøy som er tatt i bruk for å kunne utføre datamanipulasjon:

XSSFSheet – er et objekt som representerer et ark i en XLSX-fil, og inneholder metoder for å jobbe med rader, kolonner og celler i arket. XSSFSheet brukes til å navigere og manipulere dataene i Excel-filen.

XSSFCell – er et objekt som representerer en celle i en XLSX-fil. Denne måtte brukes for å få manipulert de nødvendige cellene.

setCellValue – er en metode som brukes for å sette verdien av den valgte cellen i et gitt ark (Web-interface i vårt tilfelle). Den tar en verdi som parameter og skriver den til den aktuelle cellen.

ForceFormulaRecalculation – Dette er en metode som tvinger Excel-filen til å oppdatere alle formlene i den. Dette sikrer at alle formler er korrekt evaluert og gir de riktige resultatene når applikasjonen leser data fra filen.

FormulaEvaluator evaluateFormulaCell– Denne klassen med metoden er ansvarlig for å evaluere formler i en valgt celle. Den sørger for at alle formler er korrekt evaluert og returnerer de riktige verdiene før applikasjonen leser eller skriver data fra eller til filen. Denne brukes for å få verdiene av formelen i stedet for selve formelen.

getRow – Dette er en metode som brukes for å hente en rad i et gitt ark. Den returnerer et objekt av typen XSSFRow, som inneholder informasjon om raden og alle cellene i den.

getCell – Dette er en metode som brukes for å hente en celle i et gitt ark. Den returnerer et objekt av typen XSSFCell, som inneholder informasjon om raden og alle cellene i den.

getSheetAt – Denne metoden brukes for å hente et spesifikt regneark fra en XLSX-fil. Her henter vi regneark med indeks 0, da regnearket som skal brukes er det første i filen.

Kodeeksempel:

```
XSSFWorkbook workbook = new XSSFWorkbook(inputStream);
workbook.setForceFormulaRecalculation(true);
XSSFSheet sheet = workbook.getSheetAt(0);

XSSFCell flow = sheet.getRow(1).getCell(1);
flow.setCellValue(excelInput.getFlow());
```

I kodeeksempelet over, kan man se hvordan verktøyene har blitt tatt i bruk i applikasjonen. Først ble det laget et **XSSFWorkbook**-objekt fra en **inputStream** som tar inn bedriftens XLSX-fil.

Etter dette, blir «**ForceFormulaRecalculation**» satt til true, slik at Excel blir tvunget til å beregne regnearket hver gang filen blir brukt. Uten denne, vil ikke kalkulasjonene utføres ved lagring, som kan føre til gamle verdier i resultattabellen.

Deretter hentes **XSSFSheet**-objektet (regnearket) ved å bruke **getSheetAt**-metoden. Så blir verdien i cellen på rad 1, kolonne 1 satt til verdien som blir hentet fra **flow**-variabelen i **ExcelInput**.

4.3.2 Forhindring av konflikter

Samtidighet

En excel-fil har ikke mulighet for at flere manipulerer den samtidig. For å forhindre kollisjon mellom flere brukere, er det tatt i bruk en «**synchronized**»-blokk for å forhindre samtidighet. Vi tar i bruk et låseobjekt (**fileLock** i kodeeksempel) som brukes til å kontrollere tilgangen til Excel-filen blant brukere.

Kodeeksempel:

```
private final Object fileLock = new Object();
private List<Pleat> getPleats(ExcelInput excelInput) {
   try {
      synchronized (fileLock) {
         writeExcel(excelInput);
         List<Pleat> pleats = readExcelWeb(excelInput);
         log.info("Retrieved data from excel file.");
         return pleats;
      }
   } catch (IOException e) {
      throw new RuntimeException(e);
   }
}
```

En tråd vil prøve å skaffe seg låsen til «fileLock»-objektet. Om ingen andre tråder har låsen, vil tråden få låsen og kunne utføre den koden som eksisterer inne i synchronized-blokken. Om låsen allerede er opptatt, vil tråden som forsøker å skaffe seg låsen blokkeres, og må vente til låsen blir frigjort etter tråden har utført koden.

Denne mekanismen sikrer at kun én bruker i applikasjonen om gangen kan få tilgang til og manipulere Excel-filen, og forhindrer dermed samtidig tilgang. Sannsynligheten for at noen trykker på «calculate»-knappen helt samtidig er forholdsvis lav, men det er uansett et viktig tiltak for å forhindre potensielle konflikter og feildata.

OneDrive

For at firma skal kunne videreutvikle excel-filen som er tatt i bruk i applikasjonen, ble det bestemt å plassere filen på en OneDrive-server med et permanent filnavn. Dersom firma skal gjøre endringer, må de ta kopi av arket, utføre nødvendige endringer, lagre fil med det predefinerte filnavnet, og først da kan filen i OneDrive erstattes. Så lenge «web-interface» arket i filen ikke er endret, vil dette ha null innvirkning på applikasjonen overhodet.

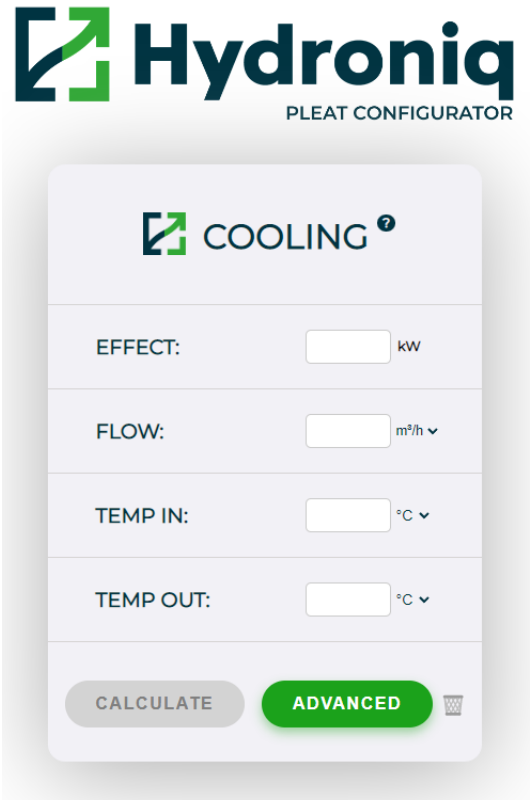
4.4 Grensenitt og funksjonalitet

For utvikling av brukergrensensittet i store deler av prosessen, ble siste versjon av konseptskissen (se 3.1.6) brukt som mal. Gruppen programmerte og utviklet første versjon av siden i henhold til denne skissen. Det ferdige produktet har mange fellestrekk med denne, som for eksempel plassering av inputfelt, farge og utseende, samt tabellen med resultater. Det er naturlig at det er blitt gjort endringer fra den originale konseptskissen, og det har ofte vært et resultat av oppdragsgivers tilbakemeldinger. Et eksempel som kan tas, er at hele «trinn 2» (3.1.6) er blitt fjernet, og innholdet i resultattabellen har fått nytt design. Sammen med bedrift, ble det fra start kommunisert frem og tilbake for å skape den mest brukervennlige og intuitive løsningen.

4.4.1 Basic konfigurator

Når applikasjonen lastes inn, blir man møtt av den «simple» versjonen av produktkonfiguratoren. Som navnet tilsier, er dette en simpel og brukervennlig versjon av produktkonfiguratoren. Her vil en presenteres med et oversiktlig brukergrensesnitt, hvor det finnes en del ulike funksjonaliteter som vil kunne hjelpe bruker å samhandle med applikasjonen (se figur 4.1).

Nedenfor vil vi ta en gjennomgang av de ulike inputfeltene:

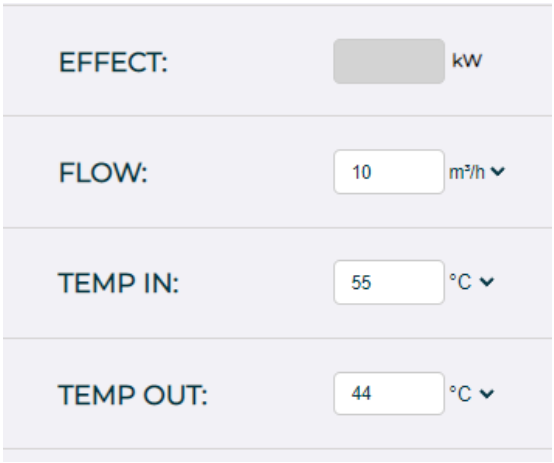


Figur 4.1 Kjølerkonfigurator

Det er fire inputfelt som brukes for å beregne kjøleren. Disse representerer hver av variablene som blir sendt inn i Web-interface-arket i Excel-filen i det brukeren trykker på «Calculate» knappen.

1. **Effect** – Dette er varmeoverføringen mellom kaldt- og varmtvann. Verdien blir oppgitt i kilowatt (kW).
2. **Flow** – Dette er strømmen av ferskvann som går inn og ut av «Pleat»-kjøleren. Her har bruker mulighet til å endre mellom enheter som kubikkmeter i timen (m^3/h) og liter i sekundet (l/s). Se hvordan under enhetskonvertering lenger nede.
3. **Temp in** – Dette er temperaturen på vannet som går inn i kjøleren på ferskvannssiden.
4. **Temp out** – Dette er temperaturen på vannet som går ut av kjøleren på ferskvannssiden.

Konfiguratoren trenger tre av fire felt utfylt for å kunne regne seg frem til et produkt. Når tre felt er fylt inn, vil den siste låse seg slik at det ikke er mulig å fylle den ut (se figur 4.2). Dette er for å sikre at det alltid er en ukjent variabel som konfiguratoren kan løse for. Hvis for eksempel "**Flow**", "**Temp In**" og "**Temp Out**" er fylt ut, vil konfiguratoren beregne "**Effect**" i resultattabell basert på disse verdiene.



Figur 4.2 Tilgang til felt

Om brukeren ønsker å endre hvilken variabel som skal beregnes, er det eneste som trengs å tømme et av de utfylte feltene. Dette vil låse opp det fjerde feltet og tillate brukeren å angi en ny verdi for den variabelen. Dette så vi på som en kritisk funksjon, ettersom at innsending av alle variablene med verdier hadde laget konflikt i Excel-filen. Dermed var dette noe av det første som ble implementert.

Måten dette ble implementert på, var ved hjelp av HTML og JavaScript. Inputfeltene er opprettet som numeriske felt, og enhetsvalgene ble opprettet som nedtrekksmenyer. JavaScript-funksjonene "**onInputEffect**", "**onInputFlow**", "**onInputTempIn**" og "**onInputTempOut**" blir kalt for hvert felt det blir tastet inn verdier, og vil oppdatere tilgjengeligheten for det fjerde inputfeltet dersom tre felt allerede er utfylt.

Vi kan ta onInputEffect som et eksempel:

```
function onInputEffect(input, isAdvanced = false) {
    const inputId = isAdvanced ? 'advanced-effect' : 'effect';
    enforcedInput(input, inputId, 1);

    // Global variable to be used to calculate
    _effect = input;


    // Checks if an input field need to be made editable.
    if (input.toString() === "") {
        _enableInputFields();
    } else {
        // Checks if an input field need to be made to read-only
        _disableInputFiled();
    }
}
```

Denne funksjonen blir kalt hver gang bruker angir eller endrer verdi i Effect-feltet. Den tar inn to parametere, som er **'input'** og **'isAdvanced'**. 'Input' er verdien som bruker angir, mens 'isAdvanced' er en boolean som angir om konfiguratoren som blir brukt er advanced-versjonen eller ikke.

Funksjonen starter med å definere **inputId** som enten '**advanced-effect**' eller **'effect'**, avhengig av om konfiguratoren er i avansert modus eller ikke.
Deretter kaller den funksjonen **enforcedInput** for å validere og begrense brukerens inndata. Denne funksjonen tar tre parametere: inputverdien, id til inputfeltet og minimumsverdien for feltet, som i dette tilfellet er 1.
Den globale variabelen **_effec**t settes deretter til inputverdien. Denne variabelen brukes senere i beregningene som utføres av konfiguratoren.
Etter dette sjekker funksjonen om inputverdien er tom. Hvis det er tilfelle, kaller den funksjonen **_enableInputFields** for å låse opp alle inputfeltene, slik at brukeren kan endre innstillingene. Dette er nyttig hvis brukeren ønsker å endre hvilken variabel som skal beregnes.
Hvis inputverdien ikke er tom, kaller funksjonen **_disableInputField** for å sette et av inputfeltene til "read-only". Dette skjer når brukeren har angitt verdier i tre av de fire inputfeltene. Det fjerde feltet settes til "read-only" for å hindre at brukeren angir en verdi der, da dette ville skapt en ugyldig konfigurasjon (fordi konfiguratoren trenger nøyaktig én ukjent for å utføre beregningene).

Dette gjelder tilsvarende for de andre JavaScript-funksjonene som er blitt brukt til restriksjonen av antall mulige felt.

4.4.2 Generelle funksjoner

Enhetskonvertering

Bedriften var veldig bastant på at enhetskonvertering skulle være en funksjon i konfiguratoren, mye på grunn av at den vil bli brukt internasjonalt. Enhetskonvertering tillater brukere å endre måleenhetene for ulike parametere, inkludert strøm (Flow) og temperatur (Temp In og Temp Out). Dette gir brukere fleksibilitet til å arbeide med de enhetene de er mest kjent med, og forbedrer samtidig brukervennligheten av programvaren.

Når det gjelder flytenheter, kan brukeren velge mellom kubikkmeter per time (m^3/h) og liter per sekund (l/s). For temperatur, kan brukeren velge mellom Celsius (°C) og Fahrenheit (°F).

For hver av disse parameterne er det en tilhørende funksjon som oppdaterer enhetene og konverterer eksisterende verdier til den nye enheten. For eksempel, i **updateFlowUnit**-funksjonen, dersom brukeren endrer enheten fra m^3/h til l/s, konverteres alle eksisterende flytverdier fra m^3/h til l/s ved hjelp av funksjonen m3hToLs. Samtidig, hvis brukeren endrer enheten fra l/s til m^3/h, konverteres alle eksisterende flytverdier fra l/s til m^3/h ved hjelp av funksjonen lsToM3h.

Tilsvarende for temperatur, i **changeTempUnits**-funksjonen, hvis brukeren endrer enheten fra Celsius til Fahrenheit, konverteres alle eksisterende temperaturverdier fra Celsius til Fahrenheit ved hjelp av funksjonen **celsiusToFahrenheit**. Og hvis brukeren endrer enheten fra Fahrenheit til Celsius, konverteres alle eksisterende temperaturverdier fra Fahrenheit til Celsius ved hjelp av funksjonen **fahrenheitToCelsius**.

Disse endringene blir deretter gjenspeilet overalt der enheten vises, inkludert i nedtrekksmenyene og i de skjulte inputfeltene, som lagrer den nåværende enheten for bruk på serveren. Dette sikrer konsistens på tvers av hele brukergrensesnittet.

