NTNU

Norwegian University of
Science and Technology

Development of a General Purpose Gamification Framework

Eivind Vea

Master of Science in Computer Science
Submission date: July 2016
Supervisor: Magnus Lie Hetland, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Development of a General Purpose Gamification Framework

Eivind Vea

July 21, 2016

Abstract

This report describes the design and implementation of a general purpose gamification framework developed in JavaScript on the Metor platform. Gamification is described as the use of game elements in none-game contexts. The purpose is to encourage and change user behaviour. Examples of existing gamification use cases and frameworks are described. A demo game shows how a general purpose framework can be used.

Abstract

Denne rapporten beskriver design og implementasjon av et generelt rammeverk for gamification utviklet i JavaScript på plattformen Meteor. Gamification er beskrevet som bruk av spillelementer i sammenhenger som ikke er spill. Formålet er å oppfordre til handling og å endre brukeradferd. Det blir vist eksempler på bruk av gamification og eksisterende rammeverk. Et eksempelspill viser hvordan et generelt rammeverk kan brukes i praksis.

Preface

This Master's thesis is the result of my Computer Science studies at Norwegian University of Science and Technology (NTNU) in the spring semester of 2016.

I would like to thank my supervisor, associate professor Magnus Lie Hetland for encouragment at all times. I have really enjoyed our always interesting weekly conversations during the project period.

I also would like to thank my wife Kris for support, patience and setting your own studies on hold while I've been working.

And finally, Ask, Idunn and Vegard - I can recommend studying Computer Science when you one day leave home.

Contents

Chapter 1

Introduction

The goal of this Master's project is to develop a framework for gamification which, in time, can be extended to be a full-fledged gamification engine. This project report describes my work with developing the Meteor-based working prototype "MondoLudo".[1]

Gamification is a term which has gained much attention recently. The most common definition of gamification is "the use of game design elements in non-game contexts". (Sebastian Deterding et al 2011)[1]

Even though it is a quite new term, there is done increasingly more research in the field. I have used the comprehensive anthology "The Gameful World"[2] as a resource in getting to know the subject.

There are numerous appliances of gamification in action, and some of these are described in this report. Surprisingly, there appears not to be any gamification frameworks that can be used by anyone to create their own gamified systems.

The next chapters discuss the concept of gamification, the features of MondoLudo, the technology which is used to make the prototype, the development of the prototype, an example of a created game, and further work that can be made based on the prototype at a later stage.

[1] Mondo (latin "World"), Ludo (latin "Game"))

1.1 Motivation

Games are increasingly popular in most demographics. In the official statistics for 2015, 36% of the Norwegian population play games daily, and as many as 80% of boys aged 9-15 and 73% of girls in the same age group play digital games on an average day. (Statistisk sentralbyrå / Statistics Norway, 2015)[3]

There is an ongoing decrease in the same groups' time spent on TV and movies, both in Norway and abroad. (Entertainment Software Association, 2015)[4]

Since playing games is so common, there is a possibility that users also would like to immerse in a gamified real world experience, given that the game mechanics don't differ too much from regular games. This is my main reason for making a gamified framework. The actual gameplay should be intuitive and easy to understand for all users.

The web application Kahoot![5] is an easy way of setting up and playing games in teams. It was initially based on a Master's thesis at Norwegian University of Science and Technology (NTNU). Kahoot! lets users create games that can be played with friends, where the game questions are displayed on a screen, TV or projector, and the participants use their mobile phones to guess the right answer.

However, although increasingly popular, Kahoot! might be regarded as a game based learning platform and not a flexible gamification engine. It nevertheless serves as an example of a successful application which lets anyone set up and play games. In the era of social media, playing simultaneously with friends in a social context fits right in. Its gameplay is easy to understand, which makes the platform suitable for anyone. This should also apply for MondoLudo

Motivation for making a general purpose gamification framework:

- No such thing is currently available
- It could be of interest to a lot of people
- Wide range of use cases
- Possible commercialization

1.2 Examples of gamification

Gamification is found in a wide range of applications. Not all of them are exclusively on platforms like personal computers or mobile phones. Many tasks can be gamified. The goal for gamification is to encourage a desired behaviour which in turn can lead to actual real world change.

Arguably the first known gamification system is that of Napoleon Bonaparte in 1795, when he first offered a reward for the person that could come up with a solution for food preservation. This ultimately lead to the invention of canned food[2]. In addition, soldiers at the time were usually paid, but money was scarce and Napoleon instead started to award medals to the soldiers who were regarded as being the bravest and strongest. This incentive allegedly motivated his armies to perform better[6].

1.2.1 Rewards

The actual gameplay can be made in numerous ways, but common for most of them is that the users get rewards or motivation during the game. The simplest form is “pointification”, which means that the users gain points when playing.

Other rewards can be leveling, unlocking of features, awarded experience points in different areas, or leaderboards showing other players’ results. All these forms of rewards can be found in regular games. The main difference in a gamified experience is that the game itself is not the main focus, it exists merely as a tool to lead to a change of behaviour outside the game.

I present three current examples of gamification in this section.

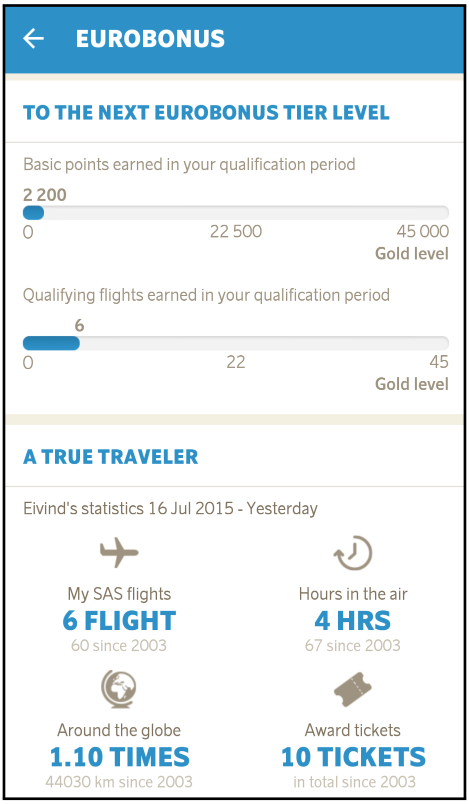
1.2.2 EuroBonus

The frequent flyer program EuroBonus for the airline SAS and the Star Alliance has had gamified elements at least since the 1990s. By collecting flights, a EuroBonus member can accumulate points for later use as payment for bonus flights or accomodation.

There are several tier levels: Member, Silver, Gold and Diamond, all with different real world rewards. You get a level upgrade based on how many flights or how many points you acquire during a one year period. This is a good example of the concept of “pointification”, where the gameplay itself is based on nothing more than collecting points.

[2] Napoleon III ran a similar challenge in 1866, which led to the invention of margarine.

Such frequent flyer programs are not new, but the advent of digital platforms means that it is easier to implement and show gamification elements than it used to be earlier. For instance, on SAS' corresponding mobile app, you will see the gamification in action: The requirements for moving up one level are easily found, displaying how many points and flights the user has accumulated. In addition statistics like total distance travelled and airborne hours are shown, which also could be considered to be gamified elements encouraging more travel. By having frequent flyer programs like EuroBonus, you can build loyalty and get people to travel with your airline instead of competing carriers.



EUROBONUS

TO THE NEXT EUROBONUS TIER LEVEL

Basic points earned in your qualification period

2 200

0 22 500 45 000

Gold level

Qualifying flights earned in your qualification period

6

0 22 45

Gold level

A TRUE TRAVELER

Eivind's statistics 16 Jul 2015 - Yesterday

My SAS flights
6 FLIGHT
60 since 2003

Hours in the air
4 HRS
67 since 2003

Around the globe
1.10 TIMES
44030 km since 2003

Award tickets
10 TICKETS
in total since 2003

Figure 1.1: Screenshot from the SAS mobile app

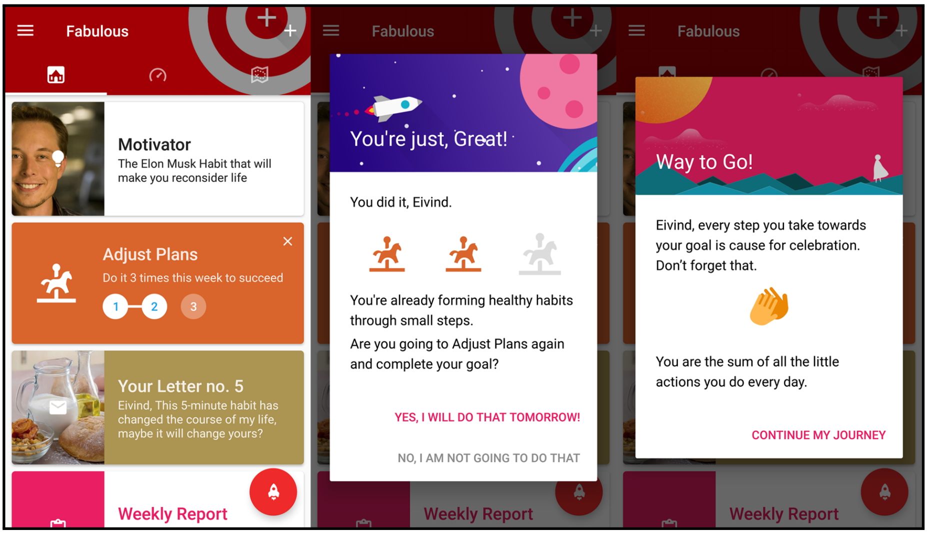
1.2.3 Fabulous

Fabulous[7] is a personal motivation app for Android. It is marketed as a science-based app incubated in Duke University's Behavioral Economics Lab.

