



**Universiteit  
Leiden**  
The Netherlands

# Informatica & Economie

Quantum Rules!

CJH Meijerink

Supervisors:

Dr. H.P. Buisman

Dr. A.W. Laarman

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

[www.liacs.leidenuniv.nl](http://www.liacs.leidenuniv.nl)

04/06/2018

## Abstract

Quantum Rules! is a platform that enables high school students to gain a better understanding of the complex quantum world. A teacher plans a visit with Quantum Rules!, as the school often does not have the required materials or equipment, enabling the students to perform these experiments.

During the visit, students are stimulated to work on the experiments which will elaborate quantum topics. To avoid that physics experiments become boring, a more motivating context for the experiments is required. In the current situation not all the students participating in the visit are motivated enough which is exactly what the developed application will prevent.

Firstly, the application will introduce gamification into the Quantum Rules! platform. Students working on the experiments have to be guided through all the steps. Traditionally this would have been achieved with a manual, the application will introduce levels. For each level the student will have to answer a question. Answering correctly will reward that students some points.

Secondly, elements of an escape room will be added to the application as to the laboratory where the experiments are conducted. The lab where the experiments are conducted will contain elements of an escape room. Multiple beamers will project relevant data and possible hints to solve questions and the lab session will only last for a certain amount of time meaning that there is a pressure to work within the time limit.

Finally, the application will track the progress of all players. The students will be working together as they will often need the help of fellow students to solve a problem but they will also compete others as the application tracks who solved what questions. The application shows the progress of the experiments realtime, stimulating the students to keep working as seeing experiments progress implies that other players are scoring points.

By using this application the students are offered such an immersive experience between the digital experiments in combination with the actual experiments in the lab resulting in an unparalleled stimulant to keep working on the experiments. This combination of science and fun is the goal of the application.

To develop this application it was decided with the mandator to work in an agile manner as requirements were expected to change during the development. Maintainability of the application is one of the key requirements and therefore the goal of this thesis is to support the documentation of the application by providing an extensive context. All the requirements for this project will be explained together with the design of the application. Also, the definitive implementation of the used frameworks and testing this implementation is mentioned. Using this approach, a solid foundation is provided for any further work.

## Acknowledgements

A very special thanks to Henk Buisman for the extensive cooperation. Together with Buisman all the requirements were elicited and the scope of the project was defined. Over several months progress seemed little as development was all done in the backend but we never lost vision of the desired result. I believe we can be very happy with the current version of the application!

I am grateful to Alfons Laarman for guiding me through the thesis. This thesis would not have been the same level without your help, many thanks for that!

With a special mention to Joost van Someren and Emiel Beinema for aiding me in the development. These two friends were the only people capable to help me as they had the required Vue.js and Loopback.io experience. Their help was crucial as I, a PHP developer, had no experience with javascript frameworks like the ones used in this application.

To Max Snijders, my coach when I started developing applications. Without his help I am sure that I would not be a developer today and not have the required skills to even start with this thesis.

Last but by no means least, thanks to my loving girlfriend and my family for supporting me in every way possible.

I thank you all!

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The project . . . . .	2
1.2	Thesis Overview . . . . .	3
<b>2</b>	<b>Background</b>	<b>4</b>
<b>3</b>	<b>Requirements</b>	<b>7</b>
3.1	Functional requirements . . . . .	7
3.1.1	Admin . . . . .	7
3.1.2	Player . . . . .	8
3.2	Non functional requirements . . . . .	8
<b>4</b>	<b>Design</b>	<b>9</b>
<b>5</b>	<b>Choosing technologies</b>	<b>15</b>
5.1	Client side: Vue.js . . . . .	15
5.1.1	Vue-router . . . . .	18
5.1.2	Vuex . . . . .	18
5.2	API: Loopback.io . . . . .	19
5.3	Socket.io . . . . .	21
5.3.1	Integration with Vuex . . . . .	21
5.4	Security . . . . .	22
<b>6</b>	<b>Quality assurance</b>	<b>23</b>
6.1	A priori . . . . .	23
6.2	A posteriori . . . . .	24
<b>7</b>	<b>Conclusions</b>	<b>25</b>
	<b>Bibliography</b>	<b>27</b>

# Chapter 1

## Introduction

Since 2016 quantum mechanics is an official part of the national physics curriculum for Dutch highschools. High schools usually have insufficient means to support practical assignments for this subject. Quantum Rules! is a platform founded by H.P. Buisman where high school teachers can book a session for their students at the university where experiments concerning quantum are being conducted by the students. Buisman is a physics teacher and liason officer for the Leiden University physics department where he pairs high schools with physics projects. Meaning that students often come to the university to conduct certain physics experiments (for example the Quantum Rules! platform) to learn more in depth about physics, but also to help students with their high school graduation, physics, project that their school cannot help with due to logistical issues concerning materials or equipment.

The aim of this project is to use gamification to increase the appeal of physics experiments for high school students. Introducing a digital platform where the students can work together and compete with each other in combination with elements from an escape room, it is guaranteed that the students will be more enthusiastic about conducting experiments.



Figure 1.1: Students working on an experiment.

## 1.1 The project

Within the project Quantum Rules! students work together on different experiments to explore different aspects of the quantum world. These experiments are conducted in a lab within the university.

An example experiment that will be conducted in the quantum context is the photoelectric effect experiment, a copy of the nobel prize winning experiment by Einstein. This experiment will be used throughout the thesis as sample experiment to refer to in order to clarify other requirements and implementations of the application. The goal of this experiment is to understand how the photoelectric effects works, which allows the derivation of the Planck constant upon completion of the experiment. Example questions that have to be answered in this experiment are:

- Place the lightbulb (Q1)
- Plot an IV graph (Q2)
- What is the stopping voltage? (Q3)
- What is the wavelength of the light? (Q4)
- What is the Plack constant? (Q5)
- Upload a photo of the setup (Q6)

The goal of answering questions like these is that the students conducting the experiment are guided through all the steps of the experiment to eventually, for this example, determine the Planck constant.

To increase the appeal of their assignment, elements of an escape room will be added. In an escape room, contestants are stimulated to work together in order to solve a set of problems within a limited amount of time. The escape room motivates the students even more to participate during the lab session as the application rewards participation. Seeing players progress in the game will only stimulate others to work harder as nobody wants to end as last and therefore the players will learn from each other, using the knowledge of the whole class to solve all the experiments. In the laboratory of the Quantum Rules! platform a lot of styling will be added to contribute to this feeling. Beamers will project relevant videos, props will be added and the students will be exposed to a time limit.

As a motivating context, challenges that society faces today have been chosen (e.g. energy transition and healthcare) for the students to solve. These challenges can only be solved with the aid of the modern technology modeled by physics experiments. At the end of the day, students share their experimental results with the rest of the class. All groups visiting the laboratory have some basic knowledge about the experiments that they will have to perform. Their visit will always be aligned with their school curriculum as the teacher will only book a session if it is relevant for the students. Some teachers will give their students more preparation than others but the level of the experiments at university will adjusted to the capabilities of the students in such manner that all players are capable of solving them.

To include the experiments in a game, this project aims at developing an interactive platform for the students to play in. This platform provides instructions for the experiments and enables the students to share their data, see progress of other students concerning the various experiments and most important of all, see how modern physics experiments are relevant for solving the issues that society faces today.

During this project certain design choices will be made, all to ensure the best result: a platform where students are motivated to keep participating in high school research and above all where they learn to work together in a scientific environment.

## **1.2 Thesis Overview**

This thesis is written for H.P. Buisman (the mandator) as the application delivered is an extension of his platform: Quantum Rules! The users are students that participate in physics experiments. With the introduction of this application, the students are offered an unparalleled experience in the context of an escape room. The setting in the laboratory, the possibility to help each other but also to outsmart other participants and above all the fact that participants get a better feeling for the social impact of physics experiment are all factors that make this a unique project. Using this digital application will stimulate experimental research on high school level.

