Creating a web application to match a sport partner

Jérémy Maret

Bachelor’s Thesis

Degree Programme in Business information technology

2019

Haaga-Helia
University of Applied Sciences

Abstract

Date

Author(s) Jérémy Maret	
Degree programme Business Information Technology	
Report/thesis title Creating a web application to match a sport partner	**Number of pages and appendix pages** **36**

The goal of this thesis is to create an application that could be used to replace and enhance the existing sport clubs booking system. This application consolidates a traditional booking system with a connection system. As it can sometimes be complicated to find partners, this solution aims to make this easier. Players will be able to create a private or open party. If an open party is created, everyone is free to join the game. It offers an opportunity to play more often and to meet new people. Thus, it helps to make sport more attractive.

Firstly, the technologies used for the development of this application are explained, then the design is illustrated to witness how it will look like and understand the decisions taken. Following this, the implementation is described starting from the database creation up to the web application. Once finished, the application was tested by some people to see if the application is easy to use and is suitable for a sport club. These tests are described in the thesis as well. Finally, the future of this application is mentioned at the end of the thesis in the conclusion.

As part of this thesis, a full stack application has been created. The application has been written in React and it uses the API created in Node.js. The API aims to let the application interacts with data stored into the MariaDB database. With the help of a calendar, it is possible to book party or/and to create an open party. It is also possible to join any existing open party if there is available space for players.

Keywords

Full stack, React, Node.js, Redux, Hapi.js, Booking system, Matching partner

ACKNOWLEDGEMENTS

I would like to thank my advisor, Juhani Valimaki for guiding me all along the thesis. For giving me good guidelines to succeed my thesis with more ease.

I would like to thank Salomé Boudet, Bryan Ijeh and Martin Boujol for dedicating time to proofread my thesis.

I would like to thank Raphaël Azzoug, Kevin Berret, Julien Dubuis and Clément Vasser for taking part of the usability testing and for going through the tests with seriousness and enthusiasm; Which was much appreciated.

Table of contents

1 Introduction

Did you ever have to deal with the difficulty of finding a sport partner? Did you ever have to cancel a practice because your partners were not available at the time you wanted to play? This is a common struggle that is happening to many people. When someone plays a sport that needs one to four partners, like most racket sports for example, it can be complicated for some people to find partners every time. Sometimes, despite the desire to play, it is not possible to practice because your partners are not available or they are just not in the mood to play. As it is a common issue, it can be interesting to gather all these people into an application that will help them to find opponents. Playing sport should be easy and only a fun time.

This thesis will be about creating a solution to help players to find partners. Every racket sport club has a booking system thus, it can be relevant to rethink it to make it more efficient. We are living in a world that is more connected than ever before and networking is really important nowadays. Indeed, people want to be connected. As a consequence, if a sport club offers a solution to meet new people, it can be a really efficient added value.

1.1 The background of the project

This idea came to my mind one year ago while I was struggling to find a tennis partner. I always have a certain difficulty finding a partner who has free time overlapping with mine. Some of my old partners stopped playing this sport, others do not have the same level and for the rest it is every time complicated to find a day and a timetable that fit our different schedules. Therefore, it became really challenging to find opponents. As racket sports cannot be played alone, people sometimes just give up faced with this difficulty. By looking at several sport clubs, I did not find one that offers something to resolve this issue.

Thereby, I decided to think about an application that can help clubs' customers practice sport differently. This project aims at making it easier to play sport as you will have the opportunity to find partners that fit your schedule and it will obviously open some doors to meet new people. My first idea was to create an application like the social dating applications by offering the possibility to match with a partner but I quickly changed my mind because this kind of application has a small chance of success in this kind of field. Sportsmen are most of the time attached to a certain club and therefore they only want to play in this club. That is why I had the idea to enhance the available booking system that can be found in sport racket clubs by adding a possibility to connect with players.

1.2 Work statement

The purpose of this thesis project is to create an application that manages sport clubs booking system so that players can find partners to play with.

1.3 Objectives and deliverables of the project

The purpose of this thesis is to deploy the first version of this application with a working booking system that gives the possibility to create a reservation with known partners. This system then has to be conceivable to create public games that every club's member can join. I will show the application to some people to have their opinion about the usability and about the concept. This solution must be easy to set up in every sports club and user-friendly.

This project is a great opportunity to improve my full stack development skills as a database, a backend and a frontend need to be written. In addition to this, I will develop my skills in project management because the project will have to be managed correctly in order to be successful. Finally, I could use my own creation after the thesis to create a Start-up and see if my project can be successful.

1.4 Out of scope

This thesis will not cover the research of customers as it is only the creation of the application. The usage of token for the communication with the API will also not be covered and the administration for the club is out of scope. It will not have an administrator page to edit the club information, their courts and every club related administration. This is because this thesis focuses on the booking system function.

2 Theoretical framework

The different chapters below are about the tools that will be used for the creation of the application. They help to give a better understanding of what these tools can provide to the development. First, it will be about a tool that helps to control the project in its entirety, then some tools used for the frontend will be presented and finally, the backend tools will be introduced.

2.1 What is Git?

Git has a really high notoriety as it is the most used version control system. It was developed in 2005 by Linus Torvalds, who is famous for his creation of the Linux operating system kernel. Git can be defined as an Open Source Distributed Version Control System. Open source as the source code of this software can be readable, editable and improvable. (What is git ?, 2019a)

It offers three main tools, the first one is a control system so you can store content into Git, the second one is a version control system, meaning you can have a control of all the versions of the files stored into this software. A history is created and you are then able to see what has been added or removed between each version. The user is also able to get back to an older version if it does not work as expected in the latest version for instance. The last functionality is that it is distributed. It indicates that the files are not centralized. There is always a remote repository which you can find on a server like GitHub for example and one or more local repositories stored into the local files of developers. With this option, the developers are able to work simultaneously without disturbing each other. (Torvalds, 2007) (Sridhar, 2018)

The subchapters will deal with the branch and the merging. These options are important features that must be understood when Git has been chosen.

2.1.1 What is the branch and what is merge?

When someone wants to work with Git, he will have to face the branch feature. This is certainly the most impressive feature of Git by the fact that its lightness offers the possibility of moving from branch to branch easily and quickly. It is important to understand how the branch feature works if you want or have to work with Git. This feature allows the developers to cooperate on a same project.

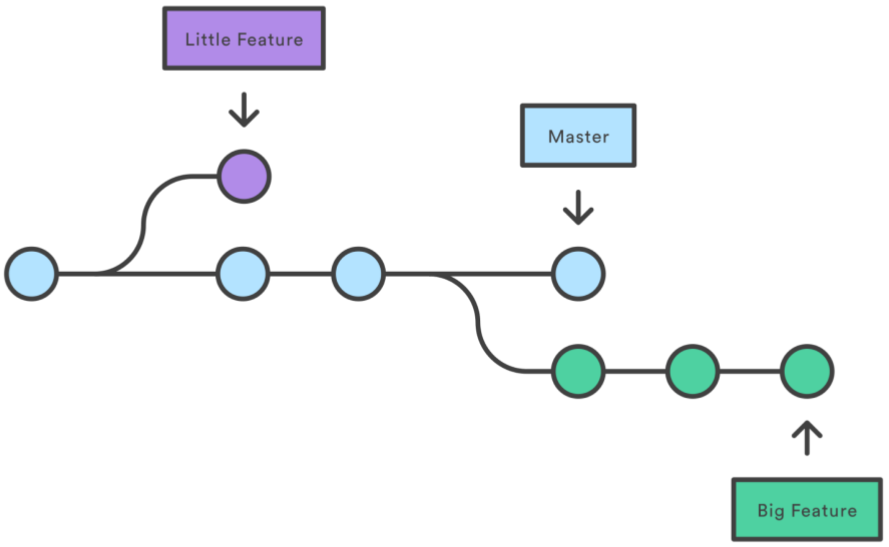


Figure 1: branch in Git (Atlassian, 2019a)