The main goal is to give its users the opportunity to learn how to get things done by defining and following rituals at set times throughout the day.

On the Fabulous website there is this description of the system:

> *We're building a tool to help you reset your habits, rewire your brain and transform your life.*



Motivator

The Elon Musk Habit that will make you reconsider life

Adjust Plans

Do it 3 times this week to succeed

Your Letter no. 5

Eivind, This 5-minute habit has changed the course of my life, maybe it will change yours?

You're just, Great!

You did it, Eivind.

You're already forming healthy habits through small steps.

Are you going to Adjust Plans again and complete your goal?

Way to Go!

Eivind, every step you take towards your goal is cause for celebration. Don't forget that.

You are the sum of all the little actions you do every day.

Figure 1.2: Screenshots from Fabulous' Android app

The gamified aspect of Fabulous is hidden; the user doesn't know anything about what will happen next. There are no points to collect, or levels to achieve. Instead, the app will at set times introduce new challenges according to what the user marks as completed. The user gets encouraging messages to continue using the app.

The players will be suggested new goals for their routines. It starts easy, like buying a notebook to use for making plans and todo-lists every day. Later it will introduce meditation as a new task to do before bedtime.

A key feature is that the app does not rely solely on text; it also uses speech synthesis to let your virtual mentor speak directly to you. It also has ambient background music while you perform your tasks.

1.2.4 Nissan Leaf

The electric car model Nissan Leaf has a gamified experience regarding eco-friendly driving. Their system is called Carwings.

By driving energy efficiently, a symbol of a tree on the front display will grow and change according to factors like speed and power usage. These driving metrics can be uploaded so the driver can compare her driving to that of others Carwings users both nationally and world wide.

Carwings uses a level system with badges and trophies, which the players can achieve every month. It also has leaderboards where drivers can see how efficiently they drive. Some statistics are shown directly on the main display in the car, while others more thorough statistics can be found on the system's website.

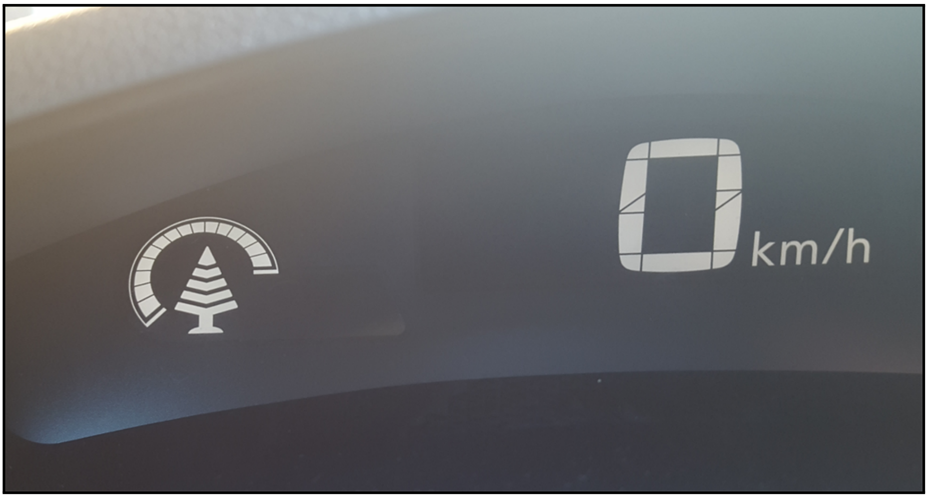
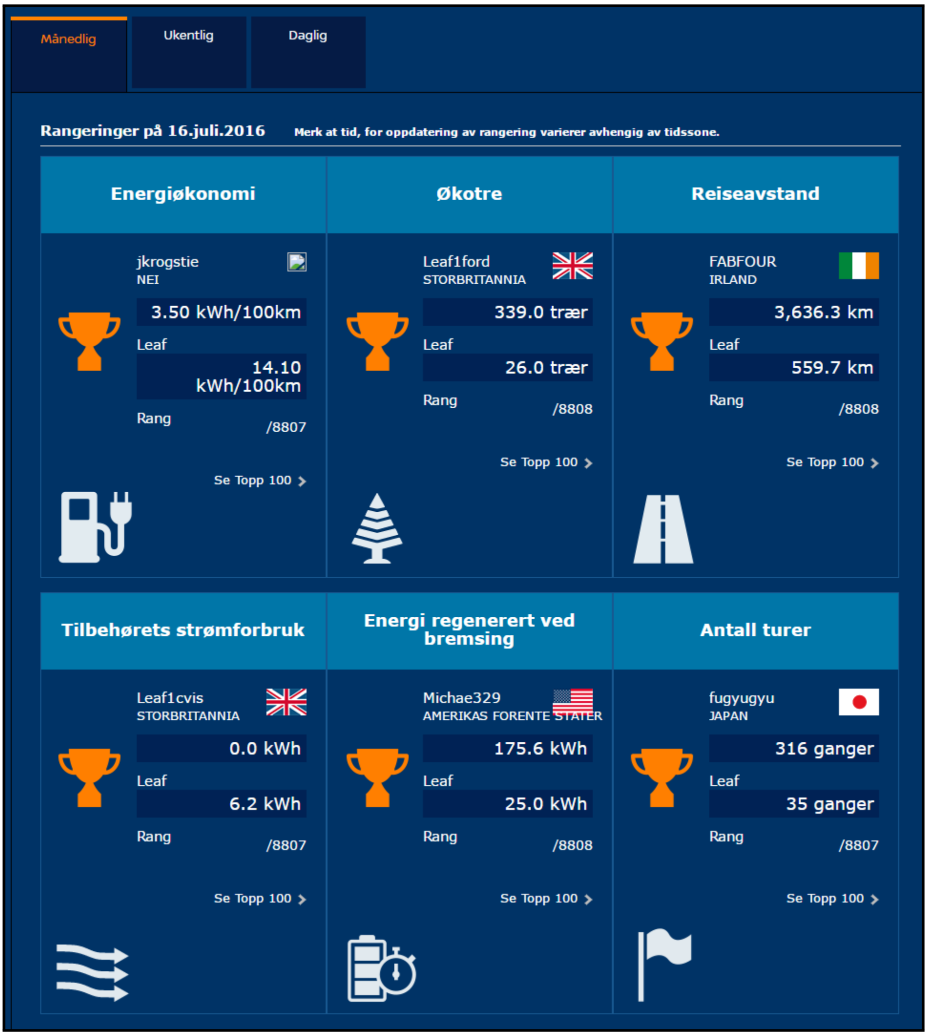


Figure 1.3: Nissan Leaf's tree on the drivers' display



Månedlig Ukentlig Daglig

Rangeringer på 16.juli.2016 Merk at tid, for oppdatering av rangering varierer avhengig av tidssone.

Energiøkonomi

jkrogstie
NEI
3.50 kWh/100km
Leaf
14.10 kWh/100km
Rang /8807
Se Topp 100 >

Økotre

Leaf1ford
STORBRITANNIA
339.0 trær
Leaf
26.0 trær
Rang /8808
Se Topp 100 >

Reiseavstand

FABFOUR
IRLAND
3,636.3 km
Leaf
559.7 km
Rang /8808
Se Topp 100 >

Tilbehørets strømforbruk

Leaf1cvis
STORBRITANNIA
0.0 kWh
Leaf
6.2 kWh
Rang /8807
Se Topp 100 >

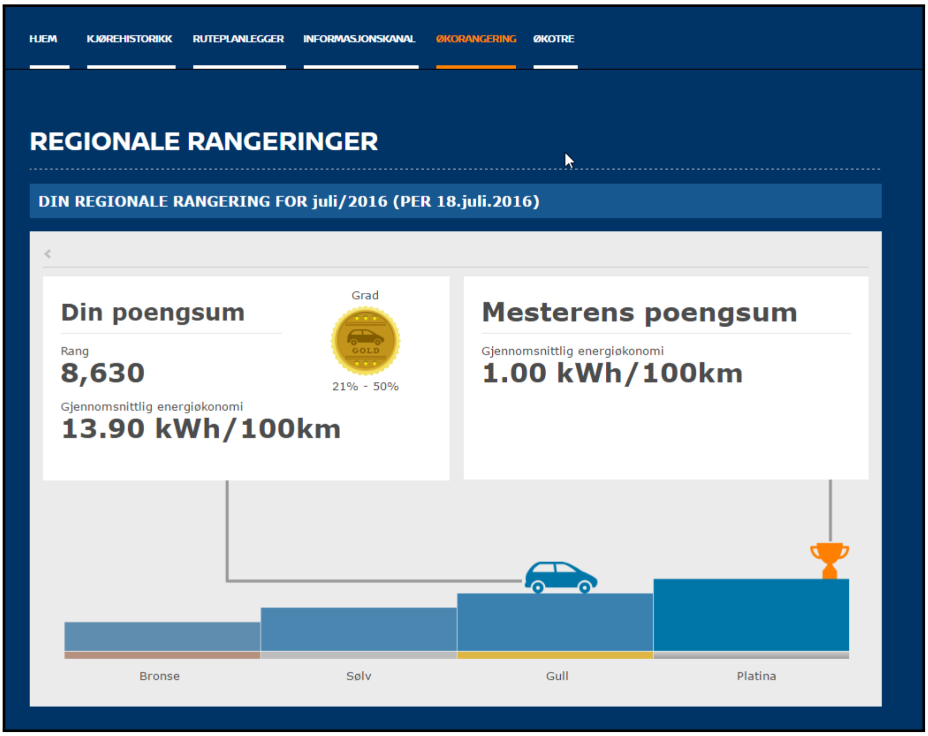
Energi regenerert ved bremsing

Michae329
AMERIKAS FORENTE STATER
175.6 kWh
Leaf
25.0 kWh
Rang /8808
Se Topp 100 >