In this chapter the context of this thesis has been briefly introduced. Chapter 2 will add context to the application by introducing all necessary terms and elaborate on what the application should be able to do. Chapter 3 shows all the requirements of the project as eventually negotiated with the mandator. Now that the context of the application has been explored and the requirements of the project have been mentioned, the architecture will be explained in Chapter 4. After providing insight in the platform that has been developed some deeper knowledge about the client side and the API of this application will be provided in Chapter 5. In Chapter 6, testing the application will be discussed together with how the quality can be assured in the future. The final Chapter 7 concludes.

# Chapter 2

## Background

The experiments are currently conducted in a lab. The mandator envisions an escape room like setting for the lab, complemented with a digital environment. This digital environment will be a web based interface with API having two different types of users.

1. Admin
2. Player

Within the admin panel of the website all settings can be managed as explained below. The players will be the students participating in the lab. They will use their laptop, mobile phone or computers in the lab to join the digital environment where they can work digitally on the experiments they are conducting in the lab.

Details about the design can be found in Chapter 5.

Keyword (entitiy)	Definition
Admin	The person capable of defining all entities, relations and instances.
Player	The person playing the game as defined by the Admin.
Project	A set of experiments adjusted for the targeted group. A project is added to the room.
Experiment	Digital representation of a physical experiment. Experiments are a child property of Projects.
Level	Steps to guide the player through the experiment in the form of questions. Levels are a child property of Experiments.
Case	High level motivating context that define the game. Cases are a child property of Projects and related to levels. Cases are gradually solved through completing levels.
Room	Instance of the project for a specific visit.
Game	Players' view of the room.

Table 2.1: Keywords summary.



## **Admin**

The admin can define the game that has to be played. Within the scope of this thesis, the admin currently is one account only accessible by the mandator. The design, as mentioned in Chapter 4, is flexible enough to support multiple admins with each their own projects in the future.

## **Projects**

A project contains a set of experiments. Whenever the game has to be played, the admin can create a room which contains a project for the players to solve. The sole purpose of the project is to be the link between experiments belonging together and the admin. For this thesis a digital project will be created that contains a set of experiments that are related to learning about quantum. For each project a different set of experiments, adjusted to the capabilities of the students that have to conduct these, can be created.

## **Experiments**

In the lab, the players have to solve experiments. For each experiment that can be conducted in the laboratory, the admin can create an experiment entity within the digital environment.

In the context of this thesis, Quantum Rules!, all experiments will be related to experiments concerning quantum physics. Whenever another theme is introduced, the experiments in the laboratory will change and so will the experiments in the digital environment. An example experiment mentioned in Section 1.1.

## **Levels**

Students that work on an experiment have to be guided through their experiment in a specific order with the help of questions. Each step in the experiment is called a level which contains a question for the players to answer.

## **Cases**

The project definition in Chapter 1 mentions the motivating context that is important for the game that will be played: challenges that society face today. These challenges are called cases.

## **Rooms**

A room is the specific instance of the project for the specific visit. Meaning that for every class that comes to university to conduct the experiments. When a school comes to visit the university a room, named eg. "Room - <schoolname> - <date>", will be created. In this newly created room, the players can solve the related project.

## **Players**

Players, the students that participate in the experiments, can join the room created by the admin where they will see all experiments that belong to the related project. When conducting a certain experiment in the laboratory, they can select that same experiment in the digital environment for them to be guided through all the steps. Answering one level at a time the players are all collaborating to solve all the cases. Different players work on different experiments with each their own levels. The players that solves a level, is rewarded. This stimulates the players to collaborate and also enhances the competition as everybody wants to be the winner.

## **Game**

A game is the players' view of a room. Functionally spoken, a game is the same as the room but only for the player and not for the admin. Therefore a game will load the related project and the players can answer all the questions belonging to the experiments. Introducing this keyword makes it easier to reference to as a game is more commonly understood than the functional room keyword.

# Chapter 3

## Requirements

Before making any decisions concerning the design, architecture or implementation of the application, the requirements have to be elicited. Together with the mandator, the following user stories were composed. These stories describe what the application should be able to do. The eventual architecture that implements all the requirements are mentioned in Chapter 4.

### 3.1 Functional requirements

#### 3.1.1 Admin

A1 Should be able to log into his/her account.

A2 Create a project containing different sets of experiments adjusted to the capabilities of a class.

A3 Create a room containing a project.

A4 Add levels to an experiment.

A5 Add cases to a project.

A6 Link cases to a level in order to add motivating context (e.g. applications of the results in society)

A7 Create rooms which the players can join.

A8 View the progress of a room.

A9 View results of the rooms.

A10 See the individual results of all players within a room.

### **3.1.2 Player**

P1 Should be able to join a room.

P2 Is not required to create an account before joining a game.

P3 Is guided through the experiments with the help of the defined levels.

P4 Must be able to play the game on computers, tablets and mobile phones.

P5 Be able to help other players in the same game.

P6 See how far the cases are solved.

P7 Progress of all the relevant cases for the room should be shown without refreshing the page.

P8 Compare with other players how many questions everybody answered and the amount of points scored.

P9 Must continue playing the game, without entering the code, when the page is reloaded.

## **3.2 Non functional requirements**

NF1 The application should be responsive for all types of devices. Meaning that a player on mobile, desktop, laptop and tablet should all see a similar interface.

NF2 Consistent styling should be used throughout the entire application.

NF3 All code must have a high level of maintainability.

NF4 Documentation should suffice for others to be able to continue with the development of this application.

NF5 Must feel like a game for the players.

# Chapter 4

## Design

The architecture of the database implementing all the definitions as mentioned in Chapter 2 and other necessary tables needed to support the logic of the application as mentioned above can be summarized in an Entity Relationship Diagram: Figure 4.1.

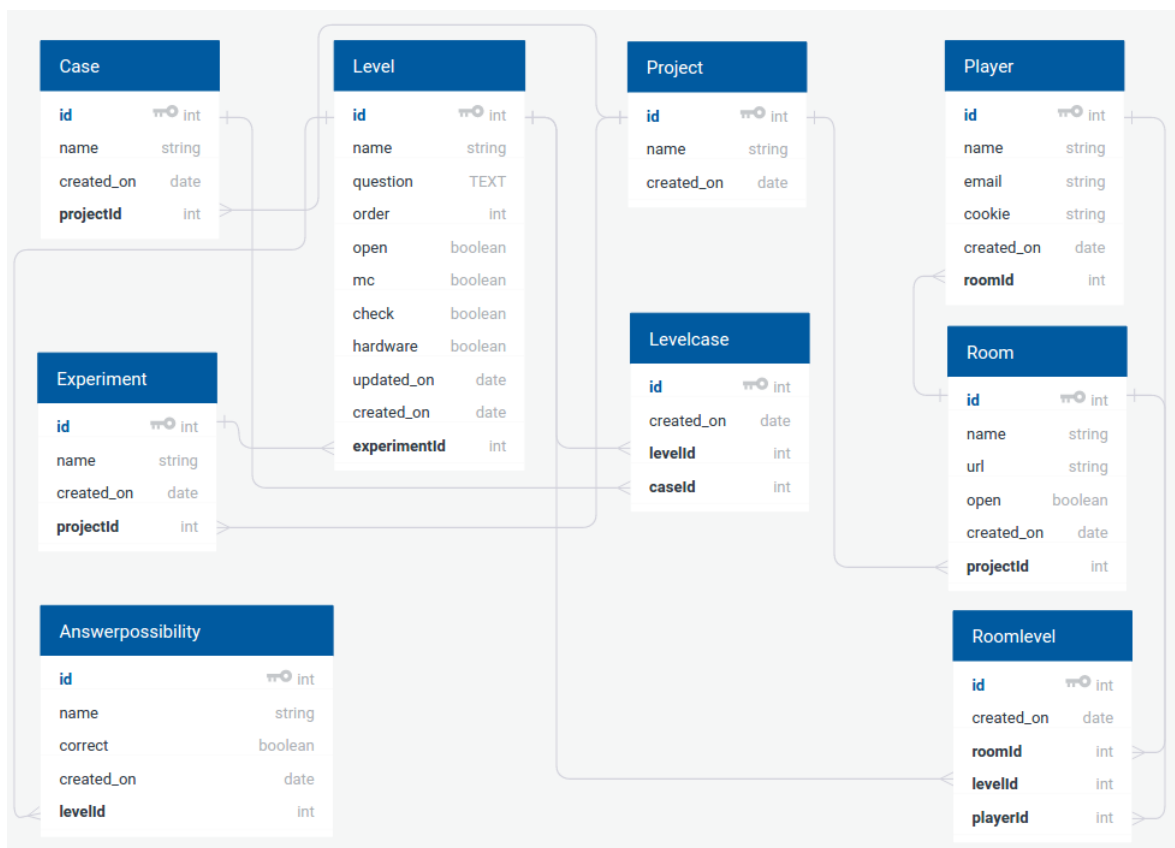


Figure 4.1: ERD of the database architecture.