A branch in Git will allows you to work on an exact copy of the productive branch without disturbing it with your change. When a new feature is in development, a new branch is created like you can see in the Figure 1, the purple and green branch has been made to work on features. These two branches work regardless of the branch master, in others words, they are independent source code. All tests and work can be made until the branch is ready to be put in production. When this state happens, the common practice is to merge the production branch into this branch to see if some changes happened during the work and when this is clear, another merge can be made in the opposite direction (the new branch into the productive branch.) (Git-scm, 2019a) (Git branch, 2019a)

But what is a merge in Git? "*Merging is Git's way of putting a forked history back together again*" (Bhabesh). With a merge, two branches can be fused to make one, so the work of both of them will be united. As you can see in Figure 2, a second branch has been created from the master branch. After this handling, some work has been made and finally a merge happened to put the new feature in the master branch so the new branch disappeared. (Git Merge, 2019a)

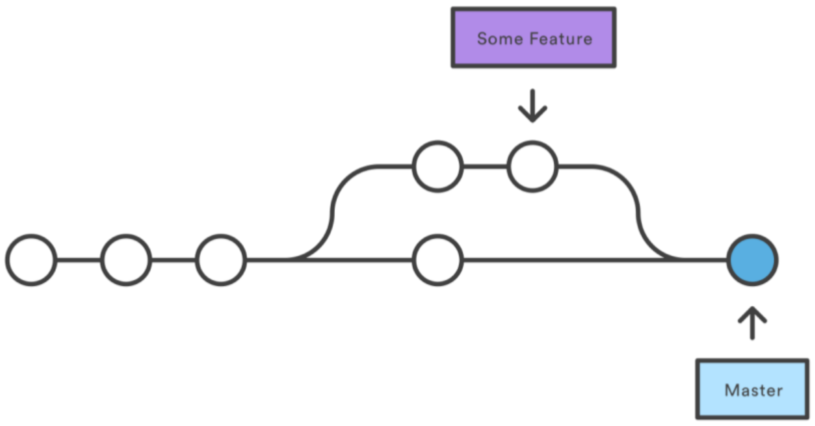


Figure 2: merge in Git (Atlassian, 2019a)

2.2 React

React is an open source JavaScript library even if it is frequently falsely presented as a framework. This library, developed by Facebook, is used to build user interfaces. React allows to create large web applications with the possibility to change the data without refreshing the page. It allows developers to make a complex application more easily than if they wanted to code it only with JavaScript. So, with this library, an interactive user interface can be made easily. The variables of a view can be set up as a state and when an event happens, only the components with this state will be rendered again, this feature gives a faster response to the views. React is declarative so the developer does not have to take care of the control flow, the code becomes easier to understand and debug. React just needs to know what it has to do, not how to do it. React is also component-based, this means that a view is composed by many components encapsulated into each other. Therefore, a complex user interface is decomposed into many fewer complex components that can be re-used everywhere. Each component manages its state independently. (Suzdalnitski, 2018) (Pandit, 2018) (React - README.md, 2019a)

Facebook developed a tool with the help of its community named “create-react-app” that will configure everything a developer needs when he wants to start a new web app from scratch. If this tool is used, the coder can directly start coding the application without bothering himself with every configuration and tools that he normally needs to configure. (create-react-app, 2019a) This library can also be used to create native mobile application with React Native, this framework allows to create single application that will work on IOS, Android and UWP by using their own native component. (React Native, 2019a)

All these points make React a solid option when a choice of a framework has to be made to develop a frontend application. As you can see in the Figure 3, this graphic, based on the survey of “State of JS” in 2018, shows that React is the most used framework/library for JavaScript. As React is an intensively used library, it provides a guarantee that a big community is behind it and the developer is insured to get full support.

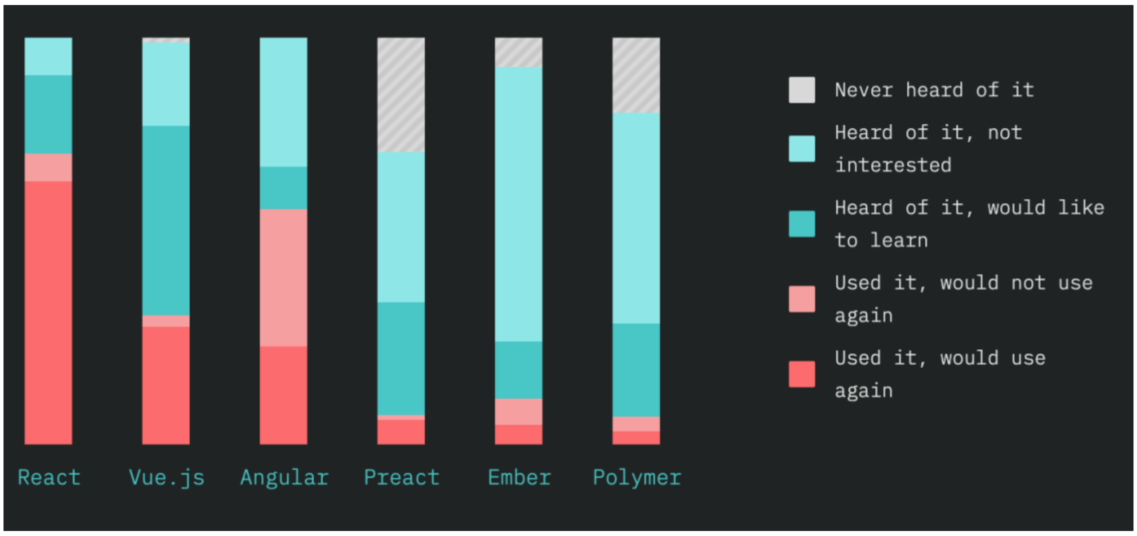


Figure 3: most used framework (Front-end Frameworks - Overview, 2018)

2.2.1 React vs Angular vs Vue

As you probably noticed in the Figure 3 from the previous chapter, three frameworks/libraries are sharing the top best: React, Angular and Vue.JS. They are the most used. As a consequence, when a web application has to be made, a smart choice between them must often be made. These three frameworks have different architectures that can be a pro for a project but can at the same time be a con for different project. That is why you have to be aware of them when you are thinking about which framework you want to use.

Like React, Angular has been created by a big company called “Google”. It is the most mature framework, as it was created in 2010 but, it is also the heaviest (approximate size 500 KB). (Daityari, 2019) Its size was the main reason why Evan You decided to create the third framework this chapter will speak about: Vue.JS. He was a developer for Google and he used Angular in his job. (Cromwell, 2016)

“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?” (You, 2016)

Angular was sometimes too heavy for most of the projects he developed then he started to think about transforming Angular into a lightweight framework. So, in the end of 2014, VueJS was released thanks to his work. It is therefore the least mature of the three but it has the highest growth curve. As you can see in Figure 4, Vue is taking the lead in terms of the number of stars on GitHub. It proves that there is a big craze behind VueJS. (Daityari, 2019)