Antall turer

fugyugyu
JAPAN
316 ganger
Leaf
35 ganger
Rang /8807
Se Topp 100 >

Figure 1.4: Carwings rankings



REGIONALE RANGERINGER

DIN REGIONALE RANGERING FOR juli/2016 (PER 18.juli.2016)

Din poengsum

Rang
8,630

Grad
21% - 50%

Gjennomsnittlig energiøkonomi
13.90 kWh/100km

Mesterens poengsum

Gjennomsnittlig energiøkonomi
1.00 kWh/100km

Figure 1.5: Carwings player achievements

1.3 Framework

The framework should have the following features:

- Available to anyone with a computer or smart phone
- Available for free
- Easy to use
- Developed in a common programming language
- Scalable
- Fast

- Easy to maintain
- Possible to commercialize
- Fun!

My framework has a few terms that should be defined.

Term	**Description**
`MondoLudo`	The gamification framework.
`Game`	Could also be called quest, adventure, task or something else, but in this context I refer to any gamified project as a game.
`Gamemaster`	The person running the game, in this prototype also the creator of the game.
`Player`	The user playing the game.
`Players`	The set of users playing a game.
`Teams`	A group of players that are playing together.
`Levels`	The levels of the game. For now these are linear and incremental, but they could also be more complex and nonlinear so that the next level could be decided based on user input while playing the game.
`Confirmation`	The feedback a player gets when a level is completed.
`Rewards`	Achievements when completing a level or game. These can be in the application itself or describe a real world reward.
`Points`	The number of points, if any, the user gets when completing a level or game.

Table 1.1: Definition of framework terms

Chapter 2

Context

When starting a project, one should strive to find out what solutions currently exists, as well as deciding which platforms and tools to use. In MondoLudo this applies both to the design of the gamification framework and the technical implementation.

2.1 Existing gamification frameworks

For the process of designing a gamified experience, there exist several frameworks that can be used.

Yu-kai Chou's Octalysis[8] is an example of how a gamification framework can be a useful tool for game creation. It is based on analysis of regular games, and places terms and concepts from games into eight categories. These are displayed in an octagon, where you visually can see the different metrics of a game based on how they score on each of the eight axes. You can use this framework for analyzing existing games as well as using it as a tool for designing a new game. In my opinion, this works well, and Octalysis' website offers analysis of a lot of popular regular games in addition to well known gamified experiences.

Andrzej Marczewski's Gamification Framework[9] is another tool for describing gamified systems. The framework is a step for step approach for the planning, design, execution and evaluation of a gamified experience. This is a different approach from Octalysis, but can serve as a useful checklist when making a game.

Zachary Fitz-Walter's PhD-thesis "Achievement Unlocked: Investigating the Design of Effective Gamification Experiences for Mobile Applications and Devices"[10] introduces yet another framework for the design of games. In this thesis, the framework is a set of actions that should be followed when designing a game. Step 1 is to "justify the motivation and requirements", step 2 is to "design the gamification experience", concluding with step 3 where one should "evaluate the effectiveness". He uses these steps to make two working gamified mobile apps.

All of these frameworks can be useful as guidelines and rules for how to make games. However, my goal is to make an easy to use technical framework for gamification, and not making the actual games themselves.

The company Funifier[11] offers gamified solutions for the business market with their platform Funifier Studio, but they dont't have any general and free solutions available for public use. The customer can set up a gamified system by choosing amongst pre-built widgets, like challenges, levels, badges, leaderboards and other game elements. In addition, Funifier Studio can be used for evaluation and analysis of games made on that platform. The games can be integrated on websites, mobile apps and even Microsoft products like Word and Excel. The system is powerful, but it is way too complex and expensive to be used by anyone.

There are numerous applications utilizing gamification in a wide array of fields, but I still haven't found what I'm looking for - a free web based gamification framework where users can make their own gamified applications to share with friends.

2.2 Why Meteor?

The JavaScript framework Meteor is a full-stack platform for building web and mobile applications. I briefly tested Meteor a while back and found it to be an interesting technology that I wanted to explore further.

By using Meteor, users can focus on coding actual functionality instead of writing a lot of code for tasks that are necessary in all web and mobile applications of this kind. Such tasks can be user login handling and administration, database functionality and client-server interaction. Meteor offers available add-on packages for a wide range of tasks.

Meteor features:

- Pure JavaScript on both client and server
- Many built in features, e.g. client-server communication based on DDP (Distributed Data Protocol)
- Real time updating on all clients
- Option to compile to native formats for iPhone and Android, which means easy distribution through App Store and Google Play

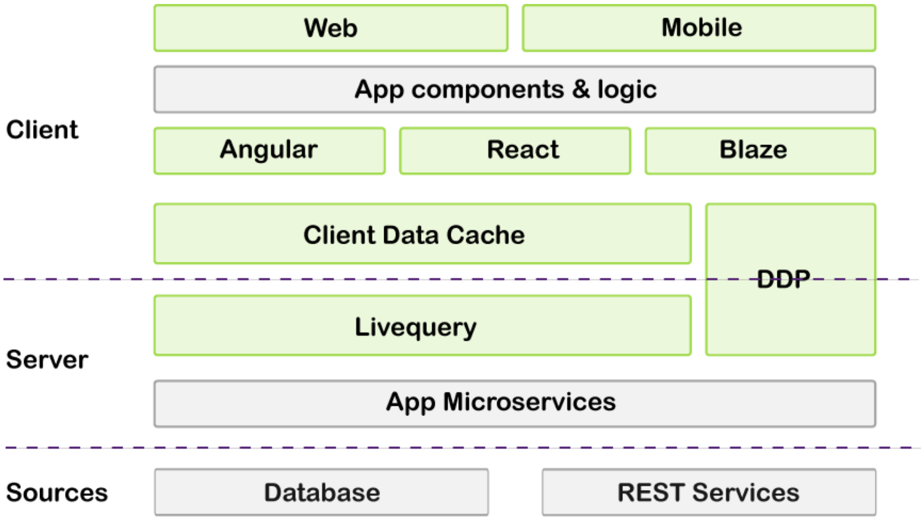


Figure 2.1: Meteor framework overview

2.2.1 Licence and organization

Meteor is open source and MIT licensed. When chosing software, one should always make sure that development will continue. An example of this is Adobe Flash, which had widespread use and ran on practically all platforms, but is now more or less dead and deprecated in most areas.

In Meteor's case the organization behind it (Meteor Development Group) is well funded[12], and it profits from hosting Meteor-based systems. Combined with a continuously more wide spread use, ongoing development of add-on packages and documentation, and a strong user community, I think that Meteor will be more than yet-another-framework and be around in the future.

2.2.2 JavaScript

One could argue that there are other options than JavaScript and Meteor. Any other back-end programming language could be used, like C# or Java. My view is that knowledge of JavaScript will be necessary for all programmers, as this is the only language used in all web browsers. The advent of Node.js means that JavaScript also can be used server side. This means that a JavaScript based solution is making sense when it comes to decide which platform to use.

A JavaScript platform is therefore suitable both for my work with MondoLudo, as well as for gaining experience that can be put to use in later work as a programmer. With the impact of HTML5 and CSS3 as client interfaces in a wide range of applications, it will be increasingly important to master these skills.

I'm experienced in HTML, CSS, jQuery, PHP and WordPress, and although never seriously considering this as a viable option, it would - maybe surprisingly - be possible to make MondoLudo on that platform as well. WordPress is not just a CMS, but a powerful platform which can be used also for other purposes.

2.2.3 Reactive rendering libraries

To make updates of data appear instantly in the clients, one would use a reactive rendering library. Any changes on the server will automatically also be updated for all clients with access to this information.

Meteor has its built-in reactive rendering library called Blaze. There is also the possibility to use Angular (created by Google in 2010) or React (created by Facebook in 2013) instead of Blaze.

Blaze is easier to use than the others. For instance, React lacks the "Spacebars" notation, meaning that reactivity is based on using the JSX format to add JavaScript directly in the HTML code. Angular uses HTML with special attribute syntax for logic and events. Performancewise Angular 2 is best, React and Angular 1 are runners up, while Blaze is the least efficient [13].