In this figure, every table represents an entity or a table required to map certain relations. How this architecture implements all the requirements will be explained by elaborating all pages of the application.

## Login

The first visible page when visiting the website is the login page. Here a player can join the room by entering his name, email and the code that is unique for the room. This code is visible for the admin when the room has been created. By sharing this code with the students visiting the laboratory, everybody has the required information to join the room. When a user has once joined a room with a specific email and that email is used again for the same room by another player, the same account is loaded into the game.

Whenever the admin wishes to login to manage the data, a simple click on the Admin button will suffice. Clicking here will load the *Auth0* framework. More information about *Auth0* can be found in Section 5.4.

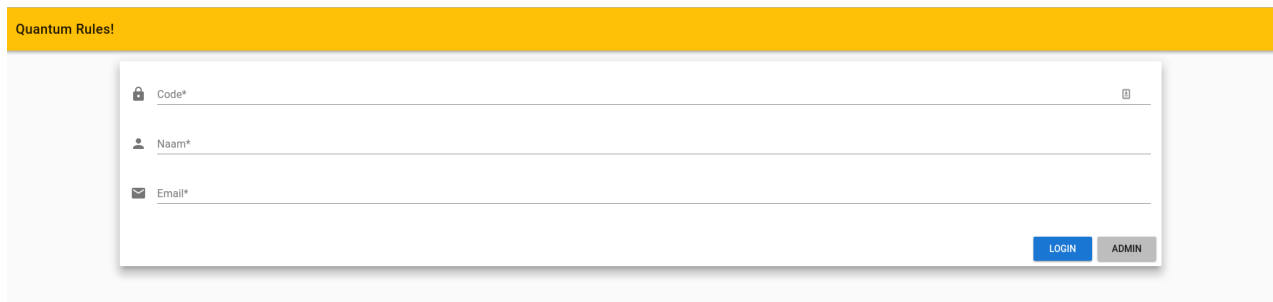


Figure 4.2: The login screen.

## Game

When a player joins the room this is the main view for the application. After the player has joined, a cookie is created to ensure that this page with all required data (the experiments and player data) will be loaded automatically.

In the sidebar the player can select an experiment to solve. At any time it is possible to select another experiment meaning that the player can help someone else or start working on a new, not yet started, experiment. After selecting the experiment, the question that has to be solved for that experiment will be loaded. Upon answering this correctly, all players viewing that experiment will automatically be presented with the new question and the *Roomlevel* table (as seen in Figure 4.1) will be updated with that level. Players not currently working on that experiment will see the cases progress if related levels are solved. When an experiment has been solved, the players will be notified about this. When all experiments are solved, it is not possible to select any experiment. The players will only be able to see a message congratulating them on completing all experiments successfully.

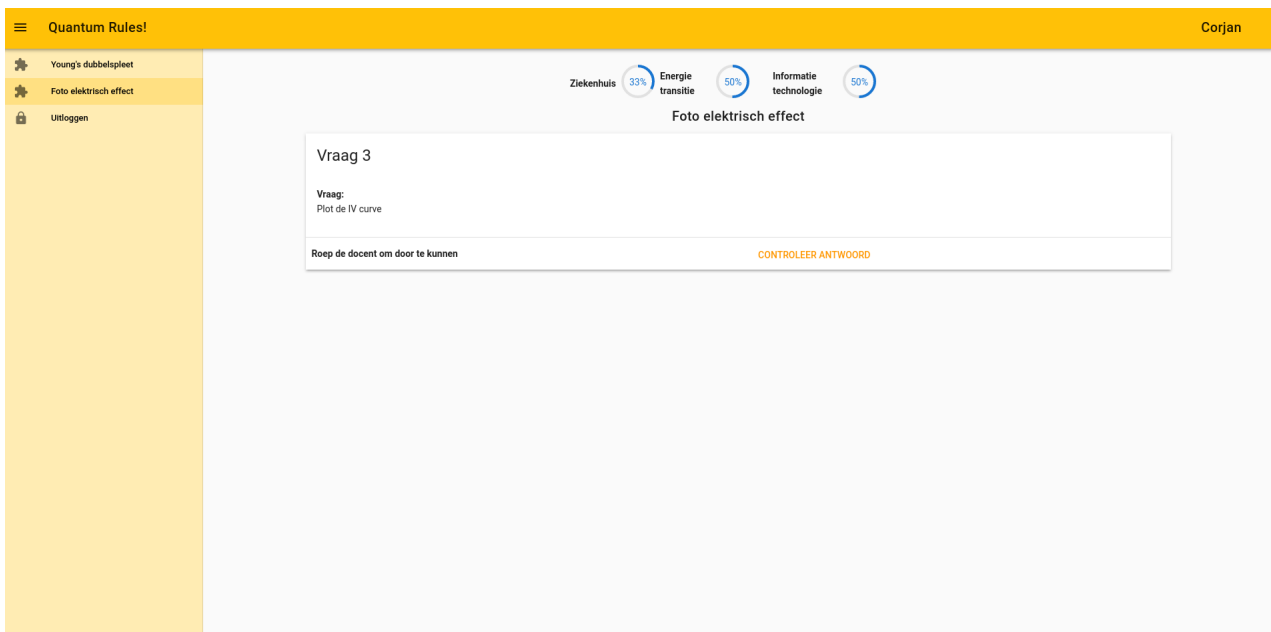


Figure 4.3: The game.

## Home

If the admin has logged into the application the home screen will list all the projects. In the sidebar the admin can toggle between this view and that of the rooms as will be explained later. In the list of projects it is possible to create a new project, rename an existing one or to delete it. When deleting a project, all related data will also be removed. A project is defined in Chapter 2 as a child property of Admins, however this relation is not mapped in the architecture as can be seen in Figure 4.1. In the final implementation it was more convenient to not map these relations as the Admin will only be one account in the context of this thesis. Adding this relation later on will not lead to any major consequences and therefore it was decided to leave this relation out of the final design.

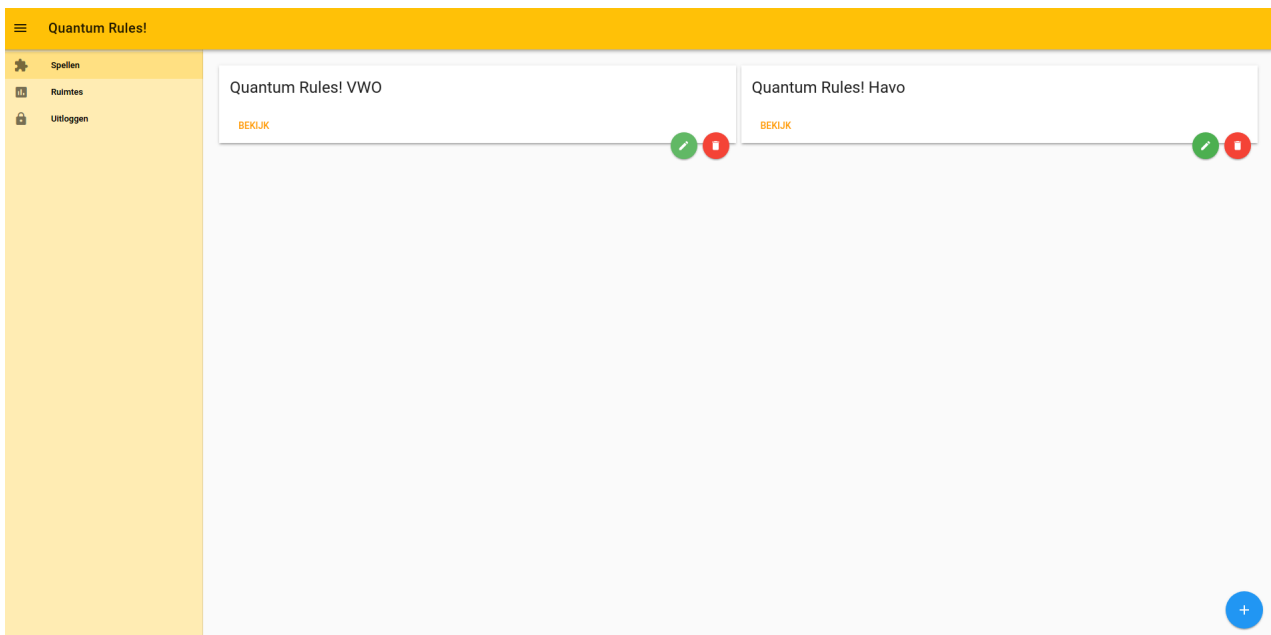


Figure 4.4: The home screen.