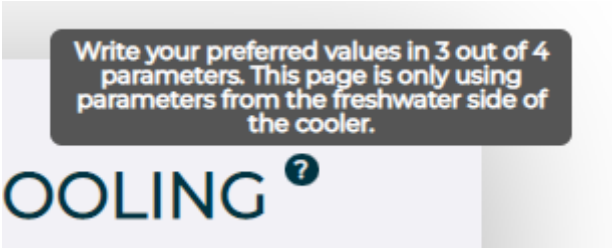
For at dette skal kunne fungere, må enhetene (uavhengig av enhetsvalg) sendes inn som Celsius for temperaturer og kubikkmeter for strømning. Grunnen til dette, er at all data i excel-fila er lagt opp for bruk av disse enhetene, slik at alt annet hadde ført til feildata. Derfor må verdiene konverteres tilbake til disse standardenhetene før de sendes til serveren. Dette gjøres ved hjelp av funksjonen **convertValues**(). Dette sikrer at beregningene utføres med riktige enheter, uavhengig av hvilke enheter brukeren har valgt.

På serversiden blir de konverterte verdiene mottatt og behandlet. I Java-koden brukes funksjonene **convertTemperatureBack**() og **convertFlowBack**() for å konvertere verdiene tilbake til de valgte enhetene før de presenteres for brukeren. Dette fører til at resultatene vil vises i de samme enhetene som brukeren valgte for inputverdiene..

Verktøytips

Ved siden av overskriften i konfiguratoren, er et lite symbol med et spørsmåltegn i seg. Denne er her for å hjelpe brukeren dersom det er usikkerhet om hvordan man bruker applikasjonen. Om brukeren drar musepekeren over dette symbolet, vil det komme opp

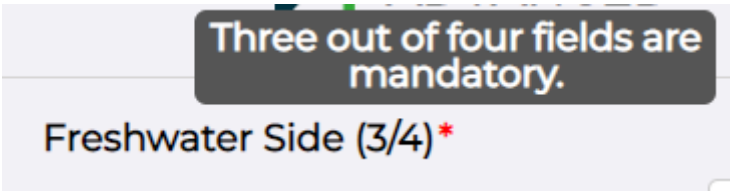
en liten melding der det står «fyll ut tre av fire felt for å bruke konfiguratoren» (på engelsk).
Dette har blitt implementert ved hjelp av HTML og CSS, og har blitt modifisert slik at tipset vil vises umiddelbart i det markør treffer spørsmåltegnet.



Figur 4.3 Verktøytips som beskriver fremgangsmåte til bruker

Hovedårsaken til at denne ble implementert, er for å gjøre applikasjonen så brukervennlig som mulig. Dette vil vise kunde kort og konsist hva de må gjøre for å få muligheten til å kjøre kalkulasjonsfunksjonen. Et verktøytips vil nemlig være til stor hjelp, uten at det vil lage for mye rot.

Det andre viktige symbolet som sender bruker på rett vei, er det røde stjernesymbolet på «advanced»-siden. Denne forteller bruker (i større tekst) at i akkurat denne seksjonen er tre av fire felt obligatoriske.



Figur 4.4 Verktøytips

Calculate-knapp

Denne knappen vil være «låst» når en først laster inn applikasjonen. Når en bruker har lagt inn verdier i tre av fire felt, så vil knappen «åpne» seg. Den vil da være tilgjengelig å trykke på. Når man trykker på knappen så vil applikasjonen sende inn de tre feltene som er fylt ut. Brukeren vil da bli møtt med en «Loading bar» og en tekst der det står «Fetching data», etter et par sekund så vil nettsiden laste seg inn på nytt og vil sende brukeren ned til resultat listen.

Advanced-knapp

Når brukeren trykker på denne knappen så vil den simple konfiguratoren bli byttet ut med den avanserte konfiguratoren. Denne inneholder 11 inputfelt totalt. Forskjeller fra standardkonfiguratoren, er at denne inneholder parametre fra sjøvannssiden. I tillegg, har den parametere kalt "constant values" som helst ikke bør endres, men er mulig dersom nødvendig.

Nullstiller (søppelbøtte)

Denne knappen nullstiller alle inputfeltene i konfiguratoren, og gir tilbake blanke felt. Dette gir brukeren muligheter til å gå tilbake til «blanke ark», slik at brukeren manuelt ikke trenger å hviske ut alle felt. Funskjonen er med på å forbedre brukervennligheten

betraktelig, da den lar brukere raskt og effektivt nullstille alle feltene,uten å måtte gå inn å viske ut verdier fra hvert felt.

Knappen fungerer som en hendelseslytter. I det bruker har lastet inn nettstedet, legger koden til en "klikk"-hendelse til knappen. Dermed vil funksjonen clearFields() bli kalt hver gang knappen blir klikket.

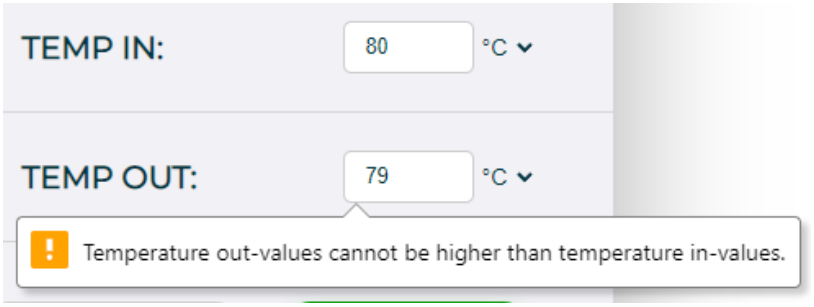
ClearFields()-funksjonen inneholder kode som tilbakestiller inputfeltene. Prosessen starter med at det blir opprettet en liste over feltenes id. Det blir brukt en 'forEach'-loop for å gå gjennom hver id i listen. Her blir de aktuelle feltene hentet, hvor det blir satt en tom streng som elementenes verdier. I tillegg til dette, blir deaktiverte inputfelt reaktivert. Dette gjelder også for 'calculate'-knappen, slik at bruker får muligheten til å starte konfigurering på nytt.

Cooling-knapp

Denne knappen tar brukeren tilbake til den simple konfiguratoren. Dette gjøres så brukerne skal ha 2 muligheter til å utføre konfigureringen av "pleat"-systemer. Det må være enkelt å bytte mellom muligheten slik at kunder får ta i bruk den delen av konfiguratoren som er mest hensiktsmessig for dem. Derfor fungerer "Cooling" knappen som en tilbakeknapp.

Temperaturrestriksjoner

En veldig simpel men nødvendig funksjon er å forhindre "temperatur ut"-verdien fra å være høyere eller lik temperatur inn-verdien. Det har derfor blitt laget en funksjon som hindrer temperatur ut-verdien i å overstige temperatur-inn.

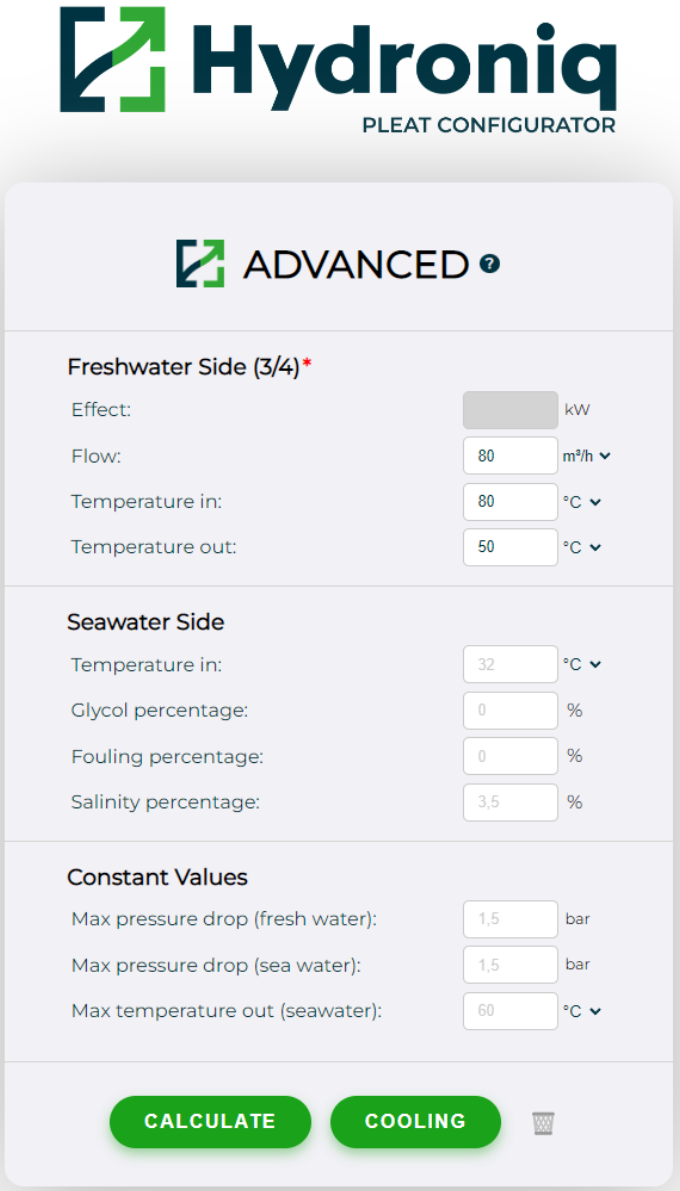


Figur 4.5 Restriksjon av temperatur ut

Dersom bruker prøver å taste inn en verdi i temperatur ut-feltet som er høyere enn det som er tastet inn i temperatur inn-feltet, vil verdien automatisk bli omgjort til tempIn–1. Her vil bruker bli varslet om at dette ikke er mulig (se figur 4.3).

4.4.3 Advanced konfigurator

Når en trykker på «advanced» knappen, blir man møtt av den «avanserte» versjonen av produktkonfiguratoren. Dette en mer avansert versjon ved at bruker får et utvidet brukergrensesnitt med flere inputfelt (se figur 4.2). Her kan brukere legge inn et bredere spekter med data for å få et produkt som er mer tilpasset til deres behov.



Figur 4.6 Avansert konfigurator

Den avanserte versjonen av konfiguratoren deles inn i tre seksjoner. De ulike seksjonene består av ferskvannsside, sjøvannsside og konstante verdier. Vi forsøkte å designe en åpen og lettleselig side med denne struktureringen., og forhåpentligvis vil det gi det en ryddig tilnærming til hvordan bruker vil håndtere informasjonen.

Først skal vi undersøke ferskvannsaspektet nærmere. Dette er delen av konfiguratoren som er ansvarlig for å håndtere data om ferskvann i kjøleren. Disse feltene er de samme som på den simple versjonen av kjøleren. Forskjellen her er at de også vil få muligheten til å endre flere parametere.

Ferskvannsiden består av fire input-felt:

1. "**Effect**" representerer den varmevekslingen som skjer mellom det kalde vannet og det varme vannet i kjøleren. Dette er en viktig parameter fordi den bestemmer effektiviteten av kjøleren.
2. "**Flow**" representerer strømmen av ferskvann som går inn og ut av kjøleren. Ved å justere denne verdien, kan brukerne tilpasse kjølerens ytelse til deres spesifikke behov.
3. "**Temperature in**" er temperaturen på ferskvannet som går inn i kjøleren. Denne verdien er avgjørende for kjølerens evne til å opprettholde en optimal driftstemperatur.
4. "**Temperature out**" er temperaturen på ferskvannet som går ut av kjøleren. Dette gir brukerne en indikasjon på hvor effektiv kjøleren er til å senke temperaturen på vannet.

Deretter kommer vi til sjøvannsiden. Denne delen av konfiguratoren håndterer data knyttet til sjøvannet som brukes i kjøleren.

Sjøvannsiden består av fire inputfelt:

- "**Temperatur in**" er temperaturen på sjøvannet som går inn i kjøleren.
- "**Glykol**-**prosent**" er andelen glykol som blandes inn i ferskvannet. Glykol er en type kjølemiddel som forbedrer kjølerens evne til å senke temperaturen på vannet.
- "**Fouling-prosent**" er andelen fouling som blandes inn i sjøvannet. Fouling er et fenomen som oppstår når uønskede stoffer, som alger eller mineraler, akkumuleres på overflaten av kjøleren.
- "**Salinity percentage**" er saltinnholdet i sjøvannet. Saltinnholdet kan ha en betydelig innvirkning på kjølerens ytelse.

Den siste seksjonen, "Konstante verdier", inneholder tre inputfelt:

- "**Max pressure drop (freshwater)**" er det høyeste tillatte trykkfallet på ferskvannssiden.
- "**Max pressure drop (seawater)**" er det høyeste tillatte trykkfallet på sjøvannssiden.
- "**Max temperature out (seawater)**" er den maksimale utgående temperaturen på sjøvannssiden.

Denne seksjonen representerer grenseverdier som systemet ikke bør overstige for å sikre et optimalt resultat. Selv om disse verdiene generelt sett ikkje endres, har vi gitt brukere muligheten til å gjøre det om nødvendig.

HTML-koden for den avanserte konfiguratoren er strukturert slik at hver seksjon og hvert inputfelt er lett å identifisere. Hvert inputfelt er laget som et "form-field" element, som inneholder en "label" og en "input-container". Inne i "input-container" finner vi selve inputfeltet, som er av typen "number", og en enhetslabel, som viser hvilken enhet brukeren skal legge inn data i.

For eksempel, i inputfeltet for "Effect" i ferskvannsseksjonen, er selve inputfeltet laget som et "input"-element av typen "number". Det har en minimumsverdi på "1", en klasse som identifiserer den som et avansert inputfelt for ferskvann, og en "th:field"-attributt som binder den til den tilsvarende verdien i modell-klassen. Ved siden av inputfeltet finner vi en "unit-label" som viser at data skal legges inn i kilowatt (kW).

I sjøvannsseksjonen og seksjonen for konstante verdier følger vi samme struktur, men med forskjellige verdier og enheter. For eksempel, i inputfeltet for "Glykolprosent", er enhetslabelen endret til en prosentlabel for å indikere at data skal legges inn i prosent.

Til slutt, nederst i konfiguratoren, finner vi "calculate"-knappen som aktiverer beregningen av kjølerens ytelse basert på de innskrevne verdiene, en "cooling"-knapp for å navigere tilbake til kjølevisningen, og en "clear"-knapp for å tømme alle inputfeltene.

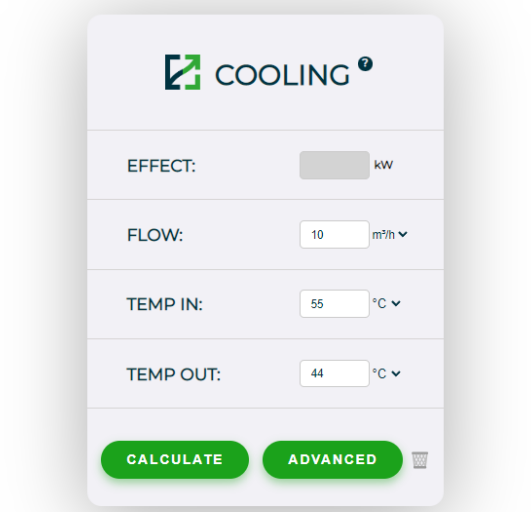
4.4.4 Produktresultat

Resultattabell

Etter at en bruker har trykt på kalkuler-knappen, vil nettsiden laste seg inn på nytt og vise en resultattabell (se figur 4.3). Denne tabellen har 5 kolonner:

- "**Model**": Denne kolonnen viser hvilken modell av Pleat-produktet som er representert i den aktuelle raden.
- "**Pass**": Denne kolonnen angir det totale antallet passasjer gjennom Pleat-produktet.
- "**Element**": Denne kolonnen viser antall elementer per passasje.
- "**Cooler Weight**": Denne kolonnen viser den totale vekten av produktet i kilogram.
- "**Area**": Denne kolonnen viser det totale arealet av kjøleren i kvadratmeter.