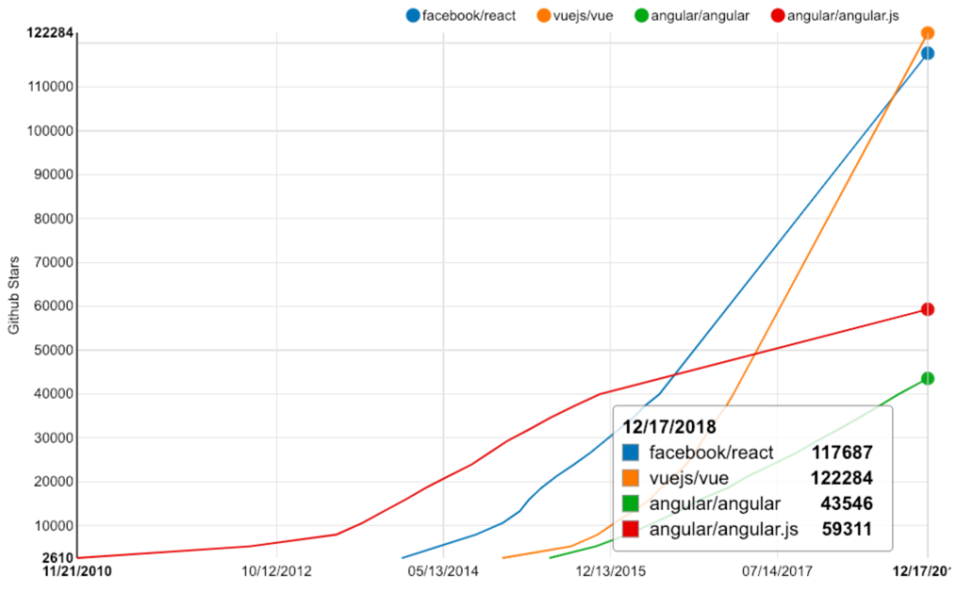


Figure 4: number of stars on GitHub (Daityari, angular-vs-vue-vs-react, 2019)

Now, if we want to have a look at the differences, the first one is that Angular and VueJS are considered as a framework and React as a library. But what does that mean? On the one hand, this is about the freedom and flexibility that they provide. A framework tells you how your application has to be structured. Angular guides you to an MVC (Model-View-Controller) structure and has many “out-of-the-box” functionalities. The developer has less choices and so he has less possibilities of going wrong but as a result of this loses flexibility. (Kumar, 2015)

On the other hand, a library gives many classes that a developer can use and he is free to take it if he has the need. A library can be called by an application that wants to use it: it is code reuse. In contrast, a framework has and use a library to work and call the developer’s code. (Kumar, 2015) So, with the library React, the “view” is the only provided thing, the rest has to be chosen by the developers. This freedom gives the possibility to find the library that fits the most for any particular project but, it also requires to take care of every added library independently when, for example, some updates have to be done. The flexibility of

React forces the developer to think about what he will use before starting to code meanwhile you can directly start with Vue or Angular. (Hamedani, 2018)

The second difference regards the DOM (Document Object Model). React and Vue are using a virtual DOM while Angular is using a regular DOM so what difference does this make? Virtual DOM is a big feature that gives the notoriety to React as it allows React and VueJS to be really fast. How does it work? When the web page is shown, an exact copy of the DOM is created into a JavaScript object. This copy allows for work to be done faster as nothing changes on the screen. When the virtual DOM has been modified, a differential check of the two DOMs are done to find what has been changed. With this manipulation, the real DOM will update only the altered part. The regular DOM works differently, it will wait for a change and only then it will update everything in the render. This one will obviously take more time to re-render as a lot of unchanged components will be rendered again. There are only in some rare cases is a regular DOM faster, this happens when the page needs to be re-rendered every time and that everything is subjected to a change. (Hamedani, 2018)

Another point where we can find a big difference is in the used language. Angular and VueJS are now using TypeScript, a statically typed language, so each variable has to be defined with a type. With TypeScript, the code becomes easier to read and then easier to debug. It provides a good set of Object-Oriented Programming (OOP) features that will give to the code more robustness and cleanness. (Hamedani, 2018) React only uses JavaScript but as this language is one of the most popular nowadays, it can be seen as a professional language. It insures that one can find many answers to each question. If typed variables seem important for a React project, it is still possible to add some features. For example, “Flow” can deal with typed checking and it is created by Facebook. (Petrosyan, 2018)

Mobile developing is more and more important these days and because of this, it is possible to create cross-platform application with all of them. React has React native, Angular offers Ionic and VueJS has Framework 7. As mobile application is not in the scope of my thesis, it is not important to know the deep differences between these three. They all provide something and that is the only thing that matters here. (Petrosyan, 2018)

The last point I will cover is the learning curve, which is a crucial point when a new technology is implemented in a team. It is about knowing how long and how difficult it will be to learn to use a technology. If we start with React, it has a really strong documentation that will help solving a lot of struggles and eventual issues as it uses JavaScript which is now the most common frontend language. Three concepts have to be learned: the first one is JSX, which is a combination between the UI templates and inline JavaScript logic. It uses a language like XML for the mark-up. The second one is that a routing library like react-router will have to be understood and finally a state management library like Redux should be known. These three points will be discussed in the next chapters. With these tools, it will be possible to create a powerful single page application. On the other hand, Angular is harder to learn. TypeScript will be the first big part to learn and then MVC has to be really well understood. Then you will have to learn how to use Angular in the HTML files. The last one, Vue, offers the most flexibility and simplicity and which makes it the easiest to learn. Typescript is not mandatory, so you can pick it up only if you want to. The developer just has to be familiar with some components of it. (Petrosyan, 2018)

After having read this chapter, you should know what framework/library is better for your project. It is important to keep in mind that there is no *better solution*, only a solution that is more suitable for a certain project. Angular will be more convenient for a huge project that wants inflexible rules, React will be more customizable, thus it can be better for a specific project and finally Vue is better for a small application that needs to be developed quickly. (Neagoie, 2018)

2.2.2 Redux

When a complex application is created with React, a need to manage application state appears. At this moment, a library like Redux is needed. But then, what is Redux? In the official documentation it is written "*Redux is a predictable state container for JavaScript apps.*" (Redux, 2019a) This means that it will help to build application that behaves the way it should. As React is a one-way data binding, if a user input undergoes a change, the model state does not update automatically. This is why a state manager must step in. Redux will be aware when a change appears and then it will inform every component that is connected to his store. When several components use the same state, everything is easier. Instead of informing every other component when a user input is edited, all components just ask the current state to Redux. Redux works with every JavaScript environment, it can be used for the client part (with React for example), for the server- side or natively. (Emmanuel, 2018) (Nguyen, 2019a) (Charruel, 2019a)

2.2.3 JSX

JSX (JavaScript XML) is a mark-up language like HTML or XML that allows to create component easier to code and read. The code becomes more human readable and so it is easier to understand and then debug. The best way to understand what JSX offers is to compare a JavaScript code with a JSX code. (Burnett, 2017) For example, this JavaScript code:

```
React.createElement('input', {
  type: 'password',
  name: 'password',
  required: true,
  maxlenght:42,
  value: this.state.password,
  onChange: this.handleChange,
})
```

In JSX, it will look like this:

```
<input
  type="password"
  name="password"
  required
  maxlength={42}
  value={this.state.password}
  onChange={this.handleChange}
/>
```

“React.createElement” becomes obsolete because JSX directly knows that the tag <input> is an element. In the second example (JSX code), you can see a difference with XML and HTML, the value of props is written a little bit differently. If the value is a “String”, there is no difference, it must be written with the quotation mark but for the rest, it must be into brackets. The example below shows that it also is possible to put an element into a variable. As you can see, an element “img” will be created and assigned in the constant “elementImg” and then, it is possible to use the variable where you want. There it is render in the DOM. (JSX in depth, 2019a) (Lerner, 2019a)

```
const  elementImg  =  <img  src={user.profilPicLink}>  </img>;


ReactDOM.render(
  elementImg,
  document.getElementById('root')
);
```

2.2.4 React-router

React-router is a library to handle navigation in a React application. This collection contains React-router-dom and React-router-native. The first one is to create a navigation for a web application and the second one is for mobile application (React Native). With this library, it will be possible to navigate in the web application from one page to another wherever you are in the application. It will be possible to create routes and then use link to follow one of these routes. (react-router, 2019a) (Yusufu, 2017)

2.3 NodeJS

In the past, JavaScript was only used in the client side. It was run by the browser to enhance web pages by making them dynamic and able to change without ~~a~~ reloading. But with the arrival of Node.js, JavaScript found a new utility: it could be used in the server side. (Dionne, 2017) As you can see in the Figure 5, when a request arrives in a NodeJS server, it runs JavaScript to react at this request.