## Experiments

This page has exactly the same functionalities as the home screen with the only difference that experiments rather than projects are listed.

## Levels

Upon clicking an experiment, a new page that lists all the levels is opened. Here a quick overview is presented of all the levels. When a level is selected, more details about it are given. It will now also be possible to edit the level. Besides changing the name and question asked, it is possible to map certain relations or change the type of question. There are four different type of questions:

- Multiplechoice: select one of the given answer possibilities.  
Q<sub>4</sub> in Section 1.1 is a sample question that could be multiplechoice. The admin can add multiple answer possibilities (400nm, 500nm, 600nm, 700nm) which the player will all see and they can select the correct one. It is possible for multiple answers to be correct.
- Open answer: allow the player to enter any answer  
Q<sub>3</sub> and Q<sub>5</sub> in Section 1.1 would be an open question. The players can enter their value for the stopping voltage or their derivation of the Planck constant. Only if the entered answer matches one of the answer possibilities the player can continue to the next level.



- **Hardware connected:** to remind the player about a certain step in the experiment  
Q1 in Section 1.1 is an example of this. Whenever the player simply did a step in the experiment, they can acknowledge this and proceed to the next level.
- **Teacher needs to check:** whenever a step has to be verified by the teacher the player cannot continue until the teacher indicates the level as correctly answered.  
Q2 in Section 1.1 is an example that the teacher needs to check. Only if the obtained curve is correct, meaning the players understand what they are doing, they are allowed to continue. As this graph is created in an external program, this application cannot check if it is correct and therefore the teacher can approve the level by checking it.

Whenever the level is multiplechoice or an open answer, the admin can add answer possibilities to the level and indicate for each possibility if it is correct. For multiplechoice questions, each possibility is shown as option for the player to select. Besides managing the answer possibilities, none, one or multiple cases can be added to a certain level meaning that whenever the player solves the corresponding level, all relevant cases progress a little more to becoming solved. In the example provided in Section 1.1, relevant cases for the experiment would be the healthcare, energy transition and mobile phones. Without the photoelectric effect research in these fields would not be possible.

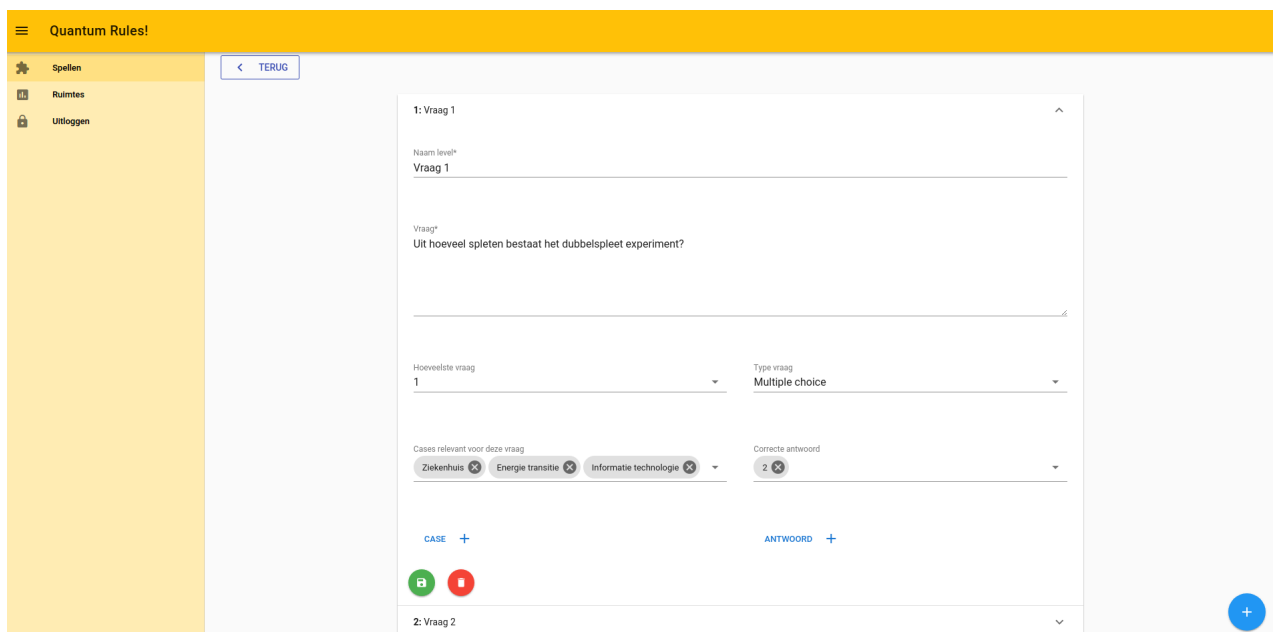


Figure 4.5: The levels.

# Rooms

In this section the admin is enabled to manage the rooms within the application. All existing rooms are listed with the indication if it is opened or closed. When a room is opened, players are able to join it. When closed, this is not possible. If a lab session has finished, the admin can close the room to prevent that any more data is added. The name of the room is not shown to anybody but the admin. The sole purpose of this field is the administrative task. By naming the rooms logically eg. "Room - <schoolname> - <date>" it becomes easy for the admin to manage the rooms necessary.

When the room is created, the unique code for the room is automatically generated. This identifier has to be entered in the login screen by the players.

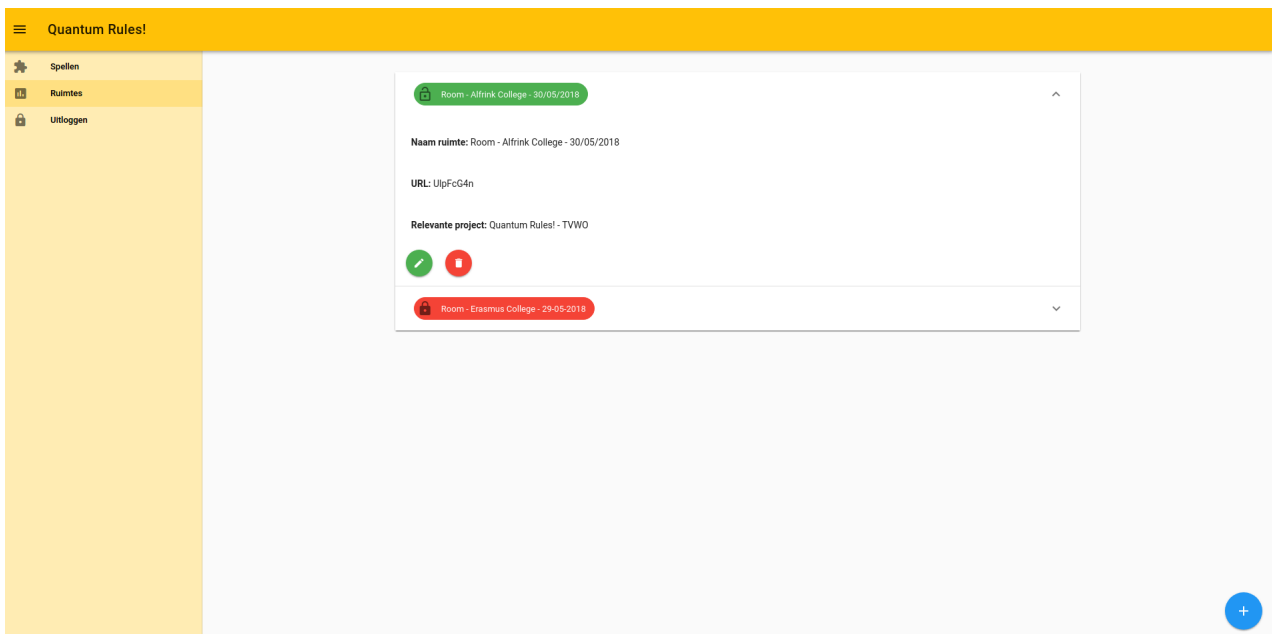


Figure 4.6: The rooms.