Når brukeren navigerer musepekeren over et element i tabellen, vil det elementet bli belyst en lys grønn farge, og musepekeren blir omgjort til en pekefinger. Dette er for å gi brukeren en visuell indikasjon på hvilket element som er i fokus, og for å vise at det er mulig å klikke på det aktuelle produktet for mer informasjon.



Result Table

MODEL	PASS	ELEMENTS	COOLER WEIGHT (KG)	AREA (M²)
DN65	1	1	175	1.8
DN65	2	2	208	3.6
DN65	3	3	241	5.4
DN65	4	4	274	7.2

Figur 4.7 Resultat

Tabellen er generert ved hjelp av Thymeleaf, en mal-motor fra Java. Det blir benyttet seg av Thymeleaf sine spesialattributter (de som starter med th:) for å dynamisk generere HTML basert på serverens modelldata.

For å generere tabellen, itereres det gjennom hver "Pleat" i listen result.pleats ved hjelp av th:each attributtet. For hver "Pleat", opprettes det en ny rad <tr> med fem celler <td> som inneholder informasjon om modellen.

Hvis det ikke finnes noen produkter som passer med brukerens parametere (det vil si, result.numberOfProducts er 0), vil en feilmelding vises i stedet for tabellen. Dette er håndtert av th:if attributtene i <h1 id="error-message"> og .

Resultatinformasjon

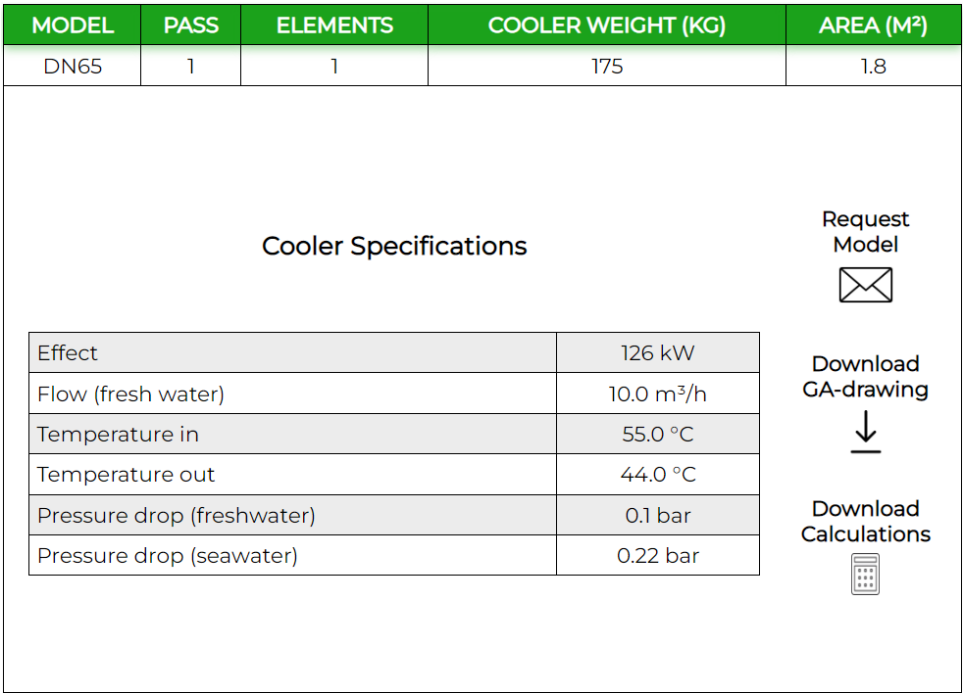
Dersom en bruker trykker på en rad i resultatlisten, vil elementet utvide seg og vise mer detaljer om det valgte produktet (se figur 4.4). Trykker brukeren på elementet igjen, så vil den lukke seg igjen å vise den normale resultat listen.

Dette oppnås gjennom funksjonen **toggleMoreDetails**(), som er knyttet til th:onclick attributtet i hver rad bestående av et produkt.

Det utvidede området av raden, inneholder mer spesifikk informasjon om det valgte produktet, som er organisert i en annen tabell. Her blir det gitt en mer detaljert oversikt over produktets tekniske spesifikasjoner, som inkluderer effekt, ferskvannsstrøm, inn- og utgangstemperatur for ferskvannet, og trykkfall for ferskvann og sjøvann. Hver av disse spesifikasjonene representeres av en rad i tabellen, og hver rad inneholder to celler: en for spesifikasjonens navn, og en for dens verdi. All info inkludert i resultattabellen var spesifikke ønsker fra bedrift. De er som følger:

- **Effect** – Dette er effekten som produktet bruker for varmeveksling når det er i bruk
- **Flow** – Dette er mengden ferskvann som går inn og ut av Pleat kjøleren
- **Temperatur in**– Dette er temperaturen på fersk vannet som går inn i kjøleren
- **Temperatur out** – Dette er temperaturen på fersk vannet som går ut av kjøleren
- **Pressure drop (freshwater)** – Dette er det høyeste tillatte trykkfallet på ferskvann siden (varm side).
- **Pressure drop (seawater)** – Dette er det høysete tillatte trykkfallet på sjøvann siden (kald side).

MODEL	PASS	ELEMENTS	COOLER WEIGHT (KG)	AREA (M²)
DN65	1	1	175	1.8



Figur 4.8 Kjølerspesifikasjoner

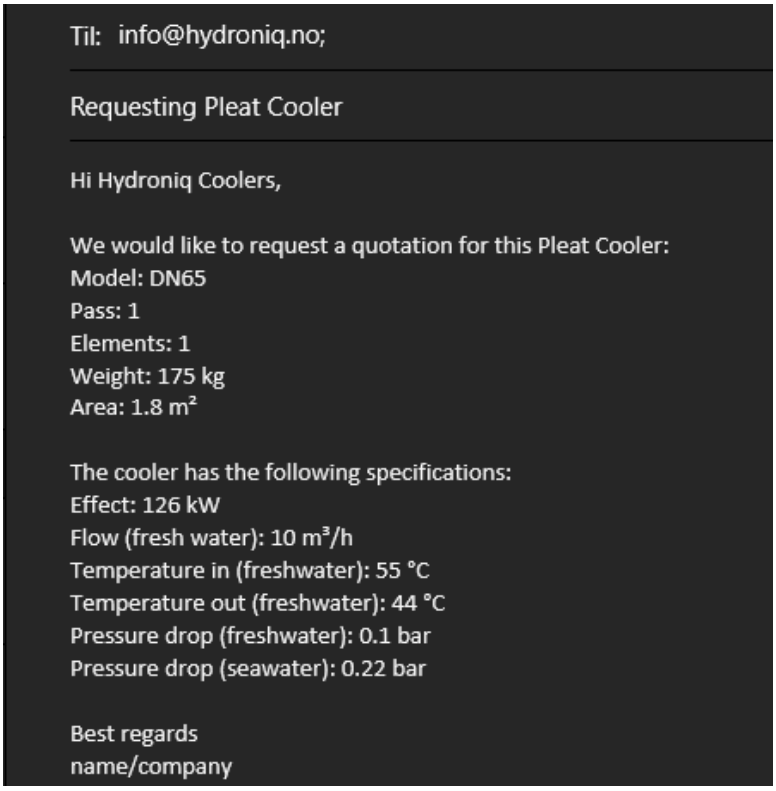
4.4.5 Produktfunksjoner

Disse produktfunksjonene vil være med på å hjelpe kunde å finne sitt produkt. Det er viktig å merke seg at alle disse funksjonene er initiert av en brukerhandling, det vil si et museklikk. handlingene er koblet til spesifikke funksjoner i koden gjennom th:onclick-attributtet (se 3.2.3).

Forespørsel av ønsket produkt

Dette er en funksjon som hjelper brukere med å sende en automatisk generert e-post til Hydroniq Coolers for å forespørre om et spesifikt produkt. Når brukeren klikker på "Etterspør modell"-ikonet, vil **sendEmail**()-funksjonen bli kalt. Denne funksjonen tar en rekke argumenter som representerer de forskjellige spesifikasjonene til Pleat-kjøleren.

Disse argumentene blir deretter brukt til å generere en URL-kodet streng som inneholder e-postens tittel og kropp. Denne strengen blir så brukt til å åpne brukerens standard e-postklient med en forhåndsutfylt e-post. E-posten inneholder en hilsen, en forespørsel om et tilbud på den spesifikke Pleat-kjøleren, og en liste over dens spesifikasjoner.



Figur 4.9 Resultat av forespørselsfunksjon

Nedlastning av GA-tegning

Funksjonen for å laste ned en generell arrangementstegning (GA-tegning) er utformet for å gi brukerne en enkel og direkte metode for å få tilgang til detaljerte visualiseringer av Pleat-kjøleren. Dette kan være spesielt nyttig for tekniske fagpersoner, ingeniører, eller andre som trenger en mer detaljert forståelse av kjølerens layout og konstruksjon.

Denne funksjonen gir brukerne muligheten til å laste ned en generell arrangementstegning (GA-tegning) av Pleat-kjøleren. Når brukeren klikker på "Last ned GA-Tegning"-ikonet, vil **openGADrawing**()-funksjonen bli kalt. Denne funksjonen tar to argumenter som representerer spesifikke spesifikasjoner for modellen.

Den bruker disse argumentene til å generere en streng som representerer navnet på den tilsvarende PDF-filen. Denne strengen blir så brukt til å omdirigere brukeren til en URL som starter en nedlastning av PDF-filen.

Årsaken til å bruke en streng til å omdirigere brukeren til korrekt PDF-URL, er fordi at den legger opp en direkte vei mellom brukers valg og den relevante tegningen. Brukervennligheten vil som et resultat av dette øke, ved at det blir minimert antall trinn som kreves for at bruker skal få tak i ønsket informasjon.

Nedlastning av kalkulasjoner

Denne funksjonen gir brukerne muligheten til å laste ned en PDF-fil som inneholder alle beregningene og detaljene for Pleat-kjøleren. Når brukeren klikker på "Last ned kalkulasjoner"-ikonet, vil **onCalculations**()-funksjonen bli kalt. Denne funksjonen tar Pleat-modellen som argument.

Den oppretter deretter et skjema-element og et skjult inputfelt i HTML-dokumentet. Inputfeltet er satt til å inneholde en JSON-streng som representerer Pleat-modellen. Skjemaet blir så automatisk sendt til serveren, som sannsynligvis genererer PDF-filen basert på dataene i Pleat-modellen. Se vedlegg **C** for et eksempel av PDF-en.

I dette tilfellet, ble det i starten påbegynt en generasjonsfunksjon i JavaScript, som ved hjelp av bibliotekene jspdf og html2canvas genererte en pdf basert på variablene og HTML-filen. Dette ble revurdert, og det ble derfor opprettet en kalkulasjonskontroller (se mer om denne prosessen i delkapittel 4.1.2) som tok seg av genereringen ville være mye mer optimalt. Dette vil være hensiktsmessig for å ta i bruk like fremgangsmåter gjennom hele prosjektet, slik at en unngår kaotisk og rotete "spaghetti"-kode. En annen grunn var at det ble enklere å kalle på Thymeleaf-attributtene, slik at en på enkelt vis gjorde det mulig å hente ut nødvendig data for bruker.

Denne fremgangsmåten muliggjorde også håndtering av unntak (exceptions). Her har dette blant annet blitt tatt i bruk ved å gi internalServerError dersom "renderer"-en ikke får til å konvertere fra HTML til PDF. Det blir også returnert en badRequest-feilmelding dersom metoden ikke får til å konvertere brukers krav til et objekt av Peat-klassen. Årsaker til dette kan være at forespørselen har blitt avbrutt midt i prosessen, eller på grunn av dårlig tilkobling.

4.5 Distribusjon

Distribusjon av en webapplikasjon refererer til prosessen med å gjøre applikasjonen tilgjengelig for endelige brukere. Dette kan bestå av å sette opp en produksjonsklar server, konfigurere nettverksinnstillinger, sikkerhetsinnstillinger og å overføre applikasjonen til serveren. Her inngår også distribusjonen av selve arbeidet, som i dette tilfellet består av versjonskontroll.

4.5.1 Server

Konfiguratoren er utviklet med tanke på at den skal bli brukt i produksjon på firmaets nettside etter ønsker fra oppdragsgiver. Derfor i utviklingsprosessen ble det brukt en Linux Ubuntu-server fra NTNU. Dette har hjulpet til testing av programmet og teste designet av web-siden på ulike enheter.

Det populære Java-applikasjonsutviklingsrammeverket Spring brukes til å skrive koden for å sette opp serveren. Som servletbeholder svarer Tomcat og håndterer forespørsler for oss. Programmering på serversiden krever tilstedeværelse av servlet-beholdere for å administrere all kommunikasjon mellom klient (f.eks. nettleser) og server som er vert for applikasjonen. (se 3.2.1)

Tomcat blir konfigurert i applikasjonen med å endre 'unloadDelay' til 8000 millisekunder. Dette betyr at serveren venter 8 sekunder før den avlaster en servlet. Dette kan være nyttig for å unngå problemer med applikasjoner som tar litt tid å lukke sine ressurser.

Applikasjonen krever at det blir programmert i Java versjon 17, og derfor stilles det samme kravet til serveren. Derfor for å distribuere applikasjonen på en Linux Ubuntu-server, er det nødvendig å sikre at Java 17 er tilgjengelig på serveren. Dette er et krav for at applikasjonen skal fungere.

I tillegg til Java 17 er det også implementert funksjonalitet for å integrere med OneDrive på Linux-serveren ved hjelp av "Abraunegg" sin OneDrive-pakke for Linux. [10] Dette gir muligheten til å bruke OneDrive som en del av applikasjonen. OneDrive er konfigurert til å kun laste ned endringer, aldri laste opp endringer og kun laste ned de filene som er relevante for applikasjonen. Det er verdt å merke seg at integrasjonen med OneDrive ble gjort etter ønske fra oppdragsgiveren, slik at oppdragsgiveren har muligheten til å legge til, endre eller oppdatere Excel filen og GA-tegninger (se eksempel i vedlegg **B**). Programmet er også lagt til rette for å ta i bruk disse nye endringene, selv om det kan være nødvendig med hjelp fra utvikler for å implementere all funksjonalitet.

4.5.3 Git Repository

I løpet av prosjektet har vi utnyttet et Git Repository for å strukturere, kontrollere og distribuere arbeidet vårt. Det har vært vår sentrale lagringsplass for hele kodebasen til prosjektet og har vært avgjørende for å koordinere arbeidet i teamet. (Se vedlegg **A)**

Vi har brukt GitHub, en skybasert hostingtjeneste for Git repositories, for å gjøre samarbeidet enklere. Med GitHub-prosjekter kunne vi organisere og prioritere arbeidet vårt på en visuell måte, noe som gjorde prosjektstyringen mer intuitiv og effektiv.

For hver sprint opprettet vi en serie med 'issues' som representerer de forskjellige arbeidsoppgavene som trengte å bli gjort. Disse oppgavene var tildelt til forskjellige teammedlemmer basert på deres evner og tilgjengelighet. Hver issue ble deretter fulgt opp med en egen gren for å skille arbeidet og unngå potensielle konflikter med resten av koden.