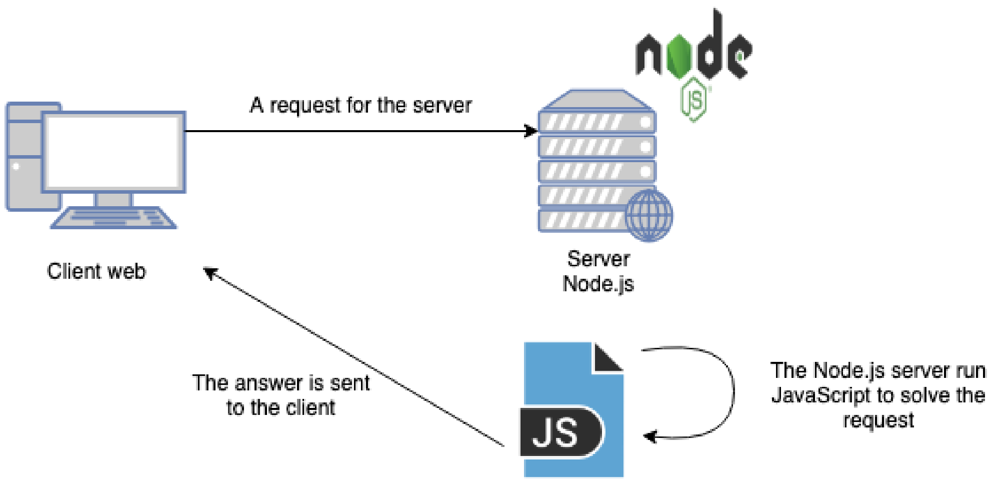

A request for the server
Client web
Server Node.js
The answer is sent to the client
JS
The Node.js server run JavaScript to solve the request

Figure 5: NodeJS execution

But why is it good to use JavaScript in the backend? There are a lot of significant reasons. The first one is that all the scripts are written in JavaScript and it means that the developers are able to use the same language in the frontend as in the backend. JavaScript is the most used language nowadays to build the front of a web application and so it is now possible to write the frontend and the backend seamlessly.

The second advantage is that it is really fast, thanks to its V8 execution engine and its non-blocking model. The V8 execution engine created by Google and used in Google Chrome allows to execute JavaScript code really fast. As it is opensource, Node.js chooses to use it and then it is able to work really fast with JavaScript code. Node also has a non-blocking I/O model, thanks to the event-based language JavaScript. Non-blocking means that it can make asynchronous functions and so, if a function is waiting for something, the engine can work on the next one and it will come back when the last function can go on. A third advantage is the NPM module. It is the mainly used package manager for NodeJS and it offers a huge number of modules and libraries. It is then possible to find something for almost any needs. Node.js has also a big and growing community which ensures good help and documentation. (Nebra, 2018)

In the end, Node.js is a really good choice when a backend or API needs to be built, but when this choice has to be made, we must take into consideration that the logic could be complex with the asynchronous functions and it will not be suitable if the application is going to be used to compute high case. This environment is suitable when it gets a lot of little requests, but if it gets only requests that are asking for big calculation processing, it is better to use something else. (Dionne, 2017)

2.3.1 RESTful API

A RESTful API is an application program interface (API) that follows the constraints of the representational state transfer (REST). But to understand this term, it is better to separate these words first. API means application program interface, and is nothing more than an interface that will give you the possibility to use a web service. An example in real life of an API would be a remote controller for a television. As shown in the Figure 6, a human that would like to watch television cannot do so it if he does not have an interface to communicate with it. The user of a web service does not know how the web service (television) works on the inside, then the API (remote controller) is there to only accept several commands that a user can ask. With the API, the web service will know exactly what the user wants. (Reese, 2019a)

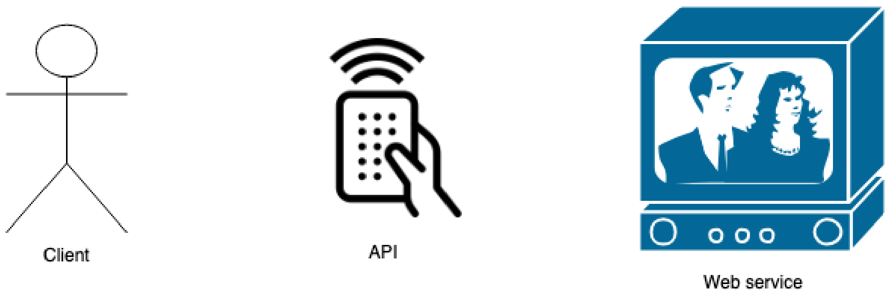


Figure 6: API in real life

A RESTful API is an API that follows some directives defined by Roy Fielding, first it must be based on HTTP. The client makes a HTTP request and receives an answer in HTTP as well. These requests aim to POST, GET, PUT and DELETE data (others requests are possible but these four are the most used, called as well CRUD for Create, Read, Update and Delete).

Then there are six architectural constraints: statelessness (nothing about the client state is stored on the server between requests), client-server (it has to be an interaction between a client and a server and the client request must have enough information so the server can work by itself), cacheable (some defined requests could be stored in cache to answer quickly to a same request), uniform interface (one way to access resource, it includes metadata in the answer, description of the information, it shows the next possible actions with this data), layered system (the web service is divided in layers that work together to build a hierarchy) and code on demand (optional, allows the code to be transmitted via the API). (What is REST, 2019a) (YoruNoHikage, 2016)

2.3.2 HAPI

Node JS is a very low-level environment. Therefore, in almost all cases, it is used with a framework like HAPI or Express. This chapter will be a presentation of the HAPI framework. HAPI is a solid framework used mainly to build backend APIs but it is possible to create server and HTTP proxy applications as well. (Haider, 2018) It allows to make scalable APIs and it is a good framework to have a RESTful API as it gives you good tools to make routing, to keep answers in cache and input/output validations. (Haider, 2018) It can work well with a database and a frontend framework. Hapi.js offers a really good code readability, it is easy to understand what is going on and thus makes it easier to debug (Hammer, 2018)

2.3.3 JOI

When a web service is working with data that can come from users, it is widely recommended to have a data validation feature. It is essential because each user can make mistakes or try to add wrong information when they want to manipulate the data located in the database. It will prevent having wrong information, for instance an age in the email column. If the database is well configured, it will make an error when it receives wrong data but with SQL errors it is often complicated to understand what really happened.

To fulfil this need, the JOI package is perfect. It is the most used JavaScript object validation package for Node. With this one, we will be able to accept only accurate data. JOI allows to create schema of how an object must look like. (Bulat, 2018) For example, it is possible to say the type or the length of the object. It also allows to specify if a field is required and a lot more. Once set up, the error begins to be clearer and the transaction will be stopped before bothering the database for nothing.

2.3.4 Knex

When the web service is interacting with a database, it can be interesting to have a query builder to make this work easier. Knex is a query builder for Node.js to interact with MySQL, MariaDB, Postgres, MSSQL, SQLite3, Oracle, and Amazon Redshift databases. (Knex.js, 2019a) It enables to create queries without using SQL language. Thereby, the SQL skill is less important in a project that uses this package and so the learning curve can go higher.