# Chapter 5

## Choosing technologies

When designing an application both the data storage as the visuals are important. Therefore, the client and API can be considered as separate projects with each their own considerations.

### 5.1 Client side: Vue.js

Vue.js has been chosen as the javascript framework to develop the client side of this application. However, the choice for this framework is not randomly. Currently more and more front-end frameworks exist and choosing one becomes more and more difficult [GCP12]. A few years ago jQuery would be the choice by default but in modern responsive times with the desire for a growing amount of choices frameworks such as Angular, Backbone, Polymer, React and Vue keeping expanding their user base.

A survey with over 28,000 thousands responses from developers all over the world [js-17] led to an interesting graph as shown in Figure 5.1.

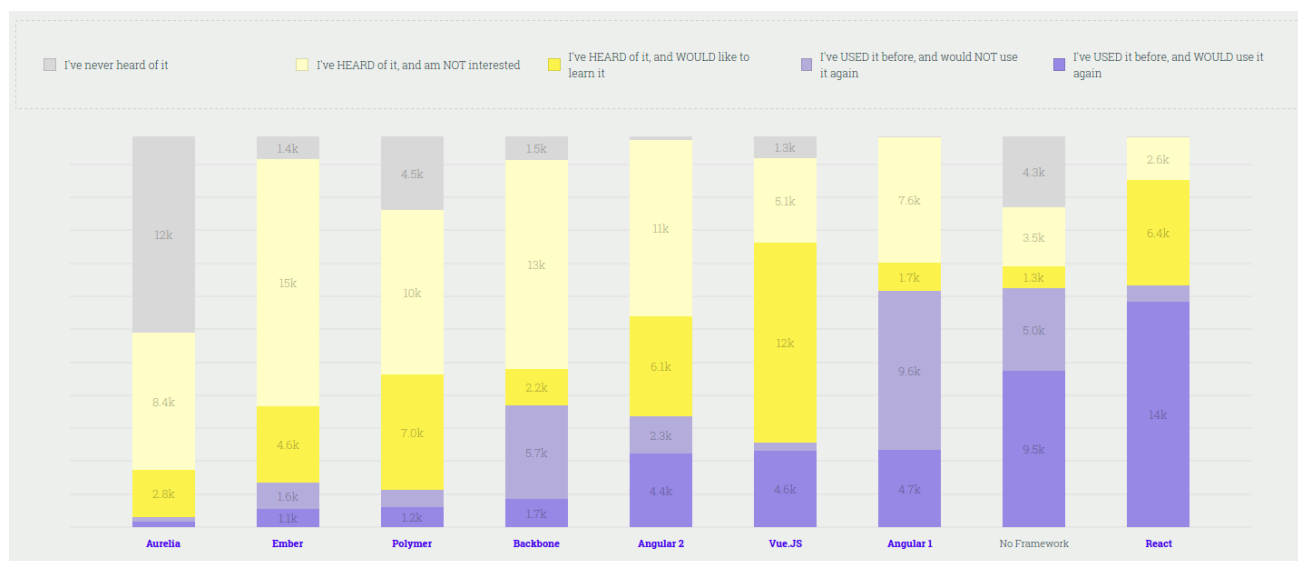


Figure 5.1: Comparison of front-end frameworks and their popularity [js-17].

A first distinction between all these frameworks is the choice between two-way vs unidirectional data binding. Angular, Backbone and Vue traditionally implemented a two-way data pattern. Contrary to these frameworks, React and the newer Angular 2 embrace a unidirectional data flow.

The main difference between these two types is that in a two-way flow, the view and data will update each other whenever one of them is changed. In a unidirectional flow only one way of communication is allowed meaning that changes in the view will trigger an event for which components can listen. The component will then update the model accordingly after which the view re-rendered.

In the requirements of this project, maintainability of the application is of great importance. When considering the two-way or unidirectional data flow, there is no real difference when maintaining the application [MG16].

As a developer, I always wanted to work with a framework such as Angular. When working on projects in Angular I had a lot of difficulties in understanding this rather complex framework. In the search of a more lightweight framework I stumbled upon Vue.js. It was no coincidence that I immediately liked Vue.js as it has a lot of comparisons with Angular as the created itself, Evan You, mentions: [Cro16]

*“I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight without all the extra concepts involved? I was also curious as to how its internal implementation worked. I started this experiment just trying to replicate this minimal feature set, like declarative data binding. That was basically how Vue started.”*

I am not the only one liking the idea of Vue.js as can be concluded from Figure 5.1. Another important source for programmers to develop ideas is GitHub, a platform where developers can maintain their code, collaborate with other developers and share projects with the world. Having a popular project here can have a large impact. When considering the front-end frameworks that gained most popularity over the year 2017, Vue.js came out on top [Ram17] as can be seen in Figure 5.2.

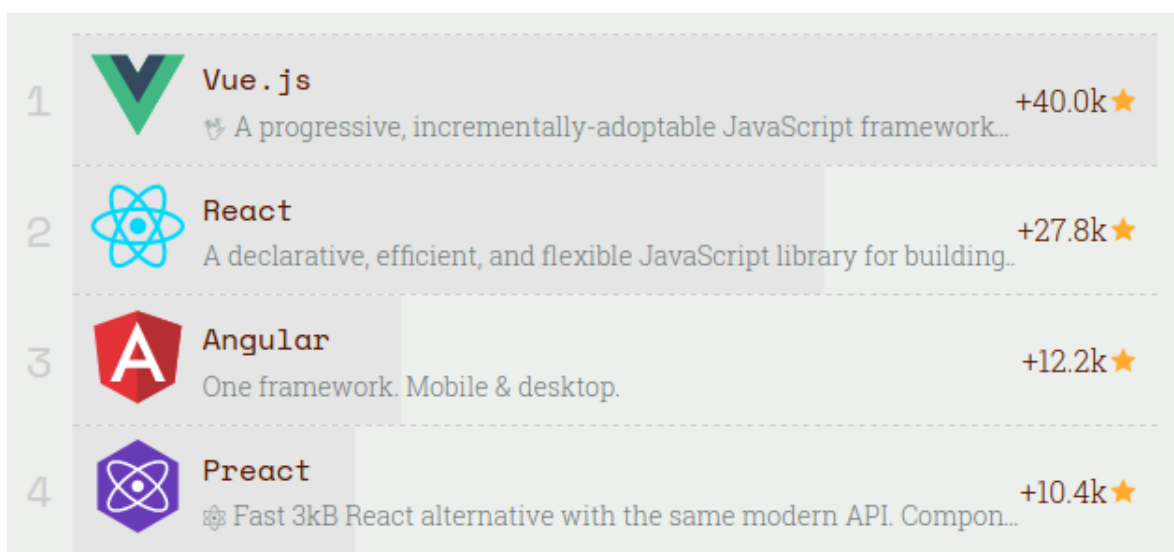


Figure 5.2: Largest popularity increase in 2017 of front-end frameworks [Ram17].

With this data the choice for Vue.js has been made clear. It is important to mention some negative aspects of using such a framework. As mentioned before, it is becoming more and more common for people to develop their own framework and publishing it as open sourced for any developer to use and contribute to. The side effect of this all is that lifecycles of certain frameworks are destined to be short [All16], as can be seen in Figure 5.3, inevitably resulting in loss of support on the long term for most.