Vi har hovedsakelig jobbet på 'dev'-branchen, noe som tillot oss å jobbe og teste nye funksjoner uten å forstyrre den fungerende 'main'-branchen. Dette gjorde det også enklere å samarbeide og dele arbeidet, da 'dev'-branchen fungerte som en midlertidig 'master' der alle nye funksjoner ble integrert og testet sammen.

Når vi var fornøyde med endringene på 'dev'-branchen, og etter grundig testing og kvalitetssikring, ble endringene slått sammen med 'main'-branchen. Denne branchen ble brukt for 'releases' - offisielle versjoner av programvaren som er klare til distribusjon til brukerne.

Bruken av Git, sammen med GitHub, tillot oss å jobbe på en organisert og systematisk måte, og sikret jevn distribusjon av arbeid og kvalitetssikret kode gjennom hele prosjektet.

4.6 Testing

Testing er en grunnleggende del av programvareutviklingen. Det bidrar til å sikre at programvaren fungerer som forventet under forskjellige forhold og hjelper til med å identifisere eventuelle feil og avvik tidlig i utviklingsprosessen. Dette fremmer høy og forbedret kvalitet på det endelige produktet. Det er derfor vi har gjennomført omfattende tester som en del av utviklingsprosessen for konfiguratoren.

Valget av JUnit som enhetstestingsramme er basert på dets popularitet, brukervennlighet, og effektivitet innen Java-basert utvikling. JUnit lar oss utforme og utføre tester på prosjektklasser på en strukturert og systematisk måte. Vi har valgt å teste alle metoder og riktig oppretting av konstruktører for å verifisere at de fungerer som forventet under forskjellige forhold. Dette inkluderer både positive og negative tester, som sikrer at programvaren kan håndtere et bredt spekter av situasjoner. [23]

Kodeeksempel

```
@Test
void testCheckStringType() {
    assertEquals("Integer", ExcelServiceImplementation.checkStringType("40"));
    assertEquals("Double", ExcelServiceImplementation.checkStringType("3.14"));
    assertEquals("String", ExcelServiceImplementation.checkStringType("hello"));
}

@Test
void testGetResultList() {
    ExcelInput excelInput = new ExcelInput("", "80", "80", "50", units);
    List<Pleat> pleatList = excelServiceImplementation.getResultList(excelInput);

    assertNotNull(pleatList);
    assertFalse(pleatList.isEmpty());
}
```

Eksempler på tester inkluderer "**ExcelServiceImplementationTests**" og "**TomcatConfigurationTest**". I "ExcelServiceImplementationTests" sjekkes funksjonaliteten til metoder som **convertTemperatureBack**, **convertFlowBack**,

checkStringType og **getResultList**. Disse testene verifiserer at metodene returnerer forventede verdier når de blir gitt forskjellige inngangsparametere.

I tillegg har vi utført tester på "**ExcelInput**"-klassen for å sikre at dens metoder og konstruktører fungerer som forventet. Dette er viktig fordi denne klassen håndterer viktig datainngang og tolkning, og eventuelle feil her kan ha fatale konsekvenser for resten av programmet.

Kodeeksempel

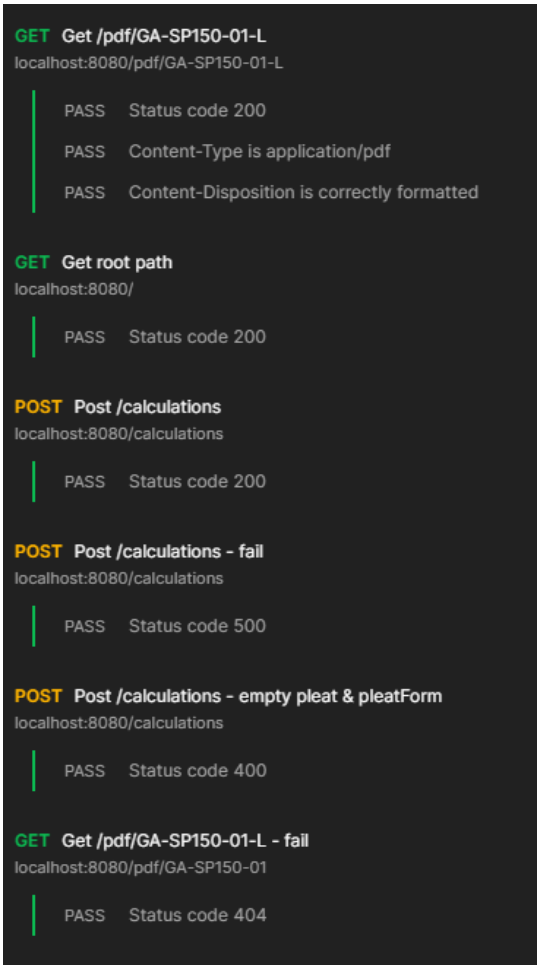
```
@Test
void testTomcatConfiguration() {
    // Checks if the TomcatFactory bean exists
    assertThat(tomcatFactory).isNotNull();

    TomcatWebServer webServer = (TomcatWebServer) tomcatFactory.getWebServer();

    StandardContext context = (StandardContext)
webServer.getTomcat().getHost().findChildren()[0];

    assertThat(context.getUnloadDelay()).isEqualTo(8000);
}
```

Vi har også inkludert tester for konfigurasjonen av vår webserver, som er en kritisk komponent i distribusjonen av vår applikasjon. For eksempel, i "**TomcatConfigurationTest**" klassen, sjekker vi at **TomcatServletWebServerFactory** bean eksisterer og at 'unloadDelay' er satt til den forventede verdien av 8000 millisekunder. Dette sikrer at vår serverkonfigurasjon er som forventet, og bidrar til påliteligheten og stabiliteten til vår applikasjon.

For å summere opp, gir disse grunnleggende testene oss tillit til programvarens pålitelighet ved å oppdage feil tidlig i utviklingsfasen. Det vil hjelpe med å skape en mer effektiv feilretting og med dette skaper også et mer robust og pålitelig sluttprodukt for kunden. Testingen er et kritisk område for komponentene våre i utviklingsprosessen, som sikrer at sluttproduktet til slutt møter krav og de forventede kvalitetsstandardene som er satt. Hensikten er ikke bare å finne feil i programmet, men også å validere og utforme systemet slik at det fungerer som forventet. Her er det prioritert hindring mot at senere endringer og oppdateringer skal kunne bryte eksisterende funksjonalitet.

Postman-tester

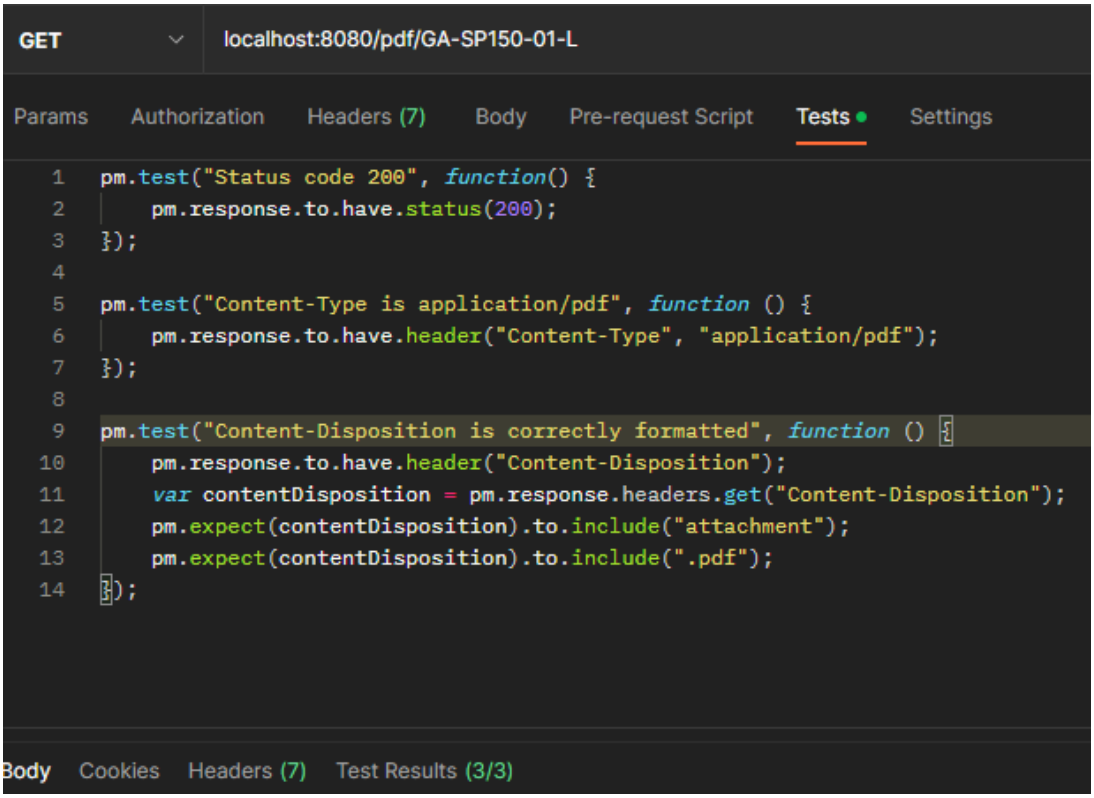
Postman er en plattform som brukes når en bygger og bruker API-er. Den hjelper til både under og etter utviklingsprosessen, og er i vårt tilfelle brukt for å teste API-funksjonalitet. [24]



Figur 4.12 Postman tester

Det har blitt laget et sett med Postman-tester som hjelper til å teste applikasjonens API. Bilde direkte ovenfor viser samlingen med Postman tester. Disse testene er blitt gjennomført med tanke på endepunkt funksjonalitet og for å kontrollere at riktig HTTP-statuskode kommer ved tilhørende forespørsel til serveren. Siden dette er en webapplikasjon med forskjellige endepunkter, må disse testene utføres for å teste HTTP-forespørslene.

Hovedsakelig har gruppen gjennomført tester for "POST og GET", dette kommer av at applikasjonen henter og sender data til et Excel-ark. Det forskjellige statuskodene applikasjonen er testet opp mot er "200, 500, 400 og 404". Med statuskode 200 menes tester som gir "OK" tilbake som svar, noe som sier at svaret er som forventet. Det andre status koden som er nevn er brukt for feilkoder i HTTP-forespørsler for indikasjoner om at testene feilet.



Figur 4.13 Postman-test eksempel

I eksempelet i fig. 4.13 tester vi først at statuskoden fra HTTP-forespørselen er 200 som sier at forespørselen er "Ok", dette indikerer at forespørselen var suksessfull. Den andre testen sjekker at "Content-Type"-overskriften i svaret er satt til "application/pdf", dette gjøres for å angi medietypen til ressursen. I dette tilfelle sjekker den om svaret er en PDF. I denne siste testen blir en resurs sjekket om den inneholder riktig format som vil være et format som inneholder både "attachment og .pdf", dette gjør at forespørselen valideres i forhold til formatet.

4.7 Design og arkitektur

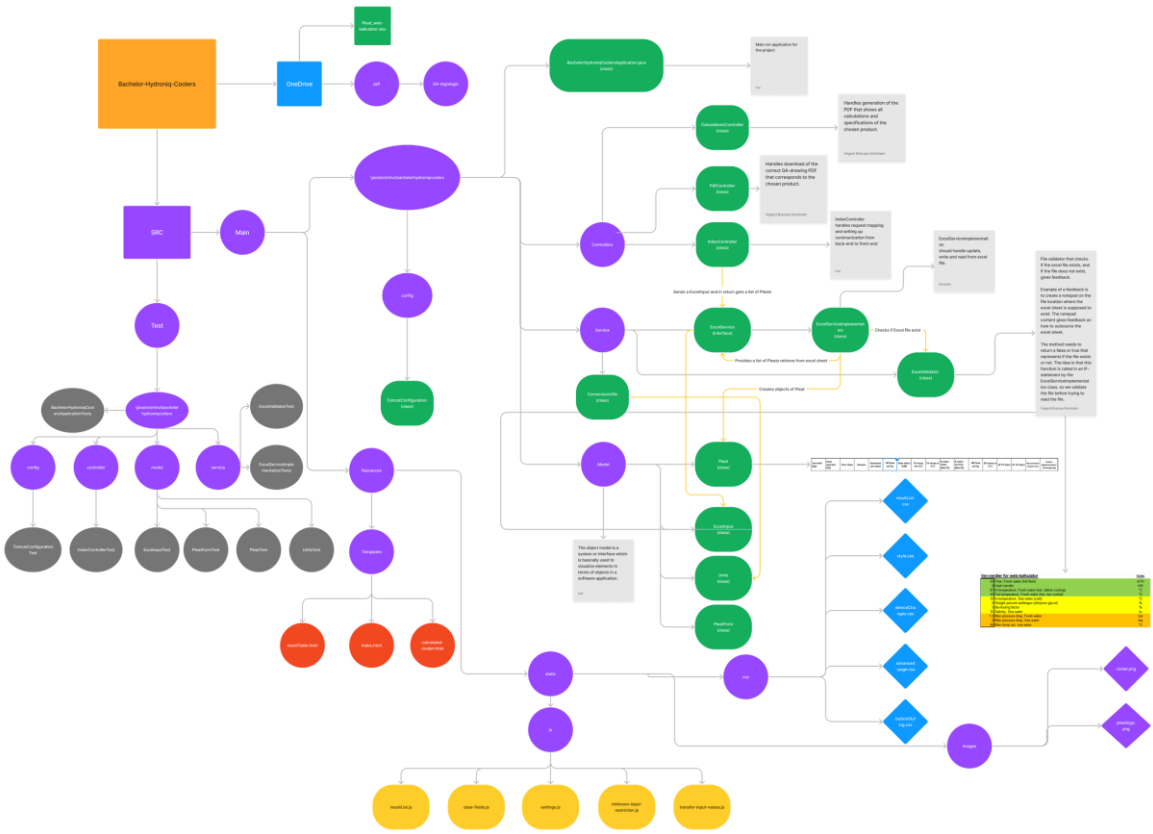
I arbeidet med design og arkitektur i prosjektet vårt, har vi satt et sterkt søkelys på å oppfylle prinsippene for lav kobling og høy samhørighet, to sentrale begreper som er forklart i detalj i kapittel 2. Teori.

La oss starte med **IndexController**, som er inngangspunktet til applikasjonen og håndterer de grunnleggende navigasjonsforespørslene. Denne klassen illustrerer både lav kobling og høy samhørighet. Den håndterer kun navigasjonsforespørsler og er dermed ikke direkte avhengig av de andre delene av systemet, noe som reflekterer lav kobling. Dens klart definerte rolle viser høy samhørighet, som gjør koden mer robust, forståelig og lett å vedlikeholde.

Et annet eksempel på implementeringen av lav kobling, er **CalculationsController**. Denne klassen håndterer HTTP-forespørsler relatert til kalkulasjoner og bruker TemplateEngine for å generere HTML-innhold. Ved å isolere denne klassen fra resten av systemet, reduserer vi avhengighetene mellom ulike deler av applikasjonen.

På den andre siden, illustrerer **ExcelServiceImplementation**-klassen prinsippet om høy samhørighet. Denne klassen har en bestemt rolle, som er å håndtere operasjoner relatert til Excel-data. Ved å fokusere på et spesifikt ansvarsområde, sikrer vi at klassen er robust og lett å vedlikeholde.