Instead of SQL, the developers build queries with JavaScript language, this language also allows them to use callbacks and promises. (Knex.js, 2019a) With this point, Knex can be really interesting in a Node project but it is important to take into consideration that it gives a little loss of performance as the JavaScript query has to be translated in SQL language. (Kuizinas, 2019) A priority has to be made: the performance or the gain of readability. "*When readability and performance are in conflict, I will always choose readability. Machines keep getting faster and cheaper. Humans only get slower and more expensive.*" (Hammer, 2018).

3 Design

Before starting to develop the application, the design must be well thought through. This is to be sure to advance in the right direction and it is easier to change something in the design if it is done before the coding part. The tools used will be similar to the tools I used for the Multidisciplinary Software Project, it will strengthen the skills I got from this lesson and the application has quite the same needs that the application we created for the project. The three main parts will be mentioned below: database, backend and frontend.

3.1 Database

The database will be built with MariaDB and will serve to gather all data from the web application. The Figure 7 shows how the database will be structured. There are all the tables, links and columns.

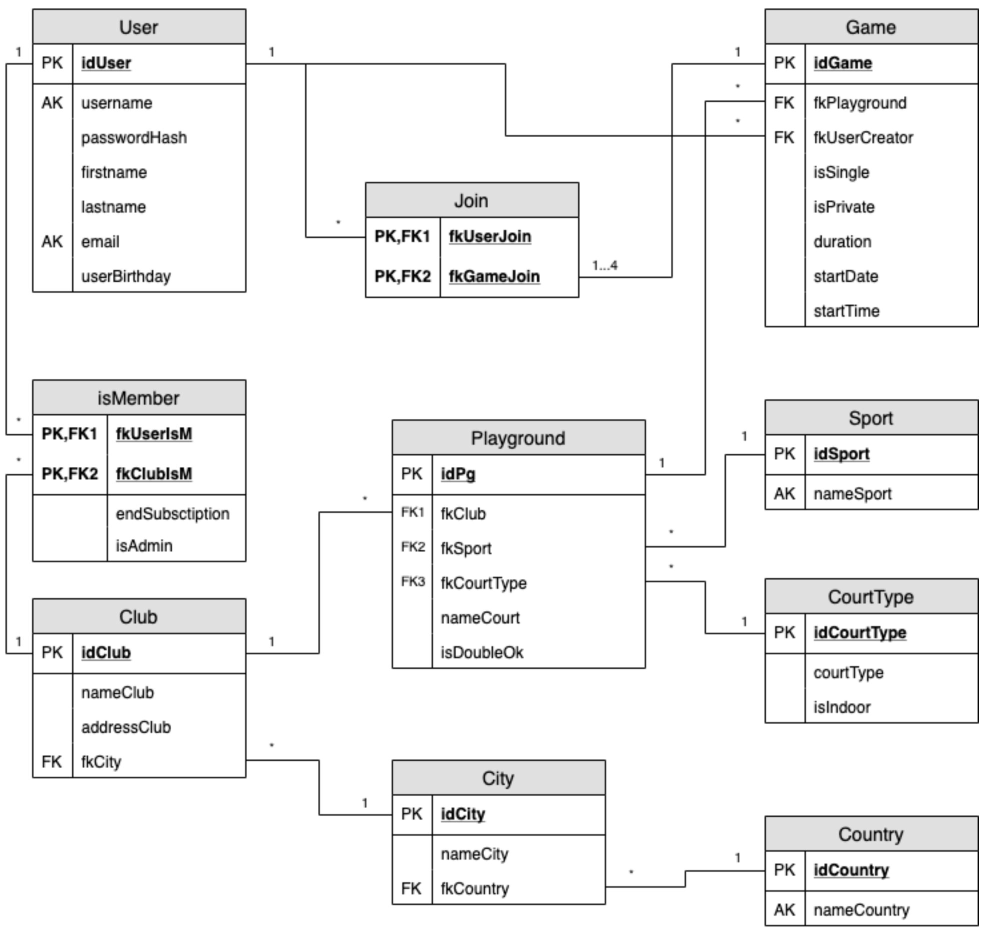


Figure 7: Database's model

3.2 Backend

The backend will be an API that will be used to interact with the database. This API will be coded in Node.js with the help of the HAPI framework. Node will allow to have a quick answer to the frontend request and HAPI has been chosen because of its gain of readability. HAPI offers a lot of features that can be interesting for the project and that are not included into some other frameworks like Express.js for example. The package Knex will be used for the query builder, so no SQL will be directly type in the backend.

3.3 Frontend

3.4 Tools

The web application will be developed with React. The solution that will be created will used a lot of components and this library fits perfectly as it is component based. The tool “create-react-app” will be used to initialize the project. Redux will manage all global states of this application and the navigation will be managed with “react-router-dom”. Finally, the graphic design will be made with “material-ui” in order for the application to have a nice look.

3.4.1 View mock-ups and navigation

Following this, there are all the mock-up of this application. The first one is the login page where it will be possible to authenticate the user. The user needs to be connected to see the other pages. No registration is possible as only the members that have subscription in the sport club are allowed to use this application.

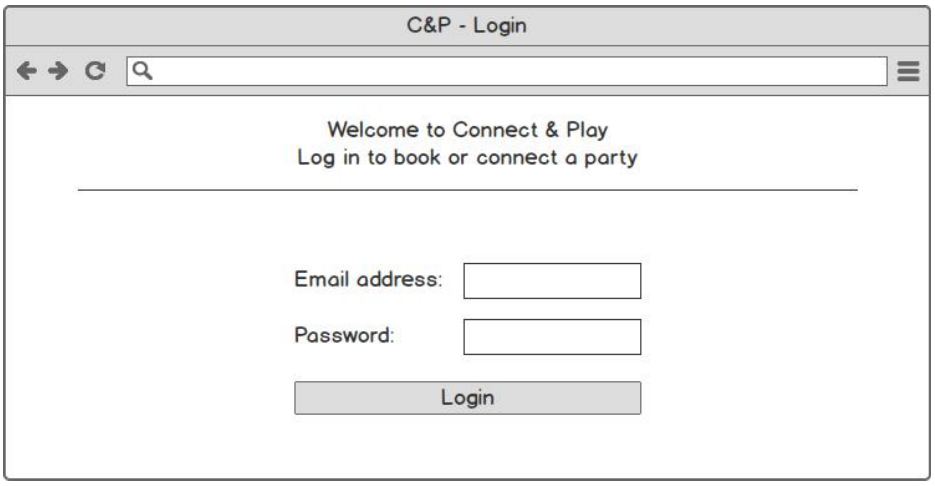


Figure 8: Login view

The second page is the home page. After the login, it is the first page that appears. It shows all the options the user can do and he can use the menu buttons to navigate or the two buttons in the centre of the page.