Most official Meteor documentation and tutorials as well as textbooks are based on Blaze, so getting a project up and running from scratch is faster than if you use Angular or React. On the other hand, Angular and React has other advantages. One feature of React is strict data control mechanisms, which in turn is leading you to follow better programming practices.

The learning curve is steeper with Angular or React, but ultimately it is more powerful and offers easier maintainability and componentizing code [14]. While Blaze is Meteor specific, Angular and React are also used on other platforms.

There are rumours that Blaze version 2 will act as a wrapper for React, but as of writing this report this is not implemented, and according to discussion threads in the official forums, nobody is certain that there will ever be a version 2 of Blaze [15].

Blaze is easier both to learn and to use but might be deprecated (or at least not encouraged), Angular is a good alternative, but most online sources agree on React being the desired Meteor UI rendering library in the future.

My decision is still to use Blaze, mainly because of ease of use and the excellent documentation and books available[16][17].

2.2.4 Blaze

Blaze is Meteor's built in client rendering system. It uses the "Spacebars" notation with curly braces defining objects. A simple example is `{{object}}`. This can be used to let JavaScript render data in the client.

The corresponding client side code is made with template helper functions that perform an operation and return data, in our setting usually a string. These functions are the links both to session specific context as well as methods defined on the server side.

Furthermore, it is used to define the inclusion and order of templates in the HTML-file in the same manner. One would for instance use Spacebars-code, e.g. `{{>template}}`, to include a template at a given place in the HTML code.

In addition, curly braces are also used for control flow and loops in the HTML-code, like if-else-conditionals and each-in loops.

Blaze lets the developer focus on coding the app's core functionality, with all the behind the scenes features and heavy lifting done by Meteor.

2.2.5 MongoDB

Meteor uses MongoDB as its database. This is an open source NoSQL database. Unlike relational SQL databases, it has collections instead of tables, and documents instead of rows. Entries are basically JSON objects.

Querying Mongo is done using the functions “find()” which returns a cursor, and “findOne()” which returns an object. The cursor is a reactive data source, and not the collection itself. The cursor can be used directly in the HTML templates by using `{{#each cursor}}` to iterate through the results.

On the client, Meteor uses Minimongo, which works as a cache of the Mongo database. Client queries are done in this cache, and not directly on the server.

2.2.6 Apache Cordova

Meteor integrates with the open source mobile application development framework Apache Cordova. This makes it possible to develop apps that also can be run on iOS and Android. Cordova employs HTML5, CSS3 and JavaScript to wrap the Meteor code into apps that can be distributed on Apple Store and Google Play.

Standard web apps could be an alternative option, but then the actual client code has to be downloaded every time the app is used. Cordova will instead bundle all the client side code, leading to loading times way faster than for web apps. In addition, Cordova offers hot code push, which means that updating the apps does not involve releasing new versions on Apple Store and Google Play.

Native apps are extremely time consuming to develop, and the main reason for making these native apps is usually to get access to phone specific features. Cordova offers plugins to access these features, like for instance device cameras or sensors.

2.3 Gameplay

MondoLudo should as far as possible be designed with a gameplay familiar to regular users. In my first prototype, that means that it is a single page app, where content is changed and updated without any page loads.

2.3.1 Challenge and response

I've used the terms “challenge” and “response” to describe the gamemaster's definition of the game tasks, and players' game feedback. The challenge could literally be describing any sort of action that is possible to track. Some examples are shown in table 2.1.

In its simplest form, this could be a question and an answer, but MondoLudo should also have functionality to describe more advanced tasks, as well as responses that are not merely textual answers.

A more complex usage could for instance be to use a mobile phone's sensors to respond to a challenge. Zachary Fitz-Walter's PhD-thesis[10] describes an app used for orientation on a university campus, where newly enrolled students use the app to get acquainted. This could be made in MondoLudo too, and an interesting feature could be to automatically check in a player at events, meaning that the player should be at a given place at a given time.

Motivation is key for gamification. An app should be able to automatically encourage its players into trying to reach their goals. This could be textual, by gaining points, and also according to hidden criteria defined by the gamemaster. The players shouldn't necessarily be aware of the behind the scenes mechanics. There is also a fine line between being encouraging and being nagging, so care must be taken when setting up these criteria.

However, in this first version the default usage is questions and answers, as the implementation of other responses, like for instance geographical coordinates or scanning of QR-codes are a little more challenging and time consuming to implement.

There should also at a later stage be implemented a simple chat client where gamemaster and player can communicate during the game. In addition, players on the same team should be able to communicate in the same manner.

2.3.2 User tracking

I have implemented a simple tracking functionality of users where progress in a game is shown. The gamemaster will se how many levels a player has completed. This should be extended so that both gamemaster and users can be informed of game status in real time. The gamemaster should at all times have the possibility of checking the progress of all players and teams, while players should see their own status and optionally those of co-players and teams. The order of the players should be sorted according to game completion.

The tracking should show which level a player is on, how many points he

Challenge	**Response**
`Textual question`	Textual answer
`Multiple choice`	Chosen element
`Location`	GPS location on map
`Location`	Scanned QR code at location
`Location`	Picture taken at location
`Image`	Response of some type that has to be accepted or declined by the gamemaster
`Sound clip`	Response of some type that has to be accepted or declined by the gamemaster
`Video clip`	Response of some type that has to be accepted or declined by the gamemaster
`Generic`	Response of some type that has to be accepted or declined by the gamemaster

Table 2.1: Examples of challenges and responses

has achieved, a percentage of correct answers, how many tries the player has used on a given level, and so on. The gamemaster can use this tracking also as a tool for granting completion of a level if a player of some reason is stuck and can't unlock the next level himself.

Furthermore, by using Google Maps it is possible to implement a map showing players' location while playing a game. This could for instance be used for a scavenger hunt where players physically should visit real world locations. When reaching a given destination, the next level will be activated, showing the next destination in the game. The gamemaster will at any given time see the physical location of players. The recent extremely popular game Pokémon Go shows that players will engage in such games.

2.3.3 Push notifications

MondoLudo should have the possibility to send user updates to mobile clients while the app is not running. The main motivation for enabling push notifications is to give the player reminders and encouragement while playing a game. In addition push notifications can be a tool to show other users' progress, both for single team players, competing players or teams, as well as giving the gamemaster real time simple updates of players' progress without having to run the app at all times.

There are a couple of open source push notification libraries available for Cordova apps built using Meteor. This is a work in progress, and as of writing there are at least two ways of implementing this. For Android one can use the library "meteor-cordova-notification"[18], and for both Android and iOS and several other platforms the library "raix:push"[19] can be used. Both libraries are hosted on GitHub.

Chapter 3

Design and implementation

I've decided to program the prototype in pure JavaScript, meaning that I've avoided the usage of jQuery completely even though this is supported in Meteor.

Certain tasks would have been easier to implement with jQuery, but I think it is a good thing to avoid external libraries or frameworks as much as possible code wise, given that it is easy to implement similar functionality without them.

I've used Donald Crockford's book "JavaScript - The Good Parts"[20] as a valuable resource to help with best practices and which issues to avoid.

3.1 Definition of features in MondoLudo

I have defined the following key features of MondoLudo.

User:

- Create user based on Facebook credentials
- Create friend list for user
- Create game list for user
- Login/logout

Games:

- Create game

- Save game
- Add users to game
- Show progress for user (points or number of challenges done)
- Set if game is public or private

Teams:

- Create team/remove/invite
- Let user add friends to team
- Mark user as admin

Other:

- Track progress (gamemaster can see players' status)
- Leaderboard pr game
  - users
  - teams
- Generate QR-code for invites and in-game usage

Media:

- Upload images
- Upload sound clips
- Google Maps

Gamemaster:

- Add Facebook friends to game
- Add game
- Add questions and answers for game

Player:

- Play game
- Respond to challenges
- Communicate with gamemaster

3.2 Setting up a Meteor environment

Starting working on a new platform often is cumbersome and frustrating (Android app development springs to mind). Meteor, on the other hand, is easy to set up and run.

3.2.1 Installation

Meteor can be installed from the terminal on OSX and Linux, and from running an installer on Windows. Meteor commands and package installs are run from the command line.

By entering the following commands

```
meteor create mondoludo
cd mondoludo
meteor
```

you will have your first app up and running on `http://localhost:3000`. No additional actions are needed.

3.2.2 File structure

In a Meteor project, not many files are necessary to for making an app. I've based my work on the documentation available, and kept the number of files to a minimum, but for more complex versions in the future, it might be useful to group code in different files.

Meteor projects have a default folder structure, where client side code is placed in the folder "client" and server side code is placed in the folder "server". It's also possible to mix the two in the same file, but for clarity this folder structure is used. Assets are placed in the "public" folder.