Programvaren vår følger en trelagsarkitektur, med klart definerte grenser mellom forskjellige ansvarsområder som kontrollerlaget, tjenestelaget og datalaget. Dette fremmer både lav kobling og høy samhørighet. Hvert lag har et spesifikt ansvar, og samhandler bare med de lagene som er direkte over eller under det. For eksempel, på kontrollerlaget håndterer IndexController navigasjonsforespørsler, mens CalculationsController håndterer forespørsler relatert til PDF-generering. Dette viser høy samhørighet innen laget, da hver kontroller har et klart definert ansvar. Samtidig viser det lav kobling mellom lagene, fordi endringer i kontrollerlaget ikke direkte vil påvirke andre lag.



Figur 4.14 Applikasjonens filstruktur

4.8 Sammenligning med tilsvarende systemer

I dette delkapittelet vil vi sammenligne vår løsning med eksisterende løsninger for å vurdere hvordan den står i forhold til konkurransen og hva som skiller den fra andre alternativer. Vi vil spesielt sette søkelys på SonFlow's SonCalc-kalkulator, som er en lignende applikasjon for beregning av varmevekslere, hvilket har inspirert både funksjonalitet og design av vår egen løsning. [25]

4.8.1 Sammenligning med SonFlows løsning

SonCalc-kalkulatoren er en etablert løsning for beregning av varmevekslere. Den tilbyr en funksjonell og teknisk tilnærming til kalkulasjonene, men mangler noen av de brukervennlige og estetiske aspektene som er sentrale i vår løsning. Vår løsning skiller seg fra SonCalc på en rekke områder:

Brukervennlighet

I vår løsning, har det største fokuset vært å utvikle et så brukervennlig grensesnitt som mulig, til fordel for både kunder og bedriftens salgsavdeling. Fra det lille vi fikk se av SonCalc-konfiguratoren, var inntrykket at den kan være ganske så innviklet og rotete. Det tar lang tid å gå gjennom alle parameterne, og det er mye informasjon som må prosesseres av bruker.

Her er vår løsning overlegen, da det eneste som møter kunde er fire inputfelt og en kalkuleringsknapp. Bruker vil også ha muligheten til å ta i bruk en avansert versjon, som inneholder flere valgmuligheter uten at det kompliserer konfigureringsprosessen noe særlig.

Design

SonCalc sitt design er simpelt, men kan fort se litt utdatert og gammeldags ut. Gruppemedlemmene ble fort samstemte om designet på konfiguratoren, og resultatet ble en modernisert, lysegråfarget løsning med lettleselig skrift, gode mellomrom og tilpasset responsivitet. Dette vil være finere og hvile øyet på, og dermed være med på å forbedre inntrykket til bruker.

Effektivitet

Firma har vært tydelig på at de vil ha en effektiv og rask løsning, og vårt inntrykk var at SonCalc ikke holdt mål på dette området. Vår løsning bruker mellom to og fem sekund på å kalkulere og visualisere resultatene, alt ettersom hvor store kalkulasjonene er og hvor mange produkter konfiguratoren kommer frem til. Dette vil øke produktiviteten, og være med på å redusere dødtid.

Kapittel 5: Drøfting

I dette kapittelet vil metode, resultater, og læringsutbytte bli drøftet.

5.1 Vurdering av metode og resultater

Det vil bli foretatt en vurdering av metodene som er blitt benyttet, samt resultatene gruppen har oppnådd i løpet av bacheloroppgaven. Her vil vi ta for oss kildekritikk, og hvordan kildene har hjulpet på veien. Det vil også bli diskutert begrensninger, endringer og avvik i løpet av prosessen, sammenlignet med hvordan planen og problemstillingen ble sett for seg i startsfasen.

5.1.1 Front-end og back-end

I begynnelsen av prosjektet antok vi at vi ikke hadde behov for en backend-løsning for applikasjonen. Det ble derfor i startsfasen jobbet primært på frontend, og det ble tenkt at alt av kalkulasjoner og data kunne konverteres fra Excel-fil til JavaScript-kode. Dette viste seg derimot å være en feilvurdering, ettersom at vi på et senere tidspunkt innså at firma spesifikt ville ta i bruk deres interne excel-fil i samhandling med vår applikasjon. Da ble det raskt konkludert at dette ikke ville fungere uten en back-end.

Det var hovedsakelig to årsaker til at det måtte utvikles en backend. Den første er at Excel-filen ikke kan være offentlig tilgjengelig. Uten en backend, kunne brukere potensielt fått hentet ut filen ved hjelp av nettleseren, noe som utgjør en massiv sikkerhetsrisiko for oppdragsgiver. Den andre årsaken var at vi hadde problemer med å kommunisere og skrive til Excel-fila uten bruk av backend-biblioteker, samt at konvertering av data og kalkulasjoner til JavaScript hadde blitt for avansert og tidkrevende.

Vi opprettet kommunikasjonen mellom frontend og backend ved hjelp av Spring Boot og Thymeleaf, slik at det kunne sendes data både til og fra. Her var mye av det som tidligere var blitt utviklet fortsatt brukbart, så en kunne bruke dette i videre implementering av en backend-siden.

5.1.2 Integrasjon av Excel-fil med Apache POI

En av de største utfordringene vi møtte på i løpet av prosjektet, var under integrasjonen av Excel-filen ved hjelp av Apache POI-biblioteket. En av de langvarige prosessene vi måtte gjennomgå var å håndtere flere feilmeldinger og feilresultat relatert til integrasjonen, da det i startfasen var kollisjoner mellom applikasjon og excel-fil.

For å finne feilen, måtte det dedikeres flere timer hver dag til feilsøking. Etter intense dager med konstant feilsøking, ble feilen eventuelt luket frem. En SORTER-funksjon i kalkulasjonsformlene var ansvarlig for dette, da det viste seg at Apache POI ikke støtter Excel-funksjonen SORTER. Dette førte til komplikasjoner i prosjektet, som resulterte i at Excel-filen ikke ville utføre kalkulasjoner selv om det ble overført data fra applikasjon.

Det ble startet vurderinger av alternative metoder for å håndtere integrasjon av Excel-filen for å unngå dette problemet. Det måtte vurderes om det var nødvendig å endre løsningen for å unngå bruken av denne funksjonen, eller om det var mulig å finne et annet bibliotek som støttet denne funksjonen.

Løsningen ble å modifisere regnearket til å fjerne samtlige tilfelle av SORTER-funksjonen, og en kom også frem til at dette var mulig å sorteres internt i Java-applikasjonen. Heretter gikk prosessen med Apache POI feilfritt, og det arbeidet med datahåndtering mellom front-end og back-end kunne fortsettes.

5.2 Betydning av resultatene

Praktisk betydning

Det eksisterende systemet som denne webapplikasjonen skal erstatte har kun tidligere gitt kunder muligheten til å ta kontakt og diskutere direkte med salgsteamet for å få et produkt. Med webløsningen skal kunder bli mer inkludert ved at de får muligheten til å konfigurere en Pleat gjennom en webbasert løsning. Dette gir bedriften et ekstra alternativ når det gjelder salg av varmevekslere, og det har mulighet til å ende opp som en kommersiell suksess i framtiden.

Det å få konfigurert ut en varmeveksler kan ikke sammenlignes med noen som skal på nettet for å handle klær, da det er nødvendig å ha litt tekniske kompetanse for å kunne håndtere denne konfiguratoren. Det bedriften har sett for seg er at denne løsningen kommer til å bli brukt av kunder som holder på med kjøp, design og bygging av virksomheter innen den maritime industrien. Typiske brukere kan være personer innen skipsbyggingsindustrien. Konfiguratoren krever spesifikk informasjon om effekt, strømming, temperatur inn og temperatur ut for å få ut en løsning, noe som gjør at brukere må ha en del kunnskap om sitt ønske på forhånd før bestilling.

Konfiguratoren leverer en del ekstra nyttige funksjoner for brukere. Dette består av nedlastingning av GA-tegninger, PDF-generering av kalkulasjoner, og tilbudsfunksjonen. Disse er meget vesentlige, da de skal fungere som pekepinner og hjelp for kunder. GA-tegningen vil gi brukeren mulighet til å få litt ekstra informasjon om produktet, som f.eks. høyde, lengde, bredde og plassering av uttak av rørkoblinger/flens. Samtidig vil brukeren få et grafisk bilde av produktet fra flere vinkler. Kalkulasjonens PDF skal gi en fullverdig liste av alle faktorer som har blitt beregnet for å komme fram til resultatet, samt alle av det valgte produktets spesifikasjoner. Tilbudsfunksonen tar seg av bestillingsprosessen med bedriften, hvor den vil videresende bruker videre inn til sin valgte mailtjener. Her vil bruker automatisk få inn all informasjon (av det valgte produktet) som trengs i en e-post for at bedrift skal kunne gi et tilbud til kunde.

Teknisk betydning

Siden det ble valgt å bruke Apache POI (2.2.2), har implementeringen gitt en god måte for å integrere Excel-data i konfiguratoren.

Sammenlignet med en tradisjonell database, gir bruken av Apache POI og Excel også både fordeler og ulemper. En av fordelene er fleksibiliteten. Excel er et veldig kjent verktøy som er tilgjengelig til det fleste og mange har kunnskaper om det, noe som gjør databehandling mer tilegnet brukere som kanskje ikke har avansert teknisk kunnskap.

Ved å bruke Apache POI, kan Excel-filer behandles på samme måte som data ville blitt behandlet i en database. Det vil derimot ha sine begrensninger.

Databaser er bedre designet for å håndtere store mengder data og har bedre skalerbare evner enn Excel, som fort kan gå utover ytelsen når for store mengder data blir håndtert. Videre tilbyr databaser sterkere funksjoner som passer på dataintegritet og sikkerheten rundt dataene, hvilket er spesielt viktig i applikasjoner som håndterer sensitiv data.

Til tross for disse begrensningene, har bruken av Apache POI for Excel-integrasjon gitt gruppen og klient en unik tilnærming til datahåndtering, og oppdragsgiver har fått det de ønsker ved at de kan gjøre videre utvikling gjennom Excel uten kompetanse i applikasjonsutvikling.

Sammenlignet med andre løsninger, har kombinasjonen som ble brukt (Spring Boot, Thymeleaf og Apache POI) for Excel-integrasjon resultert i en robust, skalerbar og brukervennlig løsning. Ved å kombinere alle disse teknologiene tror vi at resultatet kan gi oppdragsgiver en funksjonell og brukervennlig løsning i tiden som kommer (vedlikehold og support tatt i betraktning).

Resultat av testing

Applikasjonen har blitt grundig testet på disse tre områdene:

- JUnit-testing,
- Postman-testing av webapplikasjonen
- Brukertester

Valg av tester avhenger av faktorer som applikasjonstype, brukernes forventinger, og hvilke risikoer som er assosiert med feil i applikasjonen. Disse testene føler gruppen er et minstekrav for oppgaven.

Testing avdekket også flere feil i Excel-arket og ga innsikt i funksjoner som måtte opprettes for å behandle feilmeldinger i Excel. Excel har forskjellige feilmeldinger som for eksempel "#NUM, #DIV/0!, #VALUE!, #NAME?" også videre. For å håndtere disse feilene implementerte vi en metode som sjekket om "hashtag" (#)-symbolet befant seg i cellene. Hvis symbolet ble funnet så ble hele raden ignorert.

I JUnit-testene utviklet vi sjekkmetoder for å sikre at verdiene er innenfor bestemte kriterier, noe som hjalp med å redusere unødvendig koderepetisjoner. Det ble opprettet sjekkmetoder som sjekket at en "String" ikke kan være null eller tom, og sjekkmetoder for "Integer" og "Double" som passet på at et nummer ikke kan være negativt eller null. Disse sjekkene ble brukt i Pleat-klassen for å sikre at verdiene er gyldige.

JUnit-testene gått gjennom flere iterasjoner etter at det ble bestemt at «Builder»-mønsteret (2.3.2) skulle implementeres. Dette resulterte i at de tidligere testene måtte tilpasse seg det nye designet.

Tester vil alltid være en viktig del av utviklingsprosessen. De har bidratt til å rette opp og avdekke feil i arbeidsprosessen, og sikrer at oppdragsgiver krav og forventninger er oppfylt. Det grundige arbeidet med tester har vært viktig for å sikre at robusthet og påliteligheten til den ferdige konfiguratoren er på plass.

5.3 Hva har vi lært

Hele prosjektet har vært en stor utfordring, og det har blitt lagt inn store mengder arbeid for å komme i mål. Dette ved hjelp fra medlemmene i gruppe 9, men også fra Hydroniq, Otto Godeset og Havnevik AS. Alle og enhver har fått erfart opplevelsen av å utvikle en applikasjon fra idé til ferdig produkt. Prosessen har vært utfordrende, men også ekstremt lærerik.

I startsfasen hadde vi sett for oss å bli testet i programmerings- og-dokumenteringskunnskap, men vi fikk raskt innse at det var mye annet som måtte settes inn i. Vi måtte bryne oss på å jobbe kontinuerlig i et agilt sprint-miljø, levere nye iterasjoner av applikasjonen annenhver uke, samt presentere ideer og utført arbeid.

Prosjektet har lært oss å tenke kritisk og drøfte over ulike problemstillinger, samt gitt oss muligheter til å løse disse problemene. Gruppen har også fått innsikt over hva som skal til for å realisere et produkt sammen med en bedrift. Denne prosessen har tvunget gruppen til å måtte endre på hvordan de tenker, hvor en har måtte satt seg inn i tankesettet til en gjennomsnittlig kunde. En har måttet utarbeide en løsning etter den tanken at det skal være enkelt for flest mulig å kunne regne frem til korrekt produkt.

En annen utfordring var at gruppen jobbet innenfor et felt en hadde lite til null erfaring med fra før, nemlig varmevekslere. Dette området er ukjent for de fleste dataingeniører. Det har vært nødvendig å lære seg hva ulike varmevekslere er, hvilken oppgave det gjør, og hvor deres bruksområder ligger for å få mer innsikt til å kunne utvikle konfiguratoren. Selv om dette er hovedsakelig utenfor våre kompetanseområder, har det gitt oss som utviklere en verdifull læringsopplevelse siden brukere kan komme fra alle mulige arbeidsfelt. Det trengs ofte tverrfaglig kunnskap for utviklere, og evnen til å tilegne seg ny informasjon raskt er essensielt innenfor teknologi- og ingeniørbransjen. Denne nye erfaringen har forbedret vårt potensiale til å tenke løsningsorientert og kritisk, hvilket er ferdigheter som vil være uvurderlige i en fremtidig karriere.

Kapittel 6: Konklusjon og videre arbeid

I dette kapittelet skal det gås gjennom en konklusjon på arbeidet som er blitt gjort, samt presenteres forslag til løsninger gruppen ikke har fått tid til å implementere. Dette kan hjelpe firmaet å ta i bruk, videreutvikle og vedlikeholde applikasjonen i fremtiden.

6.1 Konklusjon

Som en del av siste semester har vi, en gruppe dataingeniørstudenter, utviklet en webbasert produktkonfigurator som en del av vår bacheloroppgave. Etter måneder med intenst og kontinuerlig arbeid, har gruppen utviklet et sluttprodukt som både gruppen og oppdragsgiver er stolte av.