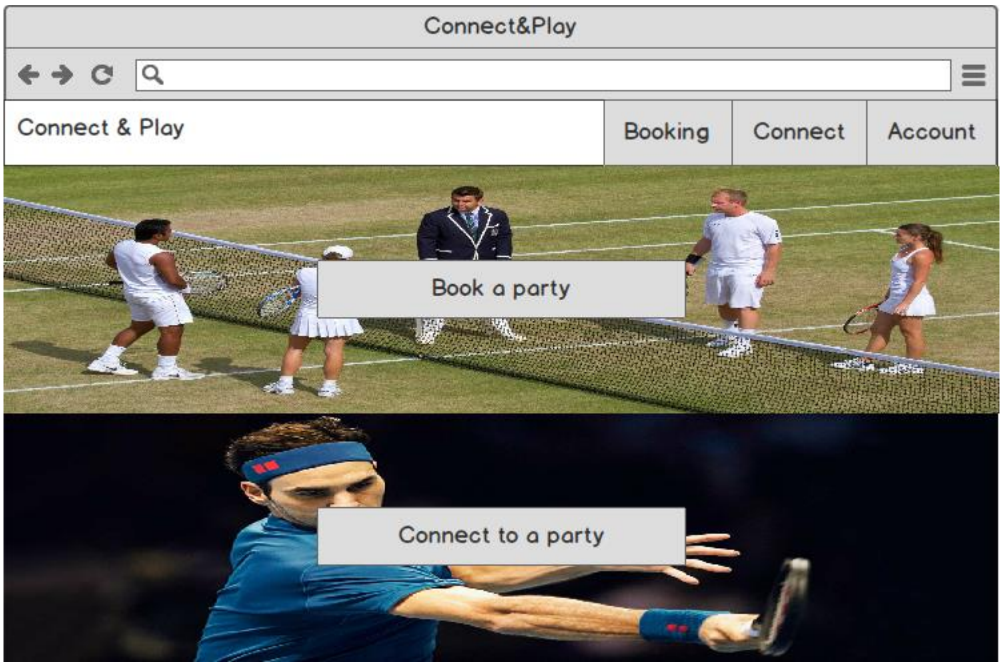


Figure 9: Home view

The booking page appears when the user clicks on the “Booking” button. It allows the user to choose a date and a sport. Then the timetable of all courts will be displayed with their respective availability. It is possible to click on any available time to open a reservation form.

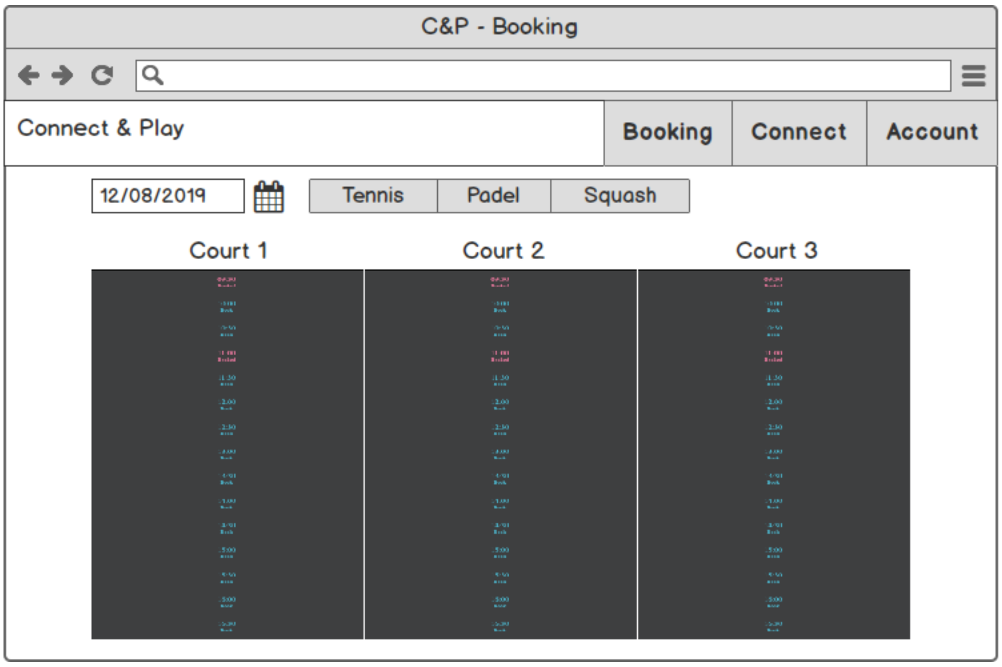
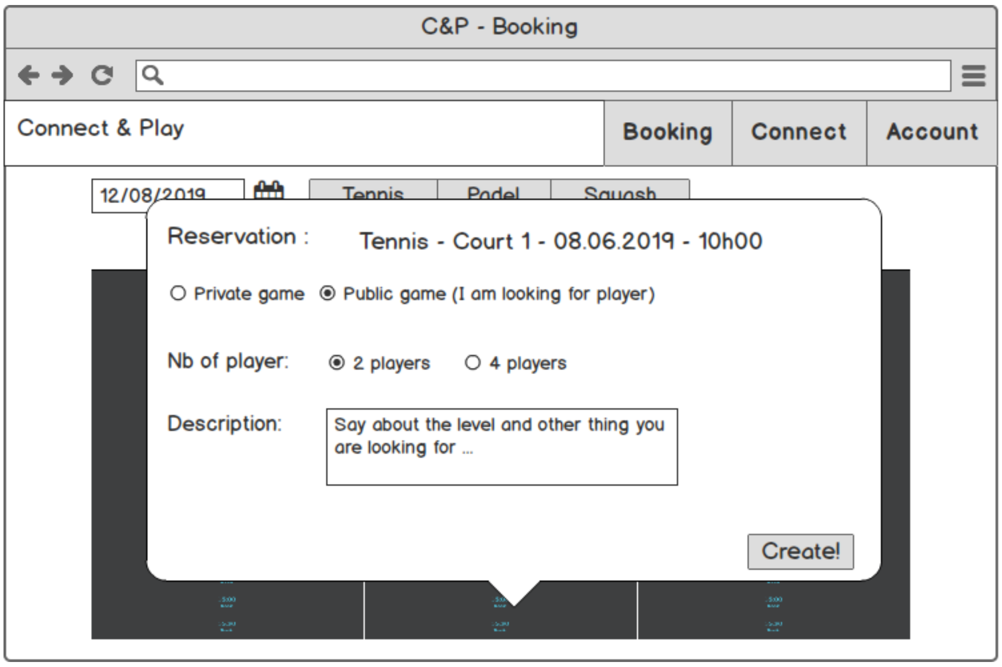


Figure 10: Booking view

If a click on an available time happens, then a pop-up appears to enter every information to the booking. It is possible to choose if you want a private or a public party. If it is a private one, you need to specify each player, and for a public one, it is required to specify the number of player and to give a description.



Reservation : Tennis - Court 1 - 08.06.2019 - 10h00

Private game [✓ Public game (I am looking for player)]

Nb of player: [✓ 2 players] 4 players

Description: Say about the level and other thing you are looking for ...

Figure 11: Booking form view

The second page is accessible with the “Connect” button. Thanks to that page, it is possible to choose a time and a sport. Then, all the games that are looking for players are displayed. It is possible to join a game by clicking the “Join this game!” button.