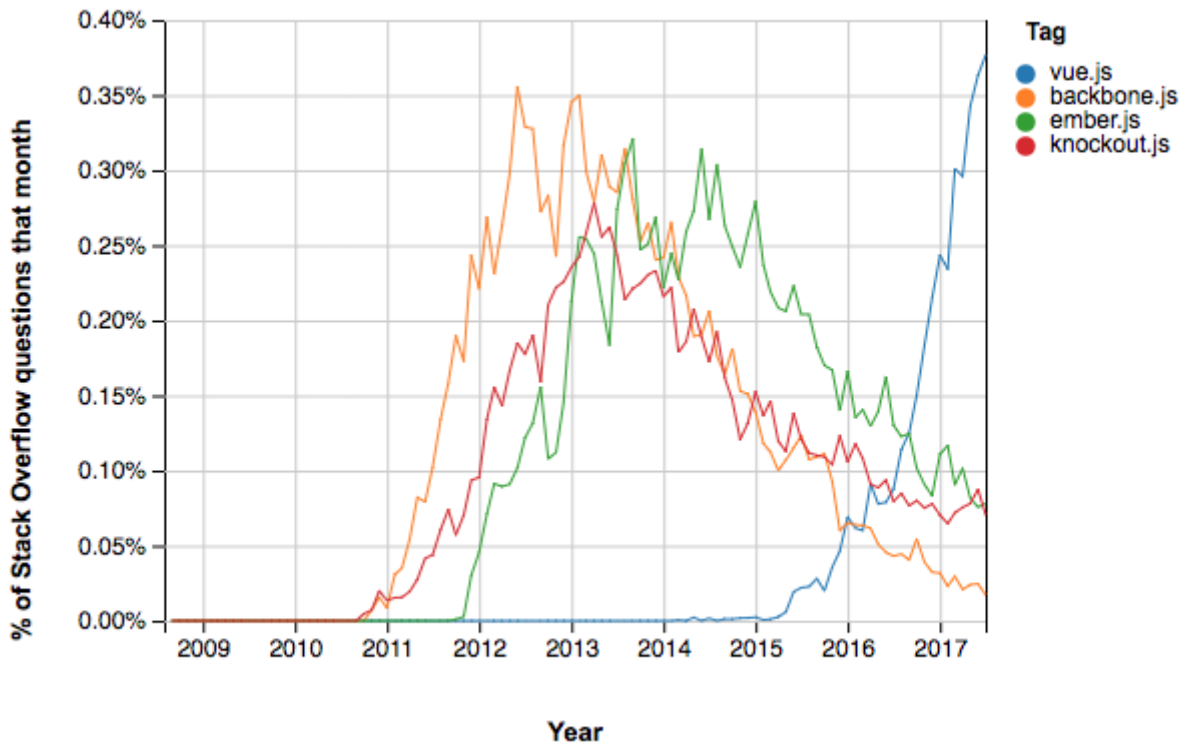


Figure 5.3: Javascript framework lifecycles [All16].

Currently Vue.js is booming in popularity and most messages are more than positive about this framework.

Another aspect that has to be taken into consideration when choosing a framework is speed. Slow performing frameworks will lack in user experience as re-rendering components will take more time. In an experiment where a HTML table was created with a thousand rows of data with each six columns Vue performed very well relative to other frameworks [Ale16]. With all types of object manipulation such as create, update and delete Vue kept performing well compared to mature frameworks such as Angular. The overall conclusion of this paper was that “[Vue] was one of the fastest framework when creating and updating elements. It also made a good result when deleting elements.” meaning that Vue.js is a good choice when taking speed performance into account.

In another research conducted by the University of Amsterdam, a lot of Javascript frameworks were compared on maturity, ease-of-use, Performance, browser support, modularity, testability, routing and templating. Their overall conclusion was that “Ember.js, Vue.js and AngularJS share a lot of common ground, but we think that AngularJS is the better framework of the three. We see no reason to use Ember.js or Vue.js instead of AngularJS.” [Koe16]

which means that Vue.js is not a bad choice but simply does not show significant advantages over AngularJS. However taking previous references into consideration there has been shown that Vue.js does show advantages over AngularJS, concluding that the choice for Vue.js is very safe and well concerned.

### 5.1.1 Vue-router

Routing can be complex but by using the official router for Vue.js: vue-router, managing the application routing is no issue at all. Using this enables the definition of all paths that the viewers may access and what content should be loaded when the viewer loads a certain path.

### 5.1.2 Vuex

Using Vuex is one of the most important decisions for this application. In a default Vue.js application every user loads its own Vue instance with the corresponding data. Every view within the application will have their own state. Within a Vue.js application there are three core parts as mentioned in [You18].

- State: The source of truth that drives the application
- View: A declarative mapping of the state
- Actions: The possible ways the state could change in reaction to user inputs from the view

These core parts are in line with the famous model-view-controller (MVC) principle researched in the book Design Patterns (Gamma et al.) [EG94]. The one-way data flow can be visualized in the following Figure 5.5.

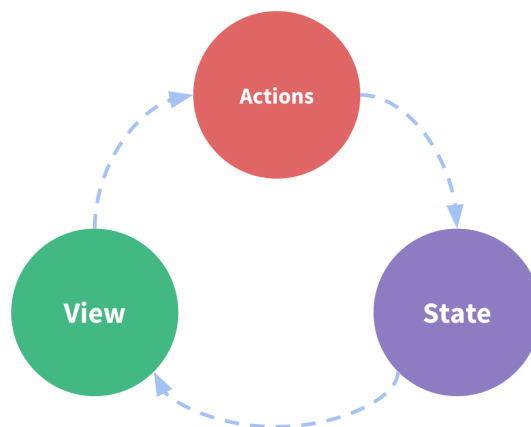


Figure 5.4: Visual representation of the Vue one-way data flow [You18].

Whenever multiple views depend on the same state or actions from different views should alter the same state, this one-way data flow becomes too complex to use. Vue.js supports the use of properties that can be passed to different views meaning that nesting components and adding properties to each of them might work

but it clearly is not a neat solution. Using Vuex solves all the issues as Vuex extracts the shared state of all components into one object, a singleton.

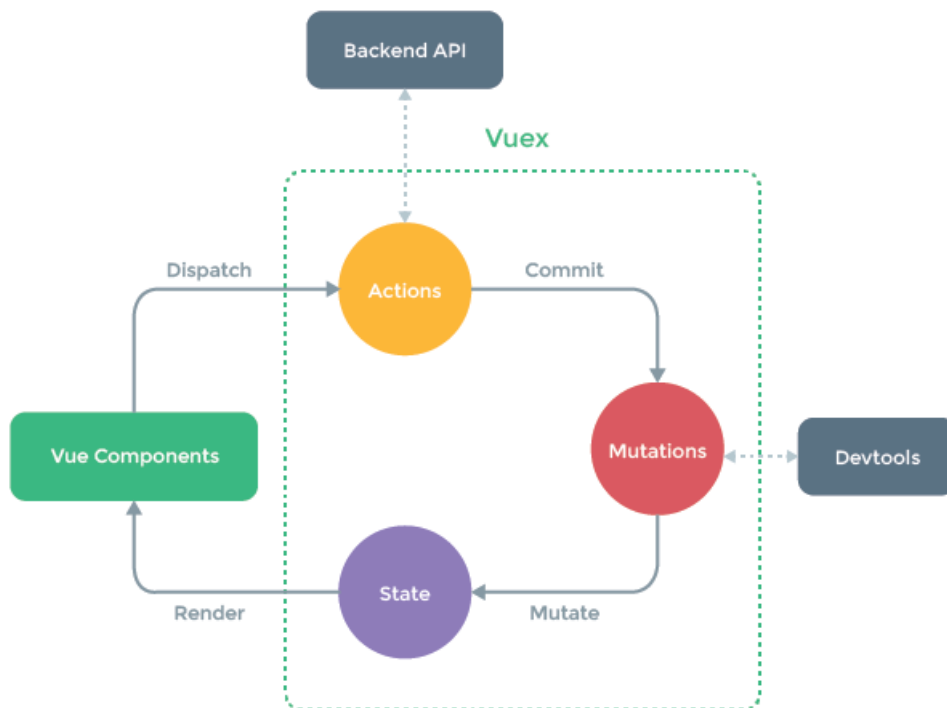


Figure 5.5: Vuex store implementation [You18].