Det har blitt brukt en fremgangsmåte strukturert rundt agile utviklingsprinsipper, med fokus på iterativ utvikling gjennom bruk av sprinter. Ved starten av hver sprint har det blitt tatt en grundig vurdering av tilbakemeldinger fra sluttmøter i foregående sprint. Dette har gitt informasjon som forenkler planlegg og tillater oss å sette klare, realistiske mål for arbeid i nye sprinter.

Testing har vært en stor del av de fleste sprinter, noe som har lagt til grunn for høy kvalitet og funksjonalitet gjennom mesteparten av utviklingen. Det har blitt produsert nye utgivelser av applikasjonen kontinuerlig. Dette er noe som har gitt gode muligheter for brukertester, som igjen har resultert i et godt grunnlag for feilretting og utbedringer. Kontinuerlige utgivelser har skaffet prosjektet flere stabile versjoner å gå tilbake til, samtidig som det har markert vår fremgang i løpet av prosessen.

Kjernen i vår utvikling av applikasjonen har bestått av implementering av ulike teknologier som skal samhandles. Dette har vært kritisk for å kunne tilrettelegge for en brukervennlig webapplikasjon, som har måttet håndtere data og kalkulasjoner uten konflikter. Java og Spring Boot har stått som skjelettet for backend-koding, og Thymeleaf har forenklet den dynamiske utviklingen av websiden. Apache POI-biblioteket har hatt hovedansvaret for Excel-integrasjon. Det har gitt muligheten for maksimal utnyttelse av Excel-filen som datahåndtering.

Ved å kombinere alle disse forskjellige teknologiene, har vi klart å utvikle og levere et produkt som møter krav og forventninger som ble satt til oppgaven. Resultatet er en brukervennlig og robust varmeveksler-konfigurator som får jobben gjort for både kunde og bedrift.

6.2 Videre arbeid

Gruppen er fornøyde med sluttproduktet, men det er alltids rom for forbedring og videreutvikling. Det vil foreslås følgende tilleggsfunksjoner dersom det skulle bli relevant i fremtiden:

- Oppsett av applikasjon på en server må ventes med til etter bacheloroppgaven,
- Vurdere muligheten for å konvertere fra Excel til en database for databehandling, eksempelvis SQLite, MySQL PostgreSQL også videre. Dette bør gjøres av noen som har kompetansen i programvareutvikling, samt forståelse for kuldetekniske utregninger.
    - Dette vil forbedre skalerbarheten som vil øke svartiden og effektiviteten betraktelig.
    - Videre vil dette forbedre dataintegriteten, og vil minske sjansen for korrupt data og sørge for datakvalitet
    - Vil også sørge for at samtidig tilgang til data ikke vil være et problem
    - Bedre sikkerhet
    - Bedre mulighet for sikkerhetskopiering
    - Bedre ytelse
- Utvikle brukernivåer gjennom autorisering og autentisering, slik at administratorer får ekstra tilgang sammenlignet med vanlige brukere
- Med en konklusjon om at en database vil være en naturlig vei å gå, så gir dette også bedre muligheter for å legge til nye produkter, GA-tegninger og kalkulasjoner for applikasjonen. Her må det implementeres autentisering og kanskje noen former for autorisasjon. Mulig autorisasjon som kunne blitt lagt til er rollebasert tilgang, slik at det bare er brukere med administrator-tilgang som får muligheten til å legge til eller endre produkter i databasen.
- Utvikle en mobil-app versjon av web applikasjonen for å øke markedspotensialet kan være en god innovasjon.
- Feilsøking og reparere feil i eksisterende løsning for å fjerne svakheter som ikke har blitt oppdaget av utviklerne.
- Implementere nye løsninger som viser produktene som 3D-modeller, her er det mange forskjellige teknologier som kan tas i bruk. For eksempel kan en WebGL-basert 3D-motor brukes til å gjøre dette på en nettleser. Det finnes populære biblioteker som Three.js til dette.

Alt i alt, er det mye som kan implementeres og videreutvikles, men det finnes alltid en grense på mengden arbeid en kan få til på et semester. Gruppen er stolt av innsatsen, og det er oppdragsgiver også i stor grad. Gruppen ønsker Hydroniq Coolers ASlykke til med videre bruk av applikasjonen, og vi håper de vil få godt nytte av den!

Samfunnspåvirkning

Gruppens utviklingsprosess inkluderer ulike elementer som kan ha innvirkning på miljømessige og sosiale forhold. Dette spesifikke prosjektet har gitt gruppen muligheten til å oppnå digitalisering og automatisering av bedriftsprosesser som mål. Muligheten for å endre krav innenfor arbeidsmarkedet kan potensielt være en effekt av dette, som vil kunne bidra til å bedre arbeidsforholdene gjennom mindre repeterende arbeid.

Bærekraftig forretningspraksis kan oppnås ved å gå digitalt med produksjonsprosesser og se på det fra et miljøperspektiv, og optimalisering av produksjonsprosedyrene våre sammen med en reduksjon i feil vil resultere i mindre svinn samt redusert bruk av ressurser. Flere FNs bærekraftsmål blir nådd av dette initiativet, inkludert mål nummer 9: Industri, innovasjon og infrastruktur, og mål nummer 12: Ansvarlig forbruk og produksjon.

Det er essensielt at etisk praksis ved håndtering av brukerdata blir en viktig prioritering. Gruppen mener at det har blitt utført nøye håndtering av konfidensiell informasjon i konfiguratoren, som er et resultat av gruppens prioritering av både dataintegritet og sikkerhet.
Når det gjelder etiske hensyn rundt gjennomføringen av oppgaven, la gruppen betydelig vekt på å opprettholde åpen kommunikasjon og et kontinuerlig samarbeid med firma. Som et resultat av dette, ble vi fast bestemt på å utvikle et produkt som tilfredsstiller deres forventninger til både funksjon og kvalitet.

Gruppen erkjenner både styrker og begrensninger ved prosessen, men er innforstått med at et såpass omfattende prosjekt alltid vil være en lærerik opplevelse. Det å ha måttet koordinert seg og jobbet sammen som en del av et team, hvor hver og en har måttet håndtere de teknologiske utfordringer knyttet til prosjektet samtidig, har vært et bevis på oppgavens omfang. En positiv konsekvens er at vi på grunn av prosjektet har fått en dyp forståelse for å implementere teorier i praksis, samt fått mulighet til å tilpasse oss etter uventede vanskeligheter under prosessen.

Bærekraftsvurderinger i forbindelse med prosjektet, består av at gruppen ser at prosjektet kan være med på fremme ansvarlig forbruk sammen med å drive økonomisk vekst gjennom jobbskaping.
En fremtidsvisjon kan være å utvide konfiguratorens muligheter til å inkludere en grundig livssyklusvurdering av alle produktene. For eksempel, i stedet for bare å vurdere ressursbruken i produksjonsfasen, kan en også ta hensyn til faktorer som energiforbruket i bruksfasen eller mulighetene for resirkulering ved produktets levetids slutt.

Dessuten vil det å inkludere spesifisert informasjon i konfiguratoren (temperatur, effekt, mengde, størrelse osv.) la forbrukere ta bedre beslutninger ved å vurdere det generelle miljøfotavtrykket i stedet for bare prislapper for et produkt når de kjøper det. Dette kan forhåpentligvis være med på å fremme bærekraftig forbruk.

Etter å ha tatt alt i betraktning, føler vi oss sikre på at prosjektet vårt kan ha mulighet til å gjøre en merkbar forskjell både sosialt og miljømessig. For å oppnå dette, kan konfiguratoren trenge fremtidige forbedringer. Vi ser for oss at Hydroniq er dedikerte til å utforske disse mulighetene, og til syvende og sist ha en innvirkning på å skape en bærekraftig fremtid.

Referanser

[1] Pedersen, B. (2022, 25. november). *Varmeveksler*. Store Norske Leksikon. https://snl.no/varmeveksler (Hentet: 10. april 2023)

[2] Fleitas, A.G. (2022, 12. oktober). *Spreadsheets vs. Databases, Everything You Need to Know.* HubSpot. https://blog.hubspot.com/sales/database-vs-spreadsheet (Hentet: 10. april)

[3] Clarity Hub. (2018, 17. september). *Low Coupling, High Cohesion.* Medium. https://medium.com/clarityhub/low-coupling-high-cohesion-3610e35ac4a6 (Hentet: 11. april)

[4] Pankaj. (2022,3. august) *Builder Design Pattern in java. DigitalOcean.* https://www.digitalocean.com/community/tutorials/builder-design-pattern-in-java#builder-design-pattern (Hentet: 11. april)

[5] Belay, H. (2023, 26. mars). *From Problems to Solutions: Understanding Design Patterns.* Dev. https://dev.to/documatic/from-problems-to-solutions-understanding-design-patterns-3b7i (Hentet: 11. april)

[6] IBM. (2023). *What is a REST API*. IBM. https://www.ibm.com/topics/rest-apis (Hentet: 16. april 2023).

[7] Mozilla. (2023). *MVC*. Mozilla Developer Network. https://developer.mozilla.org/en-US/docs/Glossary/MVC (Hentet: 16. april 2023).

[8] Gillis, A.S. (2023, mars). *Dependency injection*. TechTarget. https://www.techtarget.com/searchapparchitecture/definition/dependency-injection (Hentet: 10. april 2023).

[9] Hydroniq Coolers. (2023). *Pleat Cooler*. Hydroniq Coolers. https://www.hydroniq.no/pleat-cooler (Hentet: 10. april 2023)

[10] Abraunegg. "*OneDrive.*" GitHub. https://github.com/abraunegg/onedrive (Hentet: 22. april 2023)

[11] "*Java Spring Boot.*" IBM. https://www.ibm.com/topics/java-spring-boot (Hentet: 22. april 2023)

[12] "*What is JavaScript?*" (2023, 19. april) Mozilla Developer Network. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (Hentet: 22. april 2023)

[13] "HTML." (2023, 19. april) GeeksforGeeks. https://www.geeksforgeeks.org/html/ (Hentet: 22. april 2023)

[14] "What is Object-Oriented Programming?" (2023, 15. februar) Emeritus. https://emeritus.org/blog/coding-what-is-object-oriented-programming/ (Hentet: 25. april 2023)

[15] "*Thymeleaf.*" Thymeleaf. https://www.thymeleaf.org/ (Hentet: 25. april 2023)

[16] "*What is Scrum?*" Scrum.org. https://www.scrum.org/resources/what-scrum-module (Hentet: 25. april 2023)

[17] Brush, K. Silverthorne, V. (2022, november) "*Agile Software Development.*" TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development (Hentet: 25. april 2023)

[18] "What is Git?" Git. https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F (Hentet: 28. april 2023)

[19] "*What is CSS?*" TutorialsPoint. https://www.tutorialspoint.com/css/what_is_css.htm (Hentet: 28. april 2023)

[20] "Figma." Figma. https://www.figma.com/ (Hentet: 2. mai 2023)

[21] "*What is Tomcat?*" JavaTpoint. https://www.javatpoint.com/what-is-tomcat (Hentet: 2. mai 2023)

[22] Gaba, I. (2023, 14. februar) "*What is Java*?" Java. https://www.java.com/en/download/help/whatis_java.html (Hentet: 6. mai 2023)

[23] Gaba, I. (2023, 14. Februar) "*What is Junit?*" Simplilearn. https://www.simplilearn.com/tutorials/java-tutorial/what-is-junit (Hentet: 6. mai 2023)

[24] "*What is Postman?*" Postman. https://www.postman.com/product/what-is-postman/ (Hentet: 6. mai 2023)

[25] "*SonFlow*." SonFlow. https://sonflow.eu/ (Hentet: 6. mai 2023)

[26] "*Apache POI*." Apache. https://poi.apache.org/ (Hentet: 12. mai 2023)

[27] Rani, B. "*Introduction of Programming Paradigms.*" GeeksforGeeks. https://www.geeksforgeeks.org/introduction-of-programming-paradigms/ (Hentet: 12. mai 2023)

Vedlegg

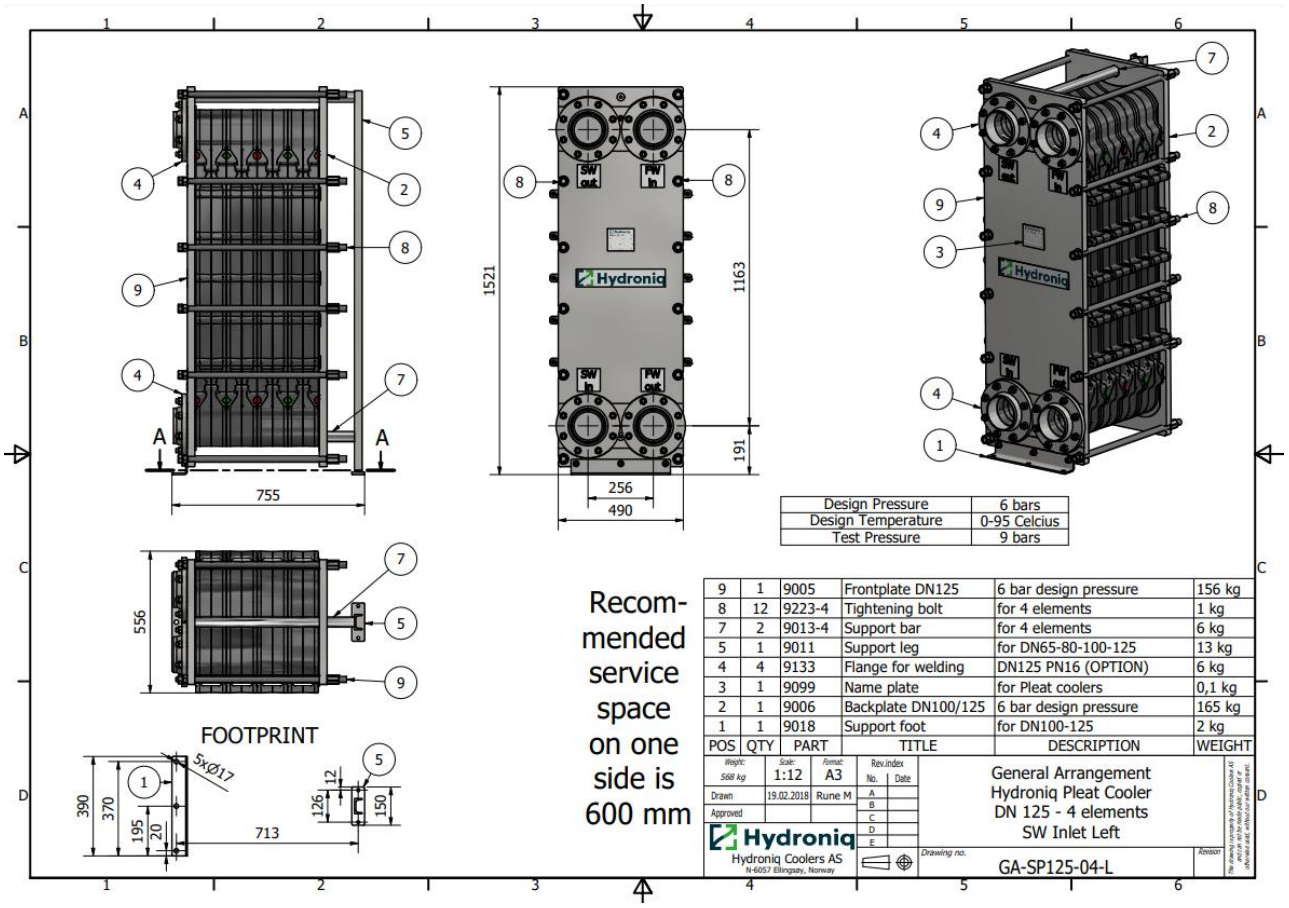
A.