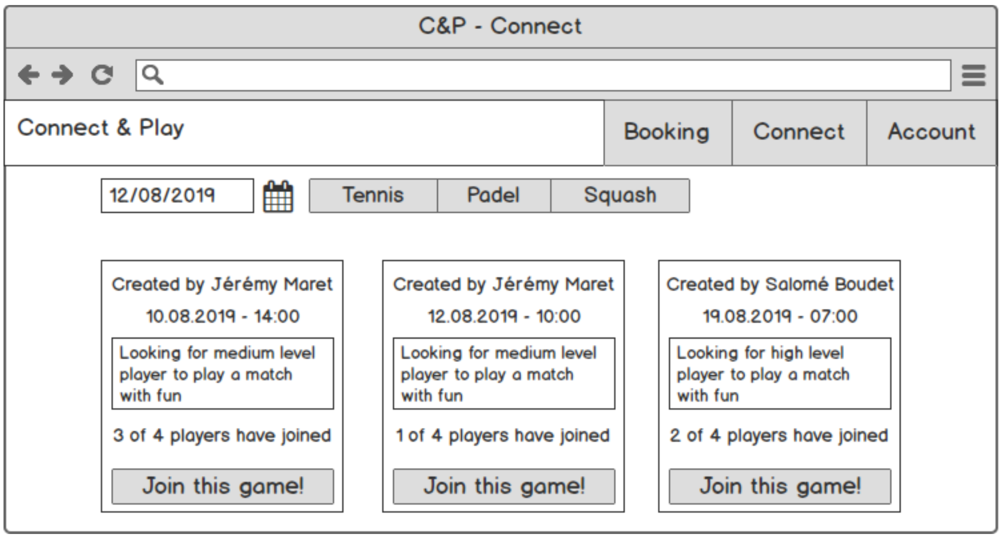
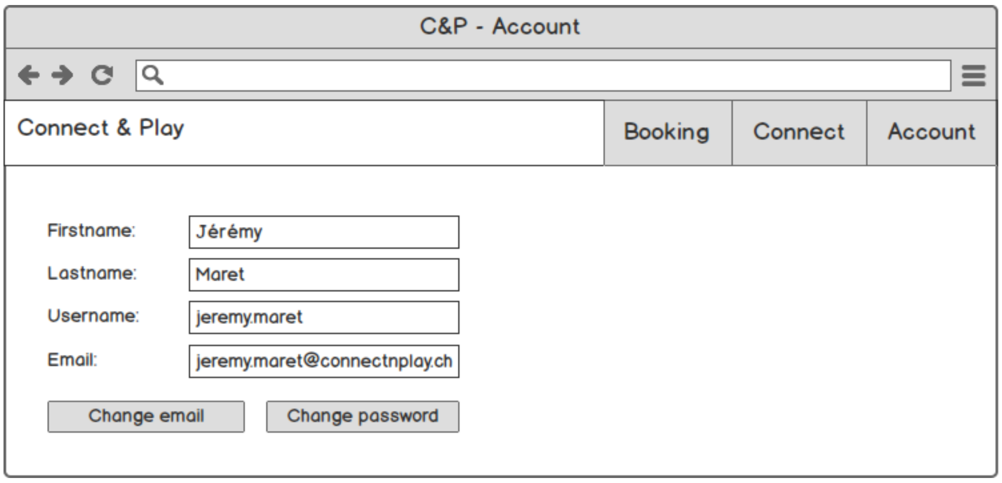

C&P - Connect
Connect & Play
Booking
Connect
Account
12/08/2019
Tennis
Padel
Squash
Created by Jérémy Maret
10.08.2019 - 14:00
Looking for medium level player to play a match with fun
3 of 4 players have joined
Join this game!
Created by Jérémy Maret
12.08.2019 - 10:00
Looking for medium level player to play a match with fun
1 of 4 players have joined
Join this game!
Created by Salomé Boudet
19.08.2019 - 07:00
Looking for high level player to play a match with fun
2 of 4 players have joined
Join this game!

Figure 12: Connect view

Finally, the account page that appears when you click on the “Account” button shows all the information about the connected user. It is possible to change the email address and the password by using to two buttons (a pop-up will appear).



Firstname:	Jérémy
Lastname:	Maret
Username:	jeremy.maret
Email:	jeremy.maret@connectnplay.ch

Figure 13: Account view

4 Implementation

This chapter puts forward the creation of the application. It describes how the whole application is structured and then how each part (database, backend and frontend) has been developed. It also explains the reason of each choice made during the development of this application. The developed application is stored on a public repository in the website GitHub.com, it is possible to access it by following this link: https://github.com/Adorax/CoPlay

4.1 Structure

Before initializing the project, the structure had to be chosen. There are three different parts that work on their own, but they are all part of the same project. The first part is the database, the second one is the backend, and the last one is the frontend. As each part does not use the files of the others, they can be completely separated but I chose to regroup them in one subfolder. It gave me the possibility to have only one git repository, which can be seen as a downside, but as I am working alone on this project, it is easier for me to have everything in one place. The Figure 14 shows how I structured every files of the project.

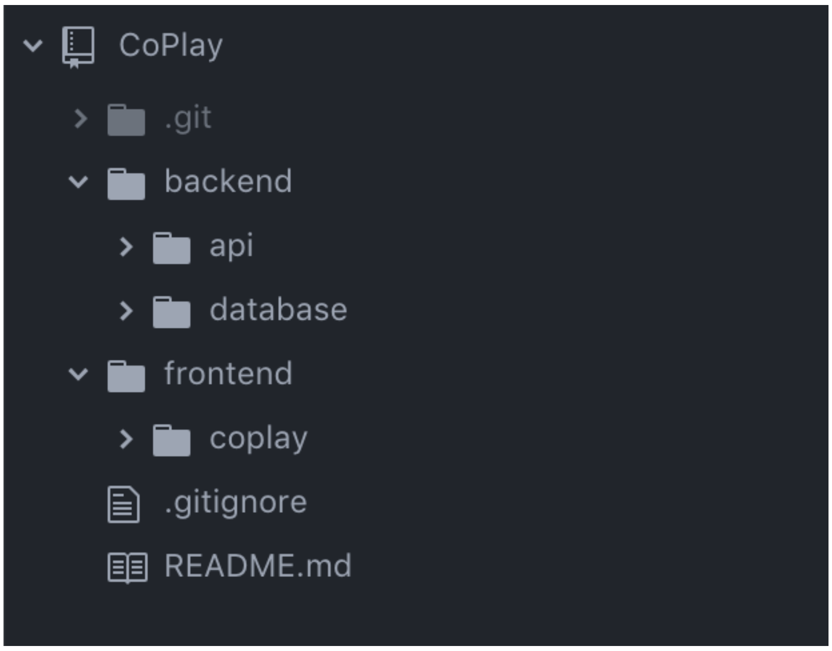
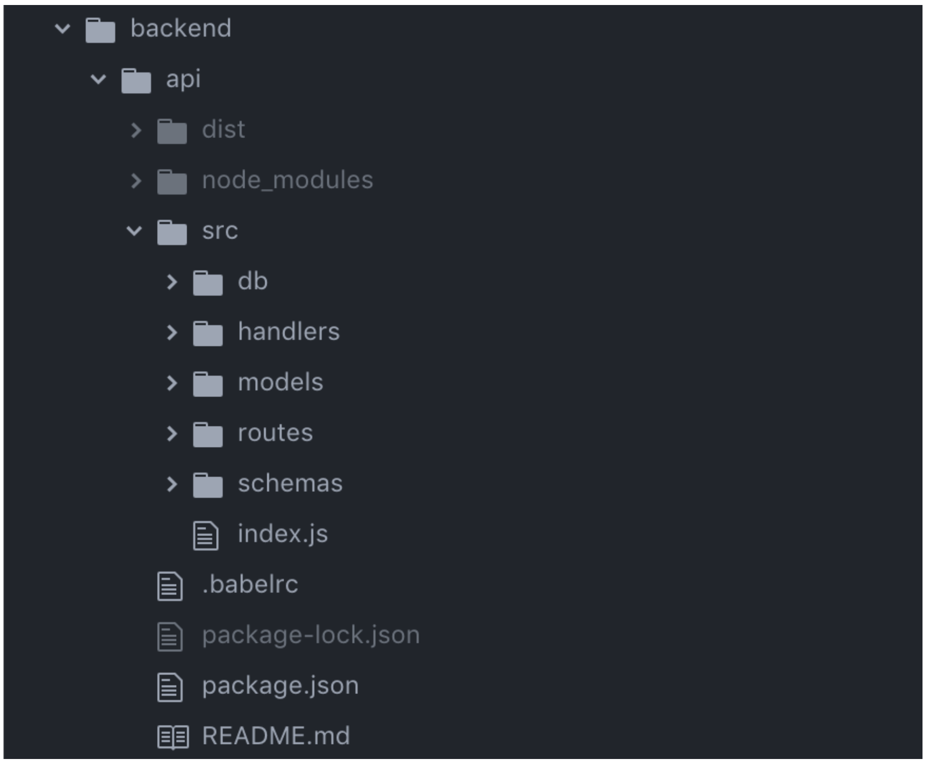