This singleton becomes the store for the entire application. Meaning that every view can enter the store to perform CRUD operations on the selected data. After each store change, all the views that use that section of the store are updated.

Another reason why Vuex has been chosen was because of its excellent integration with Socket.io as can be read in Section 5.3.

## 5.2 API: Loopback.io

When considering the data used in this project it has to be mentioned that one of the most important non functional requirements is maintainability. At such point the complexity of the data used has to be mapped. When large amount of data needs to be accessed quickly or a lot of complicated custom logic needs to be used different frameworks are considered than when the mentioned terms do not apply.

In Chapter 2 this project with all its entities was defined and the formal architecture was given. From this, it can be concluded that the data used in this project is not complex at all. The projects' scope consists of

multiple entities with traditional relations meaning that relatively simple frameworks might suffice. The main advantage of Loopback.io over the other considered frameworks is mentioned on the strongloop website itself [Gor14]. In the example provided there, the same RESTful API is created with multiple frameworks. Comparing the amount of lines code needed gives interesting results.

Framework	Lines of code
Express.js	41
Restify	41
Hapi.js	64
Loopback.io	14

Table 5.1: Lines of code needed for same API [Gor14].

With only 14 lines of code it is clear that Loopback.io is the winner, implying that with minimal effort a decent API can be created with the basic CRUD operations. Meaning this API can Create, Read, Update and Delete on all initialized endpoints.

The website is obviously biased as strongloop is a commercial product owned by IBM, that uses loopback. However, the mere fact that IBM acquired the strongloop company in 2015 [IBM15] proves that loopback has great potential. This acquisition also changed the maintainability of loopback, now IBM can provide enterprise level support and maintenance.

Besides the simplicity that Loopback.io offers, there is a visual API explorer that allows the viewer to see all endpoints with the relevant methods. Here the full functionality of the API can be tested with a nice interface as can be seen in Figure 5.6.

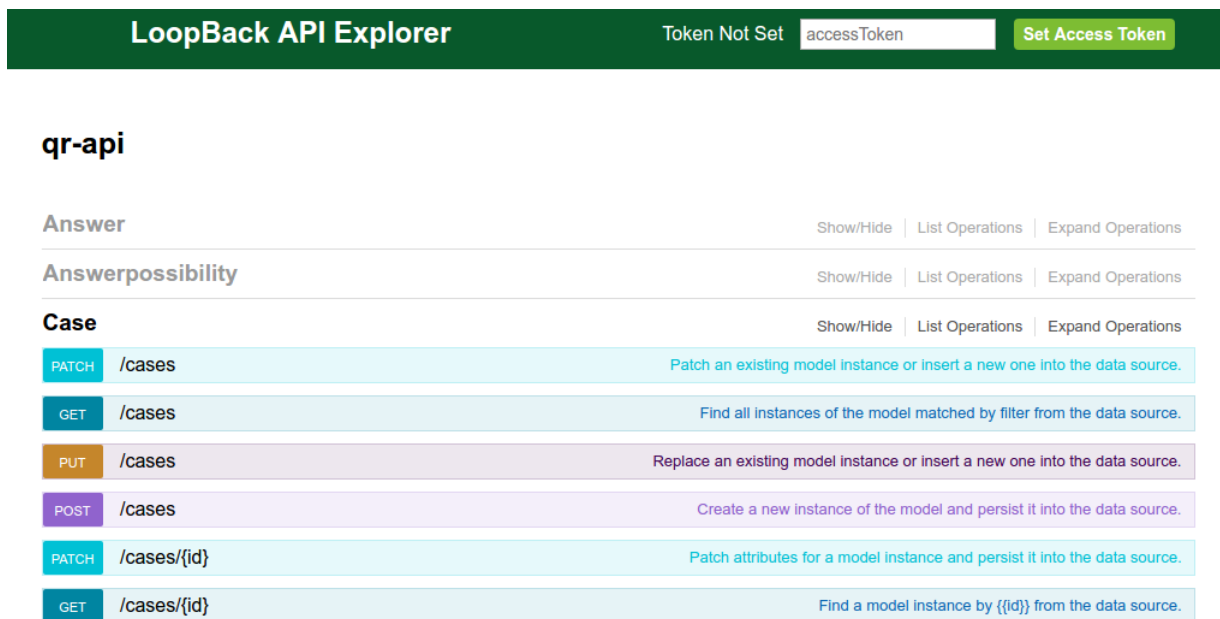


Figure 5.6: The loopback explorer.



The major drawback of using loopback is that with the simplicity, the behind the scenes obscurity increases. The learning curve of understanding what loopback does behind the scenes, the fact that users do not need to write a lot of code, is quite steep.

Because the data for this project is simple, using Loopback.io is an excellent choice. With minor effort, the desired result can be obtained. Whenever future developments require a more complex datastructure, switching to a different API can always be considered.

## 5.3 Socket.io

Implementation of Socket.io is fundamental for the application. In traditional communication between a client and a server the client will request data and the server will respond with the corresponding data. In the requirements, as mentioned in Chapter 3, it has been elaborated that the players of the game should see progress of the cases realtime. Using the traditional techniques that would only be possible by continuously requesting data from the server. Spamming a server like this is never a good idea and therefore using the modern framework Socket.io is the desired solution.

By using Socket.io a realtime bidirectional event-based communication is possible, meaning that both the client as the server continuously keep their connection open. Changes on either side can immediatly be pushed to the other resulting in them both keeping up-to-date with most recent changes. The client can send events to others using the `socket.emit()` method and listen to those that have been sent with the `socket.on()` method. This framework is based on WebSockets, a computer communications protocol [IF11].

### 5.3.1 Integration with Vuex

One of the prime reasons that Vuex was used within the application, is because of the excellent integration with Socket.io. How Vuex works has been explained in Section 5.3, triggering a mutation in the store is all that has to be accomplished. When the stores of all players in the same room are updated in identical ways, they will all have the same state meaning that all players will see the same progress of the game. Whenever a player  $x$  is working on a experiment and a player within the room ( $x$  or someone else) solves the level that has to be solved, all players that are currently viewing that level, should automatically see the next level. Also, players that are not viewing the selected experiment, should see the cases progress whenever a level with related cases is solved.

As explained in Section 5.3, whenever the store is updated, all views that are relevant to that store are also updated. Whenever a level is answered Socket.io will fire the `socket.emit()` event. All clients will use the `socket.on()` event to detect these changes. The most convenient part is now that Vuex can automatically detect these changes instead of using the `socket.on()` listener within the Vue application. By adding the prefix `SOCKET_` in the Vuex mutations, a socket event called "connect" will automatically trigger the mutation

called `SOCKET.CONNECT`. Using this logic, the store mutations `SOCKET_LEVEL` and `SOCKET_CLEARED` within the room will ensure that:

1. All players conducting the same experiment will always view the level that needs to be solved
2. All players will see the progress of the cases

Due to this seamless integration with Vuex, the implementation of Socket.io and state management was fairly easy.

## 5.4 Security

Creating an authentication system for a javascript application can be complex. Using an existing framework solves all complexity. For this application *Auth0* has been chosen to handle all security for the admin.

Type of endpoint	Included endpoints	Admin calls	Player calls
Private	/roomlevels & /players	All methods	GET
Public	/cases & /experiments & /levels & /projects & /rooms & /answerpossibilities	All methods	GET & POST

Table 5.2: Endpoints security per role

Most endpoints within the API are private meaning that only a logged in admin, with *Auth0* account, can make a call to these endpoints. The methods used within the application are all default HTTP methods [FGM<sup>+</sup>99]. Players can GET specific data as they need to be able to perform this type of request to load the game prerequisites. The only other type of method they are allowed to called is POST on the required tables for the game to be played. These are the 'Player' and 'Roomlevel' table as shown in Figure 4.1, the corresponding endpoints for these table are the so called public endpoints within the application. A random player cannot change a GET request to DELETE to delete admin data as *Auth0* will block this unauthorized request. Using the *Auth0* framework together with the recently explained logic, the application is secured.

# Chapter 6

## Quality assurance

After completion of this thesis, the first version of the digital Quantum Rules! platform will be released. Ensuring that the application does not fail upon delivery a few agreements have been made with the mandator.