Repo:

https://github.com/iHateThisName/Bachelor-Hydroniq-Coolers

Server:
http://129.241.152.39:8080/

B.


1521
1163
191
256
490
755
556
FOOTPRINT
5xØ17
390
370
195
20
713
126
12
150
Recom-
mended
service
space
on one
side is
600 mm
Design Pressure 6 bars
Design Temperature 0-95 Celcius
Test Pressure 9 bars
9 1 9005 Frontplate DN125 6 bar design pressure 156 kg
8 12 9223-4 Tightening bolt for 4 elements 1 kg
7 2 9013-4 Support bar for 4 elements 6 kg
5 1 9011 Support leg for DN65-80-100-125 13 kg
4 4 9133 Flange for welding DN125 PN16 (OPTION) 6 kg
3 1 9099 Name plate for Pleat coolers 0,1 kg
2 1 9006 Backplate DN100/125 6 bar design pressure 165 kg
1 1 9018 Support foot for DN100-125 2 kg
POS QTY PART TITLE DESCRIPTION WEIGHT
Scale: 1:12
Format: A3
Drawn 19.02.2018 Rune M
Approved
Hydroniq Coolers AS
N-6057 Ellingsøy, Norway
General Arrangement
Hydroniq Pleat Cooler
DN 125 - 4 elements
SW Inlet Left
GA-SP125-04-L

C

19. mai 2023

Hydroniq
PLEAT CONFIGURATOR

Pleat Cooler Calculations

Pleat Model	
Recommended port size	DN65
Passes	2
Elements	2
Weight (Dry)	188 Kg
Weight (Wet)	208 Kg
Area	3.6 M²

User Inputs	
Effect	115 kW
Flow	10 m³/h
Temperature in (FW)	50 °C
Temperature out (FW)	40 °C
Temperature in (SW)	32 °C
Glycol percentage	10.0
Fouling percentage	10.0
Salinity percentage	3.5
Max pressure drop (FW)	1.5
Max pressure drop (SW)	1.5
Max temperature (SW)	60 °C

Specifications		
	Cold Side	Hot Side
Medium	Seawater	Freshwater
Flow	10 m³/h	10 m³/h
Temperature in	32 °C	50 °C
Temperature out	42,1 °C	40 °C
Pressure drop	0.25	0.21
K-Value (Clean)	4443 W/m²K	-
K-Value (Service)	3999 W/m²K	-

D.

ATTEST i.f.m. Bacheloroppgave

PROSJEKTDELTAKERE	
Navn:	Ivar Bredeli (f. 07.03.1997) Steffen Watt Gravdehaug (f. 19.12.1997) Vegard Øveraas Karlstrøm (f. 23.01.1997) Kenneth Misund (f. 04.09.1996)
Start- / Sluttdato:	10.01.2023 – 22.05.2023
Prosjektbeskrivelse:	Web-løsning med produktkonfigurator for Pleat varmevekslere

Det bekreftes herved at Ivar Bredeli, Steffen Watt Gravdehaug, Vegard Øveraas Karlstrøm og Kenneth Misund, (alle Bachelor-studenter ved NTNU Ålesund) i perioden 10. januar til 22. mai har jobbet med en prosjekt- (Bachelor-) oppgave for Hydroniq Coolers AS.

Oppgavens omfang var utvikling av en web-basert produktkonfigurator for Pleat varmevekslere. I den forbindelse måtte studentene sette seg inn produktet Pleat og i beregningsmodellen vi benytter, og ta med seg denne kunnskapen inn i utviklingen av et web-grensesnitt som gjennom våre hjemmesider vil være tilgjengelig for egenberegning av Pleat-baserte kjøleløsninger for potensielle kunder.

Vi ser på en slik produktkonfigurator som et viktig verktøy i vårt salgsarbeid med Pleat produkter, og har i den forbindelse også nedlagt vesentlige egne ressurser i å støtte prosjektet underveis. Når de nå har kommet frem til en løsning som vi er tilfredse med for kommersiell bruk, er dette noe som bevitner den kompetanse og kvalitet som studentene har utvist i sitt arbeide, som har vært utført på en meget tilfredsstillende måte.

Ellingsøy, 19.05.2023
Sted/dato

Hydroniq COOLERS AS
6057 ELLINGSØY, NORWAY

Endre Kristiansen
CFO / CHRO

Hydroniq Coolers AS
Ellingsøyvegen 740
6057 Ellingsøy
Norway

Tel: + 47 70 10 42 00
info@hydroniq.no
www.hydroniq.no
NO 825 935 592 VAT

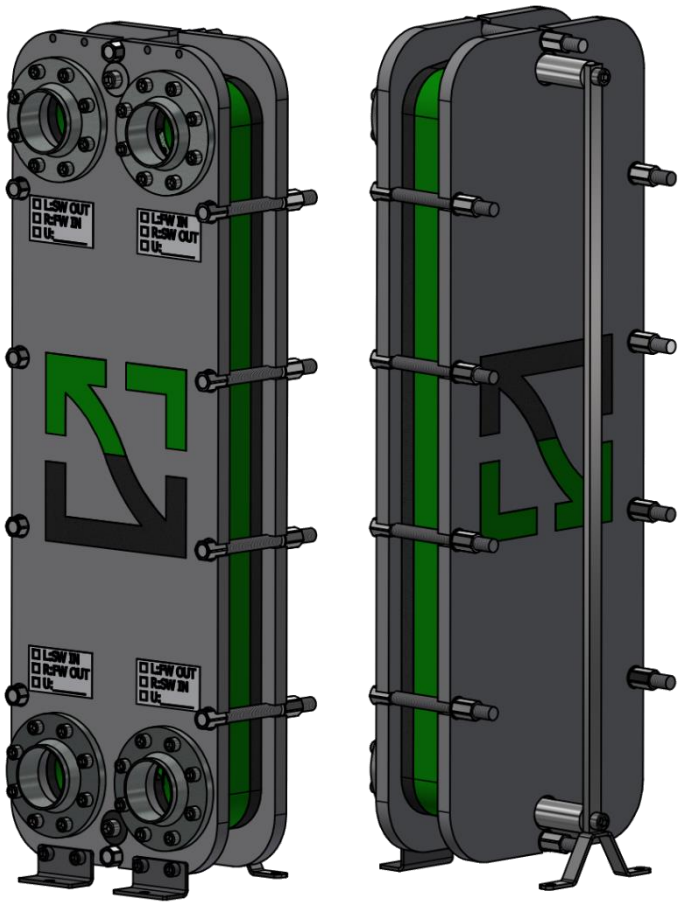
Den Norske Bank
Swift: DNBANOKK
Bank account: 5353.05.17531
IBAN: NO 32 5353 05 17531

E.

Produktkonfigurator for Hydroniq Coolers forprosjektplan

Ivar Bredeli, Steffen Gravdehaug, Vegard Karlstrøm, Kenneth Misund

Hydroniq
PLEAT CONFIGURATOR



Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
13.01.2023	1.0	Oppstart	Steffen, Vegard, Ivar, Kenneth
25.01.2023	2.0	Fylle ut de fleste punkt og finpussing.	Vegard, Ivar, Kenneth, Steffen

1. Mål og rammer

1.1 Orientering

Hvorfor denne oppgaven. Hvordan fikk dere tak i den.
Vi har valgt å gå for denne oppgaven da vi først og fremst er en gruppe med mye praktisk fagerfaring, hvor hver av oss relaterte sterkt til selve oppgaven. Vi så derfor en mulighet til å kunne sette oss inn i et nytt fagområde, samtidig som vi så på det som en utfordrende og spennende oppgave som var relevant for våre fagkunnskaper.

1.2 Problemstilling / prosjektbeskrivelse og resultatmål

Oppgavebeskrivelse, førsteutkast til problemstilling og resultatmål
Problemstilling dere skal utforske i prosjektet. Dette er et første utkast som kan endres dersom premisser endres underveis.
Resultatmålene forteller hva som skal være oppnådd når prosjektet er ferdig.
Oppgaven går ut på å lage en konfigurator for produktet Pleat fra firmaet Hydroniq Coolers. Pleat er en kjøler som blir brukt til å fjerne varmegang/energi i motorer i båter. Produktkonfiguratoren skal ut ifra kalkulasjoner kunne regne ut hvilken kjøler som er best passet til kunden basert på ulike parametere som blir oppgitt. Den skal kjøre på en nettside. I tillegg til å kode kalkulasjonene, må vi utvikle et grensesnitt for selve konfiguratoren, samt utvikle design og andre funksjonaliteter.
Utfordringen blir å kunne få kalkulasjonene fra Excel inn som JavaScript-kode. Utover dette, skal konfiguratoren kunne trenge noen få inputs slik at output blir automatisk generert/utregnet.
Når prosjektet er ferdig, håper vi å ha oppnådd et resultat bestående av en fullverdig produktkonfigurator som vil være i stand til å finne korrekt produkt ut ifra hva kunden ønsker.

1.3 Effektmål

Målet for gruppa vil være å få utvikle et sluttprodukt som vi, oppdragsgiver og potensielle kunder vil være fornøyd med. Videre håper vi at vi vil sitte igjen med god erfaring og kunnskap, som vil kunne hjelpe oss med å utvikle oss selv som ingeniører fremover.
Virksomheten, ut ifra det vi har fått inntrykk av, ønsker å sitte igjen med et sluttprodukt som de kan ta i bruk etter hvert, og som deres kunder vil få stor nytte av. De vil nok håpe på å sitte igjen med en applikasjon som i stor grad vil kunne finne frem til det rette produktet ut ifra kundens parametere.

1.4 Rammer

Behov for penger, utstyr og tid. Spesialbehov materialer og rom.
På nåværende tidspunkt ser vi ingen behov for penger, utstyr eller tid. Kommer det opp noe senere så vil vi gi beskjed.

2. Organisering

Aktører som er involvert i prosjektet er Hydroniq Coolers som er vår arbeidsgiver, NTNU som følger opp prosjektet vårt og 4 NTNU studenter som skal utføre oppgaven.
Studentene har ansvar i å ta kontakt med veileder og arbeidsgiver i jevne mellomrom for å diskutere framgang av prosjektet.

3. Gjennomføring

3.1. Hovedaktiviteter

Hva gjøres, hvem gjør det, hvorfor gjøres det, hvordan gjøres det. Når gjøres det, nødvendige forutsetninger før det kan gjøres, dokumentasjon / resultat av det som ble gjort.

- **Dialog med firma og wireframes til nettside:** Alle er med på dette arbeidet. Vi bruker firma til å tegne skisseløsninger som ligner eksempler gitt av firma, dette gjør vi fordi vi må være sikre på hva firma ønsker før vi starter kodingen. Dette arbeidet blir gjort i første sprint som er uke 3-4.
- **Utforming av nettside (HTML og CSS):** Arbeidet blir lagret og dokumentert på GitHub. Vil bli gjort i sprint 2 som er uke 5 – 6, har fokus på å lage et skall av skissen som blir tatt opp i første sprint møte.
- **Webside funksjonalitet:** Alle skal jobbe med JavaScript arbeidet siden hoveddelen av arbeidet blir JavaScript koding. Kode kalkulering og inputs-fields for konfigurator, samt vise riktig produkt utfra verdiene gitt i konfigurator som er best tilpasset verdiene. (Produkter som ikke kan brukes på gitte verdier skal skjules). Vi blir å jobbe med dette fra sprint 2 og utover.
- **Kontrollere kodekvalitet og dokumentasjon:** Skal gjøres for hver push som blir gjort til repository. Skal skrives en nøyaktig og beskrivende kommentar på hva som er gjort for hver push slik at vi kan gå tilbake å sjå hva som er gjort. Vi vil ta en gjennomgang til slutt av kode dokumentasjon for å sjå at alt er på plass.
- **Dokumentasjon på GitHub:** Issues skal gjøres i forskjellige sprints slik at vi får en jevn arbeidsflyt og har oversikt over hva som skal gjøres. Alle skal plukke og legge til issues etter behov.
- **Ukentlig referater:** Skal skrives ukentlig så det er lettere å gå tilbake å sjå hva vi har skrevet hver uke, kan være en fordel for bachelor skrivingen også. Referater blir skrevet av Kenneth og Vegard.
- **Bachelor-skriving:** Må gjøres en god del forskning på bruk av pleat og platekjølere slik at vi har god kontroll på bruk av produktene til arbeidsgiver. Vi vil først og fremst starte med selve bachelorarbeidet og skrivingen i løpet av sprint 2. Alle NTNU-studenter som er involvert i bachelor-oppgaven skal skrive.
- **Forskning på pleat kjøler og lignende produkter:** Alle skal gjøre sin forskning for å sette seg inn i funksjonalitet og bruk. Vi har satt av de første 2 sprintene til forskning og planlegging av bachelor oppgaven.
- **Sprint planlegging:** Skal ha et møte på slutten av hver sprint hvor oppgaver, oppgavefordeling og planlegging av mål for neste sprint blir utført. Vi møter til sprint møter hver partallsuke fram til uke 20. Sprint planlegging inkluderes i et møte med bedriften hvor vi holder bedriften oppdatert og får tilbakemelding på oppgaver. Oppgavene blir deretter dokumenter til GitHub etter møte med bedriften.

3.2. Milepæler

Opplisting av kritiske datoer.

- Annen hver fredag fra uke 4 – skal vi ha sprint møter og levere referat utfra dette.
- Uke 13. Muntlig presentasjon og status av prosjektet sammen med veileder
- 19. Mai poster og presentasjon
- 22. Mai innlevering av rapporten.

4. Oppfølging og kvalitetssikring

4.1 Kvalitetssikring

For å kvalitetssikre arbeidet vårt, tar vi i bruk versjonskontroll hvor vi ved bruk av ulike grener får en godt organisert plan over arbeidet vårt. Vi har en dedikert gren (main) hvor innhold aldri skal bli lagt til direkte. Denne grenen reflekterer en kvalitetssikret og problemfri kvalitet av arbeidet i vårt "repository". I tillegg til dette, har vi en gren (dev) hvor vi bruker til å teste og kvalitetssikre arbeidet før det eventuelt blir inkludert i hovedgrenen. Nytt innhold til main-grenen kommer kun i form av dev-grene til main-grenen.
Videre, blir grener dannet ut ifra dev-grenen. Disse grenene er mindre problemstillinger og utvikling av spesifikke oppgaver innen prosjektet. Informasjon om disse grenene blir dokumentert på GitHub. Direkte endring av dev-grenen kan bare skje når det er behov for mindre korrigeringer eller å løse konflikter med å slå sammen de ulike grenene.

4.2 Rapportering

Vi planlegger å rapportere om fremdrift til arbeidsgiver annen hver uke slik at bedriften er oppdatert på hva som er gjort og skal gjøres framover. Vi kaller disse samtalene for sprint/oppdaterings møter. Dette skjer på slutten av hver sprint hvor en sprint varer i to uker. Vi bruker e-post med dem om vi støter på problemer som ikke ble tatt opp i oppdaterings møtene som skjer annen hver uke. Når muligheten for oppmøte ikke er mulig så erstatter vi møte med en e-post for å holde bedriften oppdatert.