Filename	**Description**
`mondoludo.html`	Contains the HTML code for the application. This includes the Meteor-spesific templates based on Meteor's templating engine Blaze.
`mondoludo.scss`	Contains the visual styling in "Sassy CSS"-format. This extends plain CSS3 with some much needed functionality, like the use of variables.
`mondoludo.js`	Contains the definition of MongoDB collections.
`client.js`	Contains all the JavaScript functionality for client side use, and describes control flow and what data should be rendered in the HTML code.
`server.js`	Contains server side code, mainly the publication of the MongoDB collections.
`methods.js`	Contains server side code serving as an interface from the client to the manipulation of MongoDB. This is mainly because one does not want direct database operations exposed on the client side, but instead wrap all of them as methods implemented with user access control.
`facebook.js`	Contains code needed to use Facebook for user authentication.

Table 3.1: Description of source files

3.3 Implementation

The actual implementation of the prototype was a quite new experience. Earlier I have mostly used jQuery and not pure JavaScript, and not in such a comprehensive project. I have also never used MongoDB before, but this proved to be very straightforward to use, since the application is not very database heavy and defining and querying collections is simple to achieve in Meteor.

3.3.1 Packages

There are numerous packages available for a wide array of tasks. A standard Meteor project is from scratch set up with several built in default packages, so additional packages are not necessarily required.

From Meteor 1.3 on it is possible to use npm packages instead of Meteor specific ones from their package system called Atmosphere. According to Meteor's web page[21] all packages will be migrated to npm in the future.

In MondoLudo I have used the packages shown in table 3.2.

3.3.2 HTML and reactivity

I used Blaze as the reactive rendering library. MondoLudo is a single page app, and all templates are defined in just one HTML file. Templating with Blaze takes some time getting used to, but is rather simple and straightforward. The HTML code is not extensive, as all functionality is made in separate JavaScript-files.

3.3.3 Styling with SCSS

By using the package "fourseven:scss" it is possible to use SCSS ("Sassy CSS") for styling instead of plain CSS. The SCSS file is compiled to CSS and minified.

SCSS offers the possibility of nested rules, mixins and the use of variables in the stylesheet. I decided to not include too many flashy visual elements, the styling itself is clean and easy to change, and also works on mobile devices.

3.3.4 JavaScript code

All templates from the HTML can have event handlers and functions defined in "client.js". This is the bridge between frontend and backend. The functions that need database access gets this through the file "methods.js".

Package name	Version	Description
accounts-facebook	1.0.10	Login service for Facebook accounts
accounts-ui	1.1.9	Simple templates to add login widgets to an app
blaze-html-templates	1.0.4	Compile HTML templates into reactive UI with Meteor Blaze
ecmascript	0.4.7	Compiler plugin that supports ES2015+ in all .js files
es5-shim	4.5.13	Shims and polyfills to improve ECMAScript 5 support
facebook	1.2.8	Facebook OAuth flow
fourseven:scss	3.8.0_1	Style with attitude. Sass and SCSS support for Meteor.js.
insecure	1.0.7	(For prototyping only) Allow all database writes from the client
jquery	1.11.9	Manipulate the DOM using CSS selectors
meteor-base	1.0.4	Packages that every Meteor app needs
meteorhacks:npm	1.5.0	Use npm modules with your Meteor App
mobile-experience	1.0.4	Packages for a great mobile user experience
mongo	1.1.9_1	Adaptor for using MongoDB and Minimongo over DDP
npm-container	1.2.0+	Contains all your npm dependencies
random	1.0.10	Random number generator and utilities
reactive-var	1.0.10	Reactive variable
session	1.1.6	Session variable
standard-minifier-css	1.0.8	Standard css minifier used with Meteor apps by default.
standard-minifier-js	1.0.8	Standard javascript minifiers used with Meteor apps by default.
tracker	1.0.14	Dependency tracker to allow reactive callbacks
utilities:avatar	0.9.2	Consolidated user avatar template (twitter, facebook, gravatar, etc.)

Table 3.2: Meteor packages used in MondoLudo

By default, all collections are available for all clients, meaning that users will have access to all content in the database. This can be turned off by removing the “autopublish” package, but is handy in the start phase of development. With no autopublishing you need to explicitly publish all relevant collections in “server.js” and subscribe to them in “client.js”.

Also by default the client gets access to writing to the database, with calling insert, update and remove functions. The removal of the “insecure” package fixes this issue. Once again, it is useful in the start of a project, but is a security risk that should be taken care of. All database operations should be wrapped in “methods.js” for added security.

To call methods from the client, you use “Meteor.call()” or “Meteor.apply()”.

3.3.5 Collections

I have defined the following collections in the Mongo DB:

```
Games           = new Mongo.Collection('games');
GameUsers       = new Mongo.Collection('gameusers');
Questions       = new Mongo.Collection('questions');
AnswerQuestions = new Mongo.Collection('answerquestions');
UserQuestions   = new Mongo.Collection('userquestions');
Friends         = new Mongo.Collection('friends');
```

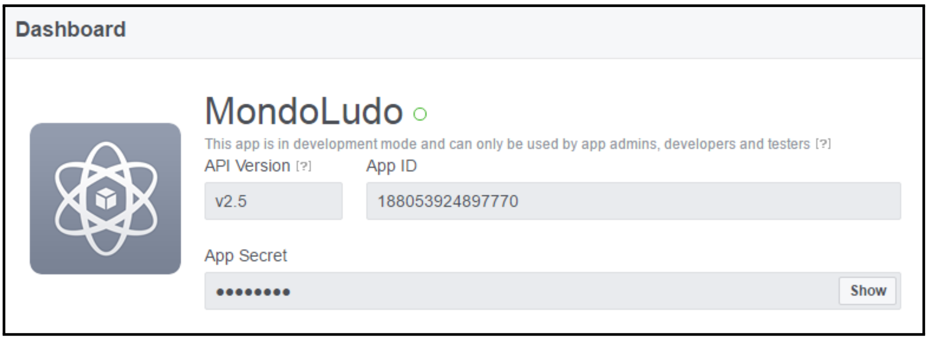
Table 3.3: Collections used in MondoLudo

3.3.6 Facebook integration

Meteor has is own user management options, but I decided to use an external OAuth service. The main reason for this is to avoid making functionality from scratch which does not bring any advantages to the table. External OAuth providers include services like Facebook, Google and Twitter.

The fact that Facebook is so wide spread, and in addition makes available your friend list for corresponding apps, is suitable for this project and it is my only supported way of authentication at the moment.

To use Facebook's OAuth service, you must first make a Facebook app at https://developers.facebook.com.



Dashboard

MondoLudo ○

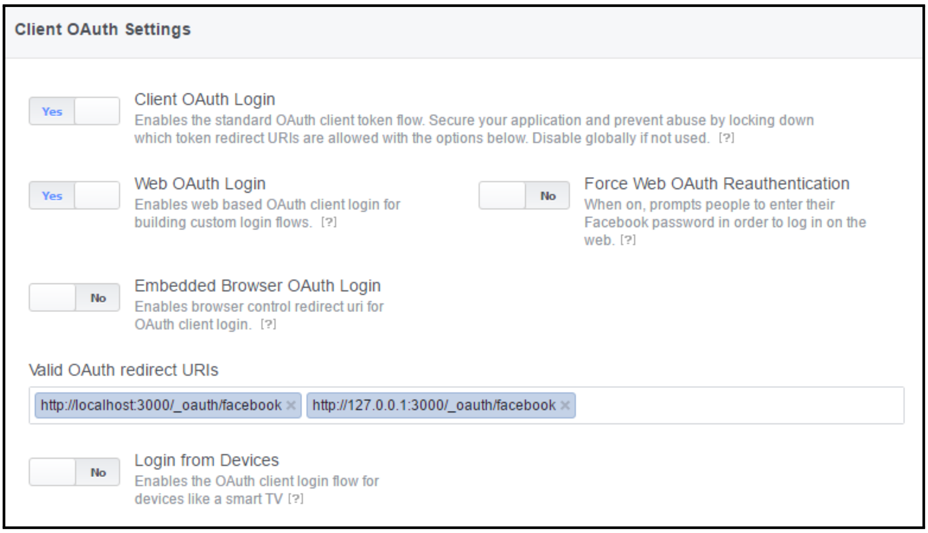
This app is in development mode and can only be used by app admins, developers and testers [?]

API Version [?]: v2.5

App ID: 188053924897770

App Secret: •••••••• Show

Figure 3.1: Creating a Facebook app



Client OAuth Settings

Yes — Client OAuth Login
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

Yes — Web OAuth Login
Enables web based OAuth client login for building custom login flows. [?]

No — Force Web OAuth Reauthentication
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

No — Embedded Browser OAuth Login
Enables browser control redirect uri for OAuth client login. [?]

Valid OAuth redirect URIs
http://localhost:3000/_oauth/facebook × http://127.0.0.1:3000/_oauth/facebook ×

No — Login from Devices
Enables the OAuth client login flow for devices like a smart TV [?]

Figure 3.2: Configuring Facebook OAuth settings

You have to enable Client OAuth Login and Web OAuth Login, and enter a redirect URI to tell Facebook where to return after login.

To use Facebook OAuth in Meteor you need to do the following:

- Install the Meteor packages “accounts-ui” and “accounts-facebook”

- add the template `{{> loginButtons}}` in your main HTML file
- Run once in browser
- Type in your Facebook app ID and app secret when prompted

This works fine in theory and numerous public examples, including several of my own test projects.