Figure 14: Project structure

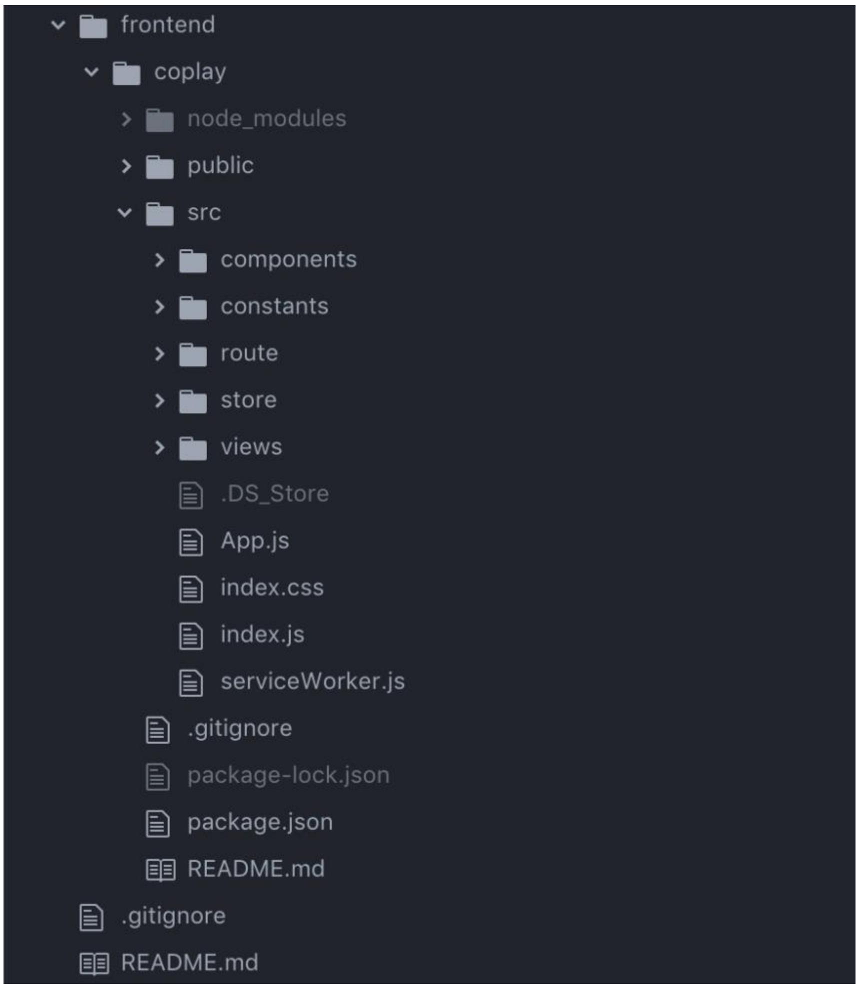
The root folder contains the README.md and two other folders: « backend » and « frontend ». The backend’s folder has two folders: « api » that encloses the Node.js API and « database » that has every SQL script to set up the database. The database and api are both located in this file as they are part of the backend, and the SQL scripts can be used to initialize the database directly into the API to make it faster. In the frontend's folder, the « coplay » React web application is stocked. With this global structure, I can easily switch between the three parts, because it happens quite often that a change has to be made in the API or in the database when you are working on the frontend. When a new feature is implemented, it can affect two or three parts. Therefore, with this disposition I can manage to work on the three parts at the same time.



```
backend
  api
    dist
    node_modules
    src
      db
      handlers
      models
      routes
      schemas
      index.js
    .babelrc
    package-lock.json
    package.json
    README.md
```

Figure 15: Backend structure

The structure of the API is shown in the Figure 15, a folder "src" is created that contains every API file. Into this folder, there is a file index.js that will serve to configurate, start the server and link the rest of the files. These files will be split into different folder: "db" contains the files to link the database, "route" has every available route, "handlers" involves the handlers used by the routes, "models" encloses the files that make requests to the database and finally "schemas" contains all the schemas used to control the request and answer.



```
frontend
  coplay
    node_modules
    public
    src
      components
      constants
      route
      store
      views
      .DS_Store
      App.js
      index.css
      index.js
      serviceWorker.js
    .gitignore
    package-lock.json
    package.json
    README.md
  .gitignore
  README.md
```

Figure 16: Frontend structure

The last structure that has to be built up is the frontend one and Figure 16 shows how it is. As I will explain in the next chapter, the frontend will be generated so that the structure must be changed to look like that after the generation. The structure I created is in the “src” folder: “route” keeps the files for the navigation of the app, “views” stores each view called by the navigation, all the components of the app are located in the “components” folder, everything about Redux is enclosed into “store” and the constants of the app are into the folder “constants”.

4.2 Setting up the environment

When the structure is defined, the environment can be initialized. In the root folder of my project, the README.md (Maret, 2019) now explains how to set everything up to run the application. But when the project began, each part had to be installed from scratch. I installed first the MariaDB server in my local computer, a database for this project also had to be created and then I created a user with the right permission into this database that is used by the API to interact with it. The API needs the right to SELECT, INSERT, UPDATE and DELETE. After this step, I initialized a git repository to get a version control of this project I did it in the root path as decided in the structure. I directly created a ".gitignore" and I put everything that did not need to be save in Git. Finally, I pushed the repository on http://www.github.com. It insured me a backup of the project as every change will be pushed in this website.

After, I set up the API by installing Node.js in my local machine and created the folder for the API, I ran the command "npm init" to create the "package.json" file. Into these files, I wrote scripts (initializing the database and start the server), I installed babel (JavaScript compiler) and Nodemon (to reload the server when a file undergoes a change.) I installed "Hapi.js" and "Knex" as decided in the design chapter. I also chose to install "hapi-swagger" to automatically generate a documentation (Hapi Swagger, 2019a). Finally, I wrote a file to reset automatically the database, for making this, I needed something to read SQL script file, I chose to use "fs" (Szabo, 2019a).

The last step was to set up the frontend. I used "create-react-app" to generate the React application which included tools so I could directly start writing code. After this step, I made the folder structure as shown in the last chapter. I finished the setting up part by adding "Material-ui" to this application (Getting started, 2019a). The links for the Roboto font and the font icons were written into the file "index.html".

4.3 Database

Following the design made previously, three scripts have been written to handle this database. These scripts allow to reset completely the database, thus, it can be possible to run them directly from the API. The first script deletes each table if they exist, the second one creates each table with all their rows and links with the other tables. Finally, the last one allows one to insert some data to be able to use the application. The inserted data will help when the frontend will be created

4.4 Backend

The first thing to do was to write the "index.js" file at the root of the "src" folder to configure and start the server. This file also permits to connect each route by connecting the index file of the route folder that concatenated every route. Then I created the route that the React application will or would use, so I did the route for each need. When a route is created, there are multiple objects to specify: the method (GET, POST, PUT, DELETE), the path, the handler (I specify every time a function stocked in the handler folder) and options (I use this to validate the parameters or payload and to write the possible response for the documentation.)The handlers are called by the created routes, I used them to call the model and then return the response they get. If the model gets a successful answer they will return the data otherwise they return an error depending of what really happened.

The SQL queries is built with knex into the model files, I configured knex in the file “knex” from the “db” folder. Finally, I am using the Schema files