### 6.1 A priori

Together with the mandator it was decided that the development would be agile [SB02] as requirements were expected to change during development. One of the biggest drawbacks when working agile is that documentation is often not completed. Therefore the final sprint, before finishing the thesis, will be documenting the code. As this step is the final stage before wrapping up, it is guaranteed that the documentation is up-to-date with the final requirements and architecture. Delivering a well documented application is only one precaution made to guarantee quality.

In the final stages of development the mandator and developer tested the application thoroughly to find any bugs or details that could be improved on. To test the application the following scenario was created:

- Create a project containing three experiments
- Connect four devices (A desktop, laptop and two different mobile phones)
- Three devices only work on one experiment and one device works on different experiments
- Each tester controls two devices
- All types of questions as mentioned in Chapter 4 should be included in the test

By performing the test as explained above, all crucial scenarios of the application are covered. The result of this test session was very positive according to the mandator. Minor feedback was given and implemented to improve the experience on all devices. This performed black-box [Lu001], system, testing has covered a lot of

test scenarios. The mandator was present during the testing which was crucial to for the black-box testing. He had little knowledge of the technical side of the application, he only knew what it was required to do. Due to his lack of knowledge about the application a realistic scenario was tested compared to when only the developer would have tested. It must however be mentioned that during the development, the developer did test all functions extensively.

This application has no white-box [Lu001] testing implemented. Due to the time constraint related to this thesis and the agile development method, it was negotiated with the mandator that there would not be any white-box testing. It is possible to implement this type of testing in a newer version of the application if desired.

## **6.2 A posteriori**

Due to the fact the application will be well documented, other developers than the original will be able to work on the application. However, the original developer will be available to keep the application running during the first 6 months of deployment. The Quantum Rules! platform will be the owner of the application meaning that anybody, that the mandator allows, can work on the application. These second stage developers will implement other features that the mandator request while the original developer will only be responsible for the application as he delivered it. Issues introduced by new features, not developed by the original developer, are not for him to fix. All issues and feature requests will be documented in the repository of the project ensuring they are stored in a central place and not lost. The code is delivered by granting people, as indicated by the mandator, access to this repository. Also, a stable version of the application will be deployed on the network of the Leiden University.

## Chapter 7

# Conclusions

This thesis described the development of the online Quantum Rules! environment. A virtual place where students will be stimulated even more to participate in physics experiments. The elements of the escape room trigger the players to keep pushing themselves to solve questions but also to help others in completing their experiment. The combination of helping fellow students while competing them are perfect complements ensuring that learning never becomes boring. The entire application in combination with how the laboratory adds elements of an escape room are an example of using gamification [HKS14] to increase the appeal of learning. This thesis has succeeded in applying the gamification concept as the Quantum Rules! platform has now become more interactive and fun for the students, stimulating them to keep learning.

## Evaluation

Developing an application for high school students with Vue.js and Loopback.io has been an excellent choice. Using these two frameworks a lightweight, responsive and modern application has been developed. Using add-ons such as Socket.io and *Auth0* resolved a lot of complex issues. The combination of Socket.io and Vuex made one of hardest requirements, P7 as mentioned in Section ??, fairly easy to implement. Overall it can be concluded that choosing the right frameworks and learning to work with them has proven the most difficult aspect of the entire development process. The first few months were flying by while only minor steps were being made. Eventually this learning process did result in a higher level of development. After new aspects were learned, old code was often refactored. Also, whenever the mandator changed requirements closer to the end of the project, implementing these changes went faster and faster.

Using an agile development method was a correct choice as requirements did change over time. The major mentioned drawback of agile was lack of documentation, this is what happened in the end. Within the agile development method, it was decided to do one more sprint at the end of the project to write all the documentation. It was however rather difficult to document all code in hindsight as the project was functionally

finished and motivation was lacking to document code and it took some time to understand all the written code again. Another major problem concerning the documentation was the fact that the most difficult aspect of developing the application was learning the used frameworks. For the documentation of this application it can be said that the most crucial aspect is to fully comprehend the used frameworks. On the other hand, due to changing requirements, documenting while developing would result in a lot of extra work as the documentation would not be correct anymore in a later stage of development. The fact that it was decided to write all the documentation in one final sprint at the end can still be considered a correct choice as the most important aspects about the application are documented correctly.

## **Future work**

Nearly all requirements have been implemented during this first stage of development. The only requirements that are not included within the application are A8, A9, A10 and P8 as mentioned in Chapter 3. The impact of A8 and A10 is that the results of rooms cannot be viewed by the admin yet. The data is however all stored correctly meaning that implementing this feature in a later stage still provides insight in the rooms created in the first version. Also, the admin cannot view the progress within a room from the admin panel (A9 and A10). The impact of this is that the admin cannot know how far the players are. The solution is to walk through the lab and checking with the players themselves or simply join the room as player without actually playing. Furthermore, the players cannot compare stats with each other (P8) meaning that there is slightly less competition in this version. The fact that the players see the cases progress creates the experience that others are solving their levels, resulting in an extra stimulant for all players to continue with playing.

Some minor wishes from the mandator have not been implemented yet. Within the application it is not possible to answer question 6 as mentioned in Section 1.1 as media cannot yet be uploaded. Within a future version, it is desired that this is possible. Also, during experiments the students often use external programs, such as labview [Ins], to generate and process experimental results. It would be nice if an automatic connection could be made between the application and such an external program enabling students to automatically transfer data. Adding an interface where students can create and save graphs within the Quantum Rules! application is another wish. This would enable students to plot their data automatically within the application which can then be checked. Finally, it would be nice to add a programming environment to the application providing a method for the students to quickly write a python script to calculate some variables. The output of such a program can then automatically be saved in the application. In short: currently a lot of external programs are still being used by the students, it is a major wish that these can be integrated within the application as much as possible.

# Bibliography

- [Ale16] Svensson Alexander. Speed performance comparison of javascript mvc frameworks, 9 2016.
- [All16] Ian Allen. The brutal lifecycle of javascript frameworks. <https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/>, 11 2016.
- [Cro16] Vivian Cromwell. An interview with evan you. <https://betweenthewires.org/2016/11/03/evan-you/>, 11 2016.
- [EG94] Ralph Johnson en Richard Helm Erich Gamma, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 10 1994.
- [FGM<sup>+</sup>99] Roy Fielding, James Gettys, Jeffrey Mogul, Henrik Nielsen, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol. *Internet Engineering Task Force (IETF)*, 6 1999.
- [GCP12] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. Comparative evaluation of javascript frameworks. In *Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion*, pages 513–514, New York, NY, USA, 2012. ACM.
- [Gor14] Alex Gorbachev. Comparing express, restify, hapi and loopback for building restful apis. <https://strongloop.com/strongblog/compare-express-restify-hapi-loopback/>, 4 2014.
- [HKS14] J. Hamari, J. Koivisto, and H. Sarsa. Does gamification work? – a literature review of empirical studies on gamification. In *2014 47th Hawaii International Conference on System Sciences*, pages 3025–3034, Jan 2014.
- [IBM15] IBM. Ibm acquires strongloop to extend enterprise reach using ibm cloud. <https://www-03.ibm.com/press/us/en/pressrelease/47577.wss>, 9 2015.
- [IF11] Alexey Melnikov Ian Fette. The websocket protocol. *Internet Engineering Task Force (IETF)*, 12 2011.
- [Ins] National Instruments. Labview. <http://netherlands.ni.com/labview>.
- [js-17] Front-end frameworks results. <https://stateofjs.com/2017/front-end/results/>, 2017.
- [Koe16] Jaap Koetsier. Evaluation of javascript frameworks for the development of a web-based user interface for vampires, 6 2016.

- [Luo01] Lu Luo. Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*, 15232(1-19):19, 2001.
- [MG16] Erik Magnusson and David Grenmyr. An investigation of data flow patterns impact on maintainability when implementing additional functionality, 2016.
- [Ram17] Michael Rambeau. 2017 javascript rising stars. <https://risingstars.js.org/2017/en/>, 2017.
- [SB02] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.
- [You18] Evan You. What is vuex? <https://vuex.vuejs.org/en/intro.html>, 2018.