Unfortunately, I nevertheless ran into some serious problems getting this to work in MondoLudo. First of all, I have not succeeded in making this login method work in Internet Explorer. Authentication works in Chrome, Firefox and Opera, but IE hangs when redirecting. Secondly, with all necessary plugins installed, the login hangs in the other browsers with a loading symbol displayed where the user avatar, username and sign out button should be.

At first I thought that it could be the Facebook OAuth redirect URI that was wrong, so I tried several different online suggestions to no avail. I also thought that there might be something in my code causing the problem, but when logging in without any of the other external Meteor packages installed in the first place, login works as it should.

Uninstalling all plugins does not solve the problem, which means that there probably is some sort of configuration that is not reverted when unistalling. Login also worked in a previous version using the same packages, so I think that there might have been a recent package update that might have caused the problem.

The current temporary workaround is as follows:

- Install a new Meteor project from scratch
- Copy all project source files to the clean install
- Install the packages “accounts-ui” and “accounts-facebook”
- Log in
- Install all other packages

This is obviously a huge setback, meaning that as of now MondoLudo can not be used by the public. I have narrowed down which package is the culprit, and it turned out to be actually two of them. The first one is the “utilities:avatar” package, which fortunately is not necessary to include as it only displays the avatar image for the logged in user.

The other one is the “session” package, and this is critical for MondoLudo to work. From the release log I found out that it was recently updated. I’ve tried installing an older version, but of some reason I get an error message, and have not succeeded in fixing the problem. I’ve searched in forums, but I haven’t found any useful suggestions.

This might be an indication of Meteor still being a young platform which is not as robust as one should expect and desire.

3.3.7 Deployment

Development and testing was run locally on my computer. When going public, you have to deploy to an external server.

There are several providers of Meteor hosting. Until spring of 2016, Meteor offered its own free deployment option where you could deploy to their servers simply by running the command `meteor deploy <subdomain>`. You would then have your app running at subdomain.meteor.com, which I’ve used for testing until the service was stopped. This service was increasingly popular and presumably costly, so Meteor replaced this service with their Galaxy hosting environment.

By deploying to Galaxy you can map your DNS name to their server, for instance www.mondolodu.com. However, rather surprisingly you need a different hosting provider for the Mongo database. There are several cloud based solutions for this, you can use AWS, Azure, Google or your own server. Galaxy and Mongo hosting is not expensive, but I haven’t set up an external app since I’m currently having problems with user authentication and have to figure that out first.

Three runtime environments should be used. One for development on your own computer, secondly a staging server as intermediate used for testing, and finally a production server where you run the publicly available app.

3.3.8 Development issues

Several issues and challenges in addition to the bedore mentioned Facebook problem arose during development.

IDs

All Meteor objects are assigned their own ID, including users. My solution for Facebook integration is to use Facebook’s own user ID for managing players when playing a game, and this led to a few problems.

When a user responds to a challenge, she uses the Meteor.userID() as owner to store the data about the game in the database. The gamemaster does not have access to all user IDs, he only has access to his friends' Facebook IDs. This in turn means that the gamemaster does not have access to the players' answers.

I solved this by automatically adding the Facebook ID as soon as the user responds to a challenge. From then on the gamemaster will have access to the status of the player's game progression. There are other ways of achieving the same thing, but this turned out to be an easy way out of the problem.

Return values from Meteor.call() and Meteor.apply()

The asynchronous functions Meteor.call() and Meteor.apply() caused a few problems. The code flow in the client does not stop and wait for returned values. Sometimes one can assign the result to a variable, which works fine most of the time, but some of the methods return `undefined` instead of the expected values.

Calling Meteor.apply() with the argument `{returnStubValue: true}` which should give a result value does not necessarily work, unfortunately. This will be even more of a problem when deploying to a production server because of latency issues. I've tried several online suggestions, including using the recommended callback functions, but there are still some calls that don't work in this manner.

The easy way to fix this is by publishing the relevant collections and then do queries with find() and findOne() from the client without going through the corresponding methods in "methods.js". Methods should mostly be used where no return value is necessary, like for database writing, or for operations where you would set a session variable with the result.

Making iOS and Android apps

I had originally planned to make mobile phone apps in addition to the web based version, but I've postponed this because of the Facebook issues. I have on the other hand read a fair bit of information about Apache Cordova, and I think it shouldn't be too difficult to implement phone apps.

Chapter 4

Using the prototype

In this first version of the prototype I have implemented the simplest type of challenges and responses: Questions and answers. This serves to show how a game is set up and played. In later versions there will be the possibility of choosing between several different types of challenges and responses (see table 2.1 on page 18).

As a father of three, I decided to make a demo game for doing household chores. Any parent knows how difficult it can be to outsource tedious chores to their children. In my family we use a sticker achievement system placed on the fridge door as motivation. When the children have collected a given number they get a reward. This can easily be made in MondoLudo instead.

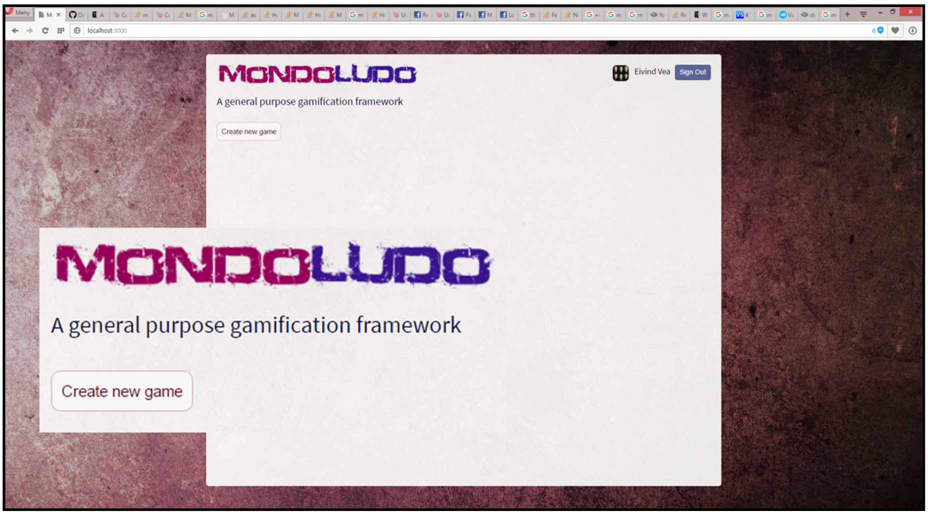
4.1 Creating a game

The default start screen consists of little more than a Facebook login button and a button for creating a game. In a full version it should be possible to choose from existing public games here.

The visual look and feel should be possible to change, based on a number of available templates. Ideally it should be possible to customize everything, like adding your own logo and changing colour schemes.

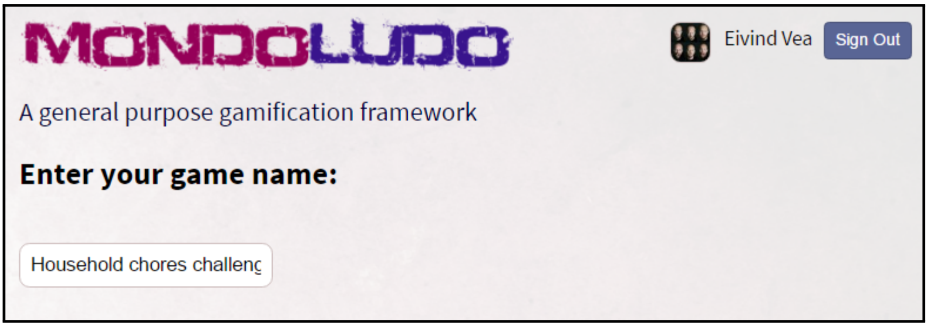
If you already are a gamemaster and/or a player, you will see your games here. In this case, we have not been logged in before, so there are no games here yet.

A new game is added by clicking the “create game” button and typing in a



A general purpose gamification framework

Figure 4.1: The start screen of MondoLudo



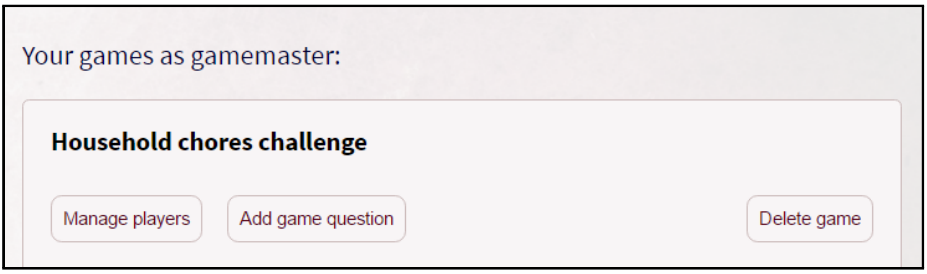
A general purpose gamification framework

Enter your game name:

Household chores challenc

Figure 4.2: Adding a new game

game name. This in turn gives you three options: Manage players, add questions or delete game.