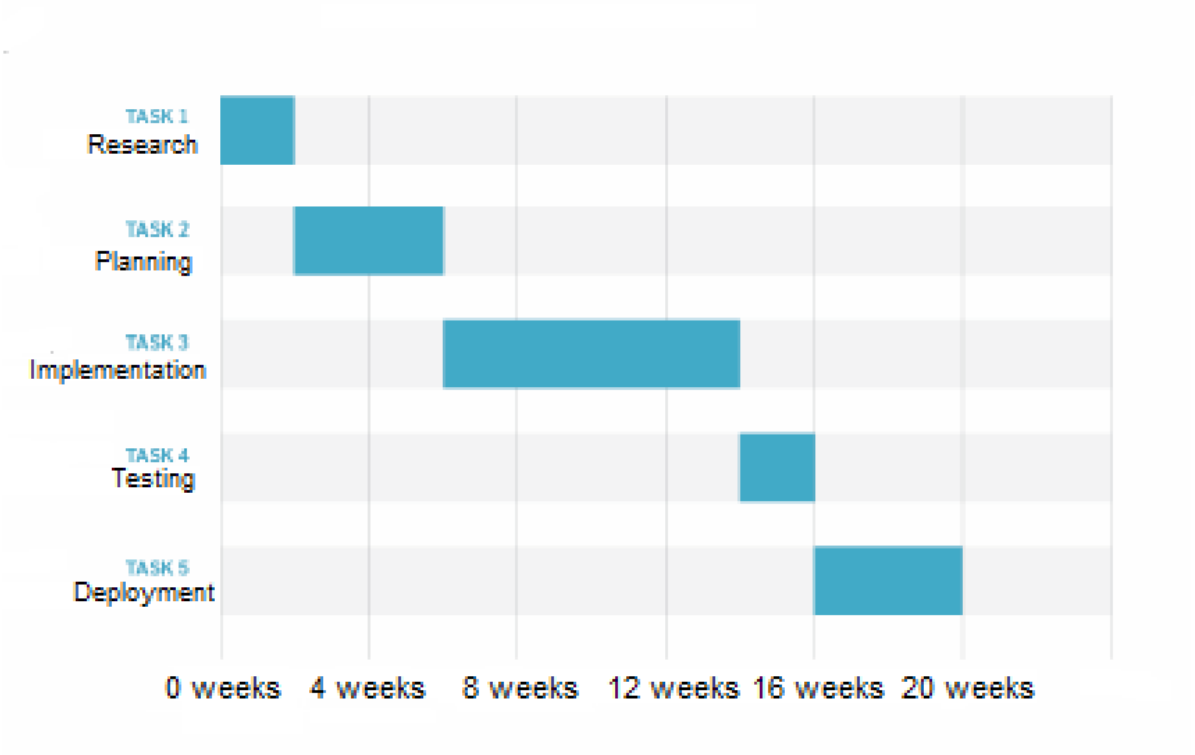
5. Risikovurdering

Vi tar i bruk ulike tjenester for oppbevaring av prosjekt. Dokumentasjon, skissetegninger og diverse som vi føler ikke er naturlig å ikke inkludere i GitHub blir lagret i Microsoft Teams. Teams støtter skylagring og mulighet til arbeide på samme dokument felles.
GitHub blir brukt til dokumentering av kode og koden som er knyttet til prosjektet. GitHub støtter også skylagring. Dette gjør at hele prosjektet er lagret i skyen som hjelper med å forhindre tap av arbeid. Både Teams og GitHub er private for å unngå risikoen for sårbarhet for et tredje parti.
For å videre forhindre konflikter og risiko, har vi avtalt å ha en åpen dialog med bedrift fra start til slutt, der vi har møtes annenhver uke for å ta opp konstruktiv kritikk og innspill.

6. Vedlegg

Følgende dokumenter leveres som separate filer ved innlevering i Blackboard i januar (obligatorisk arbeidskrav), men ikke i endelige leveransen av hovedrapporten den 20. mai!

6.1 Tidsplan



6.2 Avtaledokumenter

6.2.1 Arbeidskontrakt for bachelor-gruppen

Avtale for å definere/ beskrive hvordan bachelor-gruppen skal samarbeide gjennom prosjektet. Mal tilgjengelig på læringsplattformen.

6.2.2 3-partsavtale

Det skrives 3-partsavtaler mellom NTNU, oppdragsgiver og studenter. Mal tilgjengelig på læringsplattformen og innsida. (Standardavtale)

F.

Pleat Konfigurator

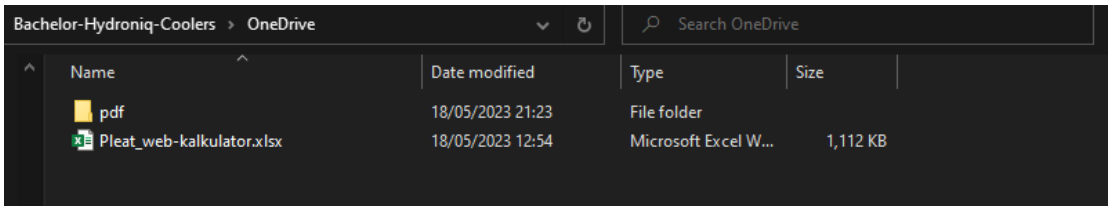
Vedlikeholdsmanual

For ansatte i Hydroniq

1. Vedlikehold- og- videreutvikling

1.1 Vedlikehold av Excel-fil

- For å endre eller bytte ut Excel-fil til konfiguratoren, må filen ha dette spesifikke filnavnet for at filen skal bli godkjent: Pleat_web-kalkulator.xlsx. Navnet kan ikke differensiere i forhold til stor bokstav eller små bokstaver.
- Dersom den oppdaterte Excel-filen skal tas i bruk, må navnet endres til Pleat_web-kalkulator.xlsx før opplastning til mappen. Den oppdaterte filen må da erstatte den eksisterende filen.
- Det er viktig at Pleat_web-kalkulator.xlsx filen alltid ligger i OneDrive folderen. Se eksempel nedenfor.



Figur 1.1

- SORTER-funksjonen i Excel kan ikke tas i bruk noen steder i filen dersom det skal gjøres endringer, da denne funksjonen ikke støttes av programmet.
- Nye produkter kan legges til så lenge eksisterende formler i "Web interface"-arket endres. Dersom nye felt og rader legges til, bør kontaktperson kontaktes slik at dette kan implementeres i programmet.
- Endringer av kalkulasjoner og fremgangsmåter kan gjerne gjøres, så lenge "Web interface"-arkets celle plassering ikke blir endret på noe som helst måte. Programmet er programmert til å lese av de spesifiserte cellene og hva hver celle er.

1.2 Web interface

- En huskeregel er at "Web interface"-arket generelt ikke bør endres, uten at dette er avtalt med kontaktperson. Det er selvfølgelig mulig å legge til rader og kolonner med nye variabler og inputfelt, men det vil derimot ha null funksjon i programmet uten hjelp fra en av oss fra gruppen. Årsaken til dette er fordi at det medfører at ekstra kode må skrives i ExcelServiceImplementation-klassen i applikasjonen. Denne klassens eksisterende kode har spesifisert hvilke celler som skal leses av.
- Grensesnittet ligger på "Web interface"-arket i Excel-filen, og de relevante feltene gjelder cellene på rad 2-12 (inputfelt) og 44-53 (resultattabell).

Avtale med ansvarlig fra gruppen dersom endringer skal gjøres. Feltene som blir lest av i programmet kan ses i figur 1.2, og dersom nye felt legges til her må disse endringene programmeres.

	Pris	Description	First calculation	Second calculation	Tørrvekt (kg)	Vekt operativ (kg)	Sice class	Passes
1		DN100/125, 1-pass with 3 elements. Recommended port size DN125.	OK	OK	415	505	2	1
2		DN150/200, 1-pass with 2 elements. Recommended port size DN150.	OK	OK	595	695	3	1
3		DN100/125, 1-pass with 4 elements. Recommended port size DN100.	OK	OK	455	575	2	1
4		DN150/200, 1-pass with 3 elements. Recommended port size DN150.	OK	OK	645	795	3	1
5		DN100/DN125, 1-pass, 5 elements. Recommended port size DN100.	OK	OK	485	635	2	1
6		DN150/DN200, 2 serial passes with 2 elements each pass. Recommended port size DN150.	OK	OK	698	898	3	2
7		DN100/DN125, 2 serial passes with 3 elements each pass. Recommended port size DN100.	OK	OK	523	703	2	2
8		DN100/DN125, 2 serial passes with 4 elements each pass. Recommended port size DN125.	OK	OK	563	803	2	2
9		DN150/DN200, 2-pass with 3 elements each pass. Recommended port size DN150.	OK	OK	798	1098	3	2
10		DN150/DN200, 3-pass with 2 elements each pass. Recommended port size DN150.	OK	OK	811	1111	3	3

Figur 1.2

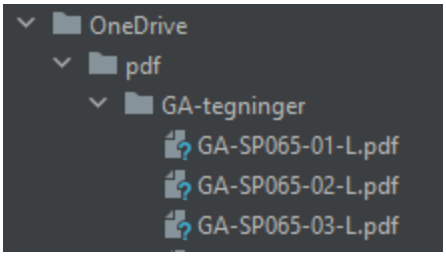
- Cellene i første kolonne i figur 1.3 brukes til å kalkulere ut produkter. Dersom noen av disse inputene fjernes eller flere legges til, må en person fra utviklergruppen kontaktes. Dette krever kodeendringer i applikasjonen, så prøv å unngå dette med mindre annet er avtalt.

	A	B	C
1		**Inn-verdier for web-kalkulator**	
2		80	Flow, Fresh water (hot fluid)
3		0	Heat transfer
4		80	In-temperature, Fresh water (hot, before cooling)
5		50	Out-temperature, Fresh water (hot, but cooled)
6		32	In-temperature, Sea water (cold)
7		0	Weight percent antifrogen (ethylene glycol)
8		0	Bio-fouling factor
9		3.5	Salinity, Sea water
10		1.5	Max pressure drop, Fresh water
11		1.5	Max pressure drop, Sea water
12	90	60	Max temp out, sea water

Figur 1.3

1.3 Opplasting av GA-tegninger

Dersom GA-tegninger skal oppdateres eller legges til, må dette skje i plasseringen OneDrive -> pdf -> GA-tegninger (se figur 1.4).

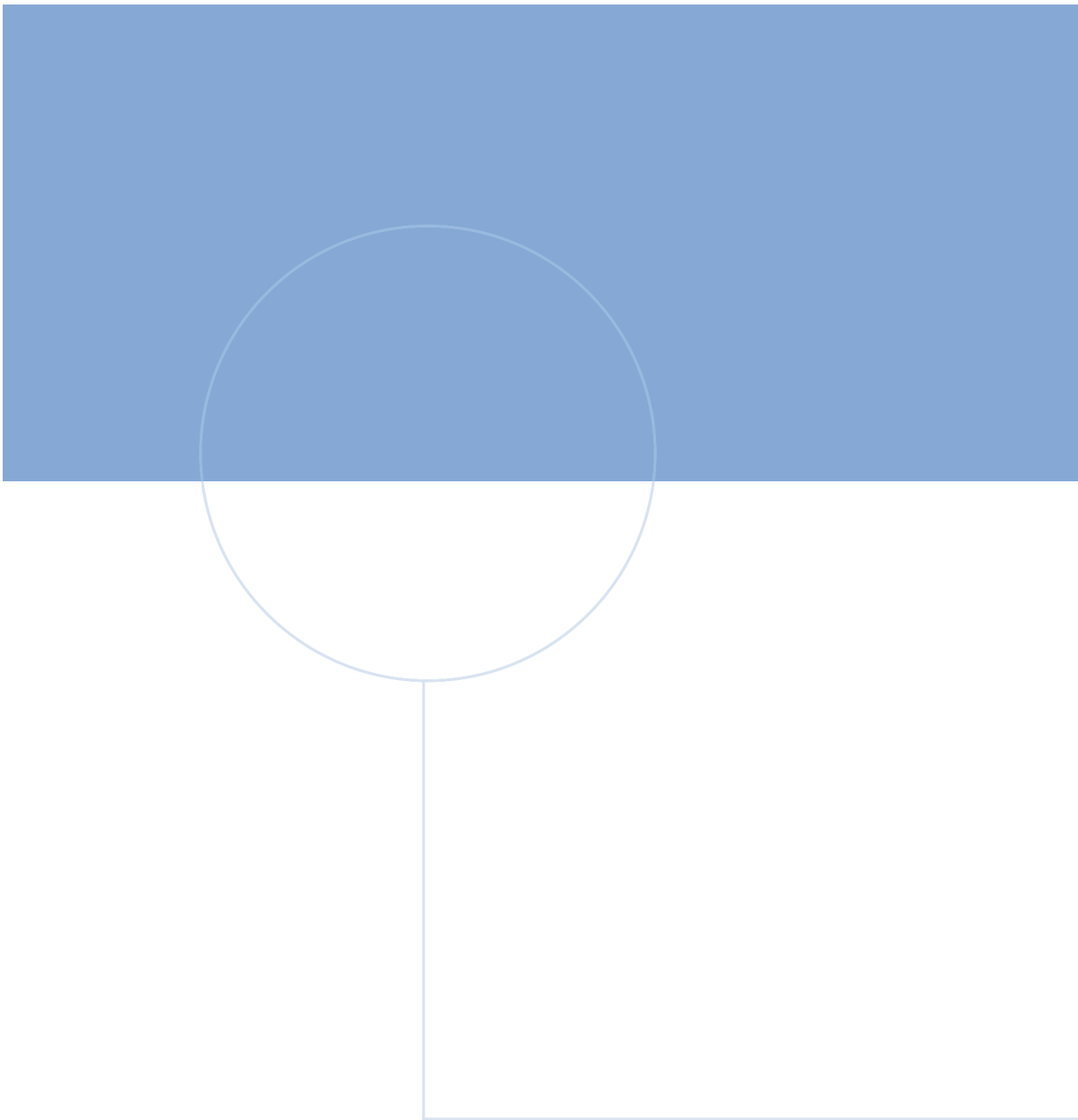


Figur 1.4 Plasseringsstruktur for GA-tegninger

Her må filnavnet skrives tilsvarende slik som en ser på bildet over. Navnet vil alltid være "GA-SPxxx-xx-L.pdf", hvor x erstattes av et tall. De første tallene etter SP vil være kjølermodellen, altså størrelsen. Dersom en har en oppdatert GA-tegning på en DN65 kjøler med 1 element, må en navngi PDF-filen GA-

SPxxx-xx-L, hvor de første tre x-ene erstattes med 065, og de siste to erstattes med 01 (antall element). En ser eksempel på filnavn på DN65-modeller med både 1, 2 og 3 elementer i figuren ovenfor.

Dersom en skal laste opp en tegning av et produkt som allerede eksisterer, må den eksisterende PDF-filen erstattes med den nye PDF-filen. Det er ingen problem å legge til nye PDF-filer så lenge filnavnet følger navnekonvensjonen. Programmet tar automatisk i bruk nye tegninger etter hvor lang tid nedlastingen tar med OneDrive.



NTNU

Kunnskap for en bedre verden