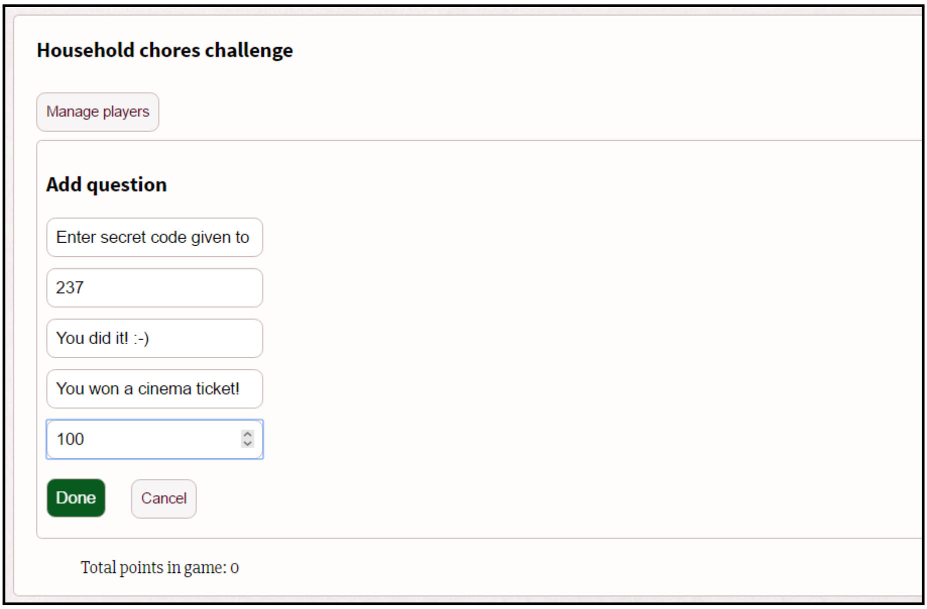


Your games as gamemaster:

Household chores challenge

Figure 4.3: Main game creation window

4.2 Adding challenges, responses and managing players



Household chores challenge

Add question

Enter secret code given to

237

You did it! :-)

You won a cinema ticket!

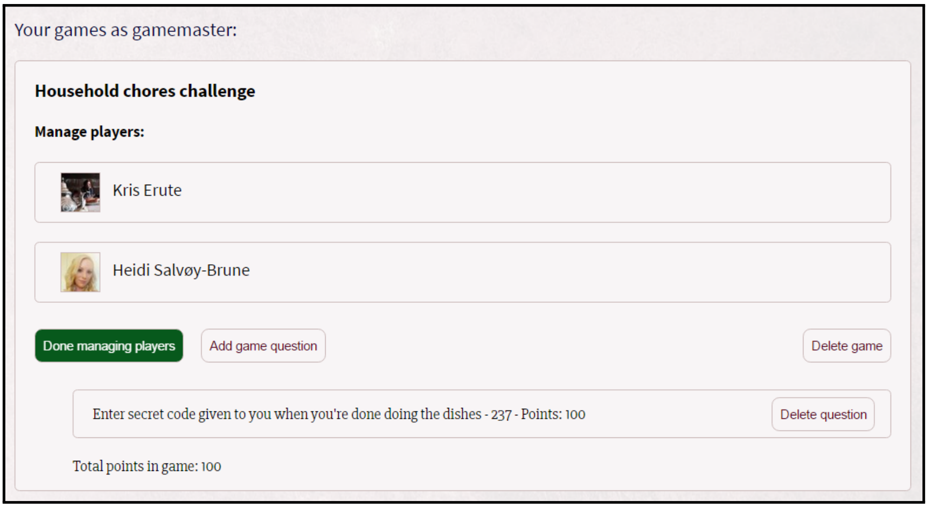
100

Total points in game: 0

Figure 4.4: Adding a question to a game

The gamemaster can enter game questions and answers, enter congratulation message and rewards, as well as the question's points. In a future version it should be possible to include pictures, videos and soundclips both as challenges, congratulation messages and rewards.

In addition it should be possible to have some sort of Google Map functionality, like adding geographical coordinates used as challenges, and automatically completing a level when the user physically is at the requested location. This is somewhat like geocaching and Pokémon Go, and could be used for setting up a scavenger hunt. Another use case could be to track hiking, where users have to visit physical locations, like in the Norwegian "TellTur"[22] application.



Your games as gamemaster:

Household chores challenge

Manage players:

Kris Erute

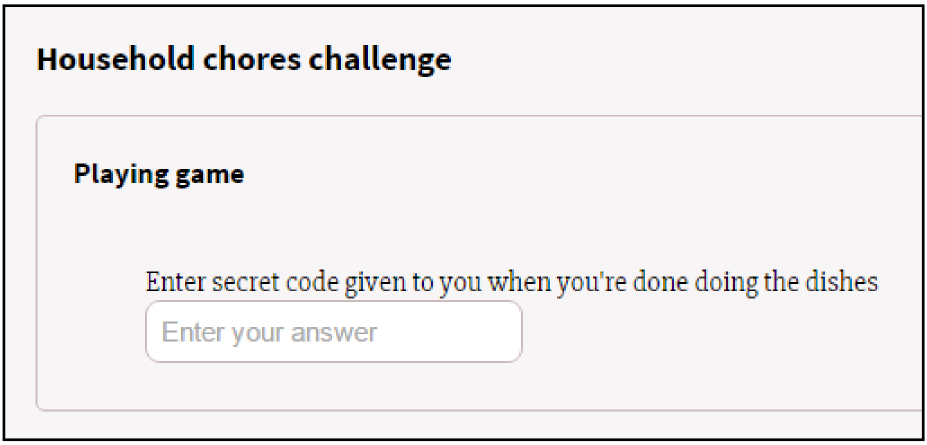
Heidi Salvøy-Brune

Enter secret code given to you when you're done doing the dishes - 237 - Points: 100

Total points in game: 100

Figure 4.5: Managing players

The player list is based on which of your friends that have already signed up and logged in at least once in MondoLudo. An invitation functionality should be added at a later stage. In addition it should be possible to see player teams here.



Household chores challenge

Playing game

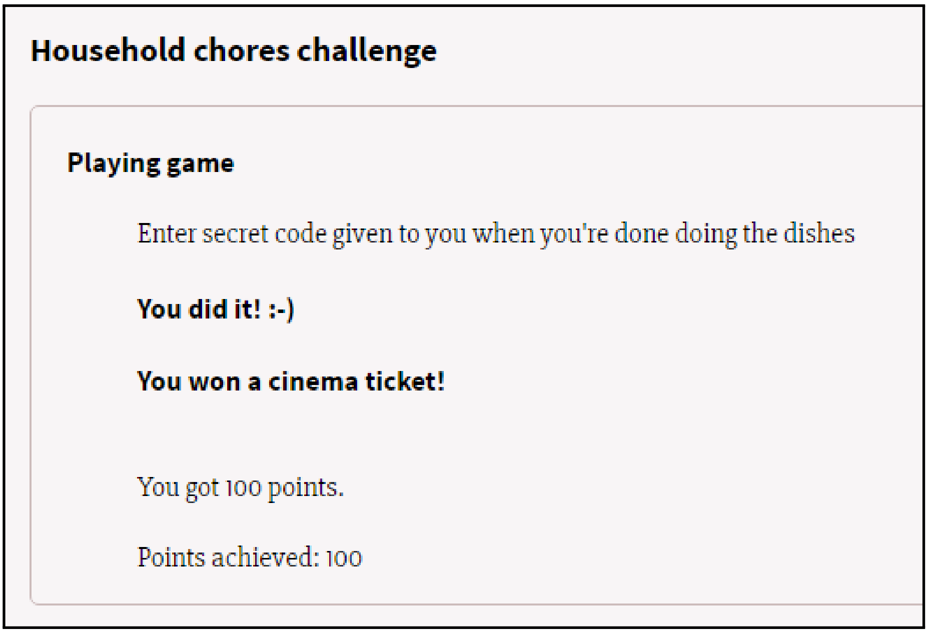
Enter secret code given to you when you're done doing the dishes

Figure 4.6: Playing the game

4.3 Playing a game

A new question added by the gamemaster in an ongoing game will show up instantly on the players' clients. The same applies the other way around; when a player completes a level, the gamemaster will see this immediately on his client as well. This is one of the key features in Meteor, instantly reactive updating of data.

Ideally players should be automatically sorted and placed on a real time leaderboard when playing.



Household chores challenge

Playing game

Enter secret code given to you when you're done doing the dishes

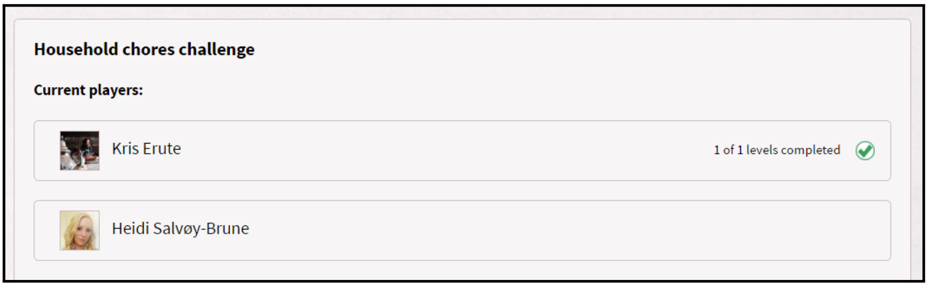
You did it! :-)

You won a cinema ticket!

You got 100 points.

Points achieved: 100

Figure 4.7: Completing a question and getting a reward



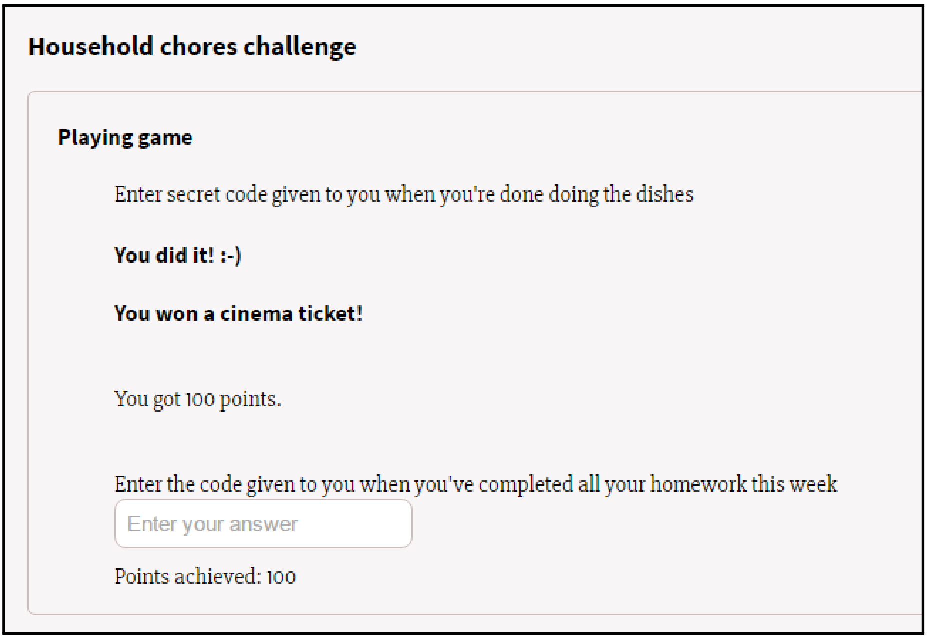
Household chores challenge

Current players:

Kris Erute 1 of 1 levels completed ✓

Heidi Salvøy-Brune

Figure 4.8: Gamemaster's view of current players



Household chores challenge

Playing game

Enter secret code given to you when you're done doing the dishes

You did it! :-)

You won a cinema ticket!

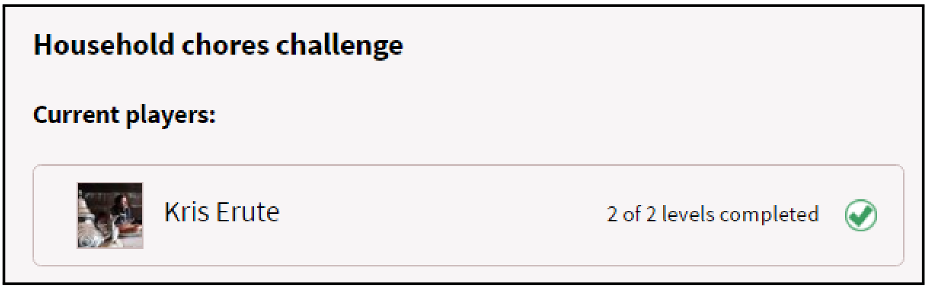
You got 100 points.

Enter the code given to you when you've completed all your homework this week

Enter your answer

Points achieved: 100

Figure 4.9: Adding a new question in ongoing game



Household chores challenge

Current players:

Kris Erute 2 of 2 levels completed ✓

Figure 4.10: Gamemaster's view of a completed game

Chapter 5

Conclusion and further work

5.1 Conclusion

I have implemented a web-based gamification framework in the JavaScript framework Meteor. The first version of the framework is simple, but shows that it is possible to use Meteor for these purposes. I've had a few challenges while developing, and it is possible that Meteor is not a mature framework quite yet. On the other hand, this is likely to change, as it is continuously developed further, and has a strong user community.

5.2 Further work

5.2.1 Technology

As for now, Blaze is sufficient as the reactive client-server layer, but I think that future versions should employ React or Angular instead. The main motivation for this is to be an early adopter of the frameworks Meteor will focus on from here onwards. In addition, both React and Angular are also used on other platforms, so programming skills in these areas are useful in a real world work situation for projects not based on Meteor.

5.2.2 Coding issues

Facebook integration issues must be resolved. I have to continue searching for solutions to this problem. Stack Overflow has so far been unsuccessful, but I will post a question there in hope of ideas about what might be the cause of the problem.

Code wise I have wrapped all database write operations in methods. This should be further investigated to make sure that users don't get unauthorized access to the database.

To avoid getting undefined return values from methods, the client functions should query directly when possible.

5.2.3 Gameplay features

MondoLudo-games could be run in three modes:

- Personal - the user is both gamemaster and player. This could be used for todo-lists or personal encouragement.
- Private - game with a gamemaster that invites friends.
- Public - anyone can play.
  - Open - anyone can join a particular ongoing game.
  - Closed - anyone can start a game and become gamemaster of games defined and made public by other users.

Media

Functionality for the upload and usage of media like images, soundclips and videos should be implemented. This can be used both in the visual look of the application, as well as being used as custom challenges and rewards. Such media should probably be hosted on an external provider other than Meteor's own architecture, as this has limitations when it comes to storage capacity,

The user experience will improve significantly if users can tailor their own look and feel of their games, so a good start would be to make more templates. The use of SCSS means that this should be fairly easy.

Geographical location

There should be implemented the possibilty of using geographical locations as challenges and responses. In addition, it would be interesting to experiment with augmented reality.

Chat client

A simple chat client should be added. This would allow communication between gamemaster, players and teams while playing a game.

5.3 Commercialization

I would like to continue the development of MondoLudo after my thesis is submitted. Since there are, as far as I've found out, currently no similar products, it would be interesting to see if there is a possibility of commercialization. The domains mondoludo.com and mondoludo.no have been acquired for later use.

MondoLudo should be made available in three versions, both on web and mobile:

- Free basic version with ads.
- Premium version without ads.
- Custom version for customers that would like to have implemented their own design or have requirements not offered in the standard versions.

Bibliography

[1] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0816-8. doi: 10.1145/2181037.2181040. URL `http://doi.acm.org/10.1145/2181037.2181040`.

[2] Steffen Walz. *The Gameful World : Approaches, Issues, Applications.* The MIT Press, Cambridge, Massachusetts, 2014. ISBN 978-0-262-02800-4.

[3] Odd Frank Vaage. Norsk mediebarometer 2015. Statistisk sentralbyrå (SSB) / Statistics Norway, 2015. URL `https://www.ssb.no/kultur-og-fritid/artikler-og-publikasjoner/_attachment/262805`.

[4] Entertainment Software Association. Essential facts about the computer and video game industry, 2015. URL `http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf`.

[5] Kahoot! URL `http://www.kahoot.it`.

[6] Christopher Cabrera. *Game the Plan : every sales rep's dream, every CFO's nightmare.* River Grove Books, Austin, TX, 2014. ISBN 978-1938416545.

[7] Fabulous. URL `http://www.thefabulous.co`.

[8] Yu kai Chou. Octalysis. URL `http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/`.

[9] Andrzej Marczewski. Gamification Framework. URL `http://www.gamified.uk/gamification-framework/`.

[10] Zachary Fitz-Walter. *Achievement Unlocked: Investigating the Design of Effective Gamification Experiences for Mobile Applications and Devices.* PhD thesis, Queensland University of Technology, 2015.

[11] Funifier. URL `http://www.funifier.com/`.

[12] Klint Finley. TechCrunch, 2012. URL `https://techcrunch.com/2012/07/25/andreessen-horowitz-keeps-eating-the-software-world-with-11-2-million-investment-in-javascript-framework-company-meteor/`.

[13] Meteor Development Group, . URL `https://guide.meteor.com/ui-ux.html#view-layers`.

[14] Discover Meteor. URL `https://www.discovermeteor.com/blog/blaze-react-meteor/`.

[15] Meteor Forums. URL `http://forums.meteor.com/t/why-blaze-2-if-we-go-for-react`.

[16] Isaac Strack. *Getting started with Meteor.js JavaScript framework : learn to develop powerful web applications in minutes with Meteor.* Packt Publishing, Birmingham, UK, 2015. ISBN 978-1-78528-554-7.

[17] Stephan Hochhaus. *Meteor in action.* Manning Publications, Shelter Island, NY, 2016. ISBN 978-1-617292-47-7.

[18] Richard Silverton (richsilv). URL `https://github.com/richsilv/meteor-cordova-notifications`.

[19] Morten N.O. Nørgaard Henriksen (raix). URL `https://atmospherejs.com/raix/push`.

[20] Douglas Crockford. *JavaScript : The Good Parts.* O'Reilly, Beijing Cambridge, 2008. ISBN 978-0-596-51774-8.

[21] Meteor Development Group, . URL `https://guide.meteor.com/using-npm-packages.html#using-npm`.

[22] TellTur (Friluftsrådenes Landsforbund). URL `http://telltur.no`.

List of Figures

List of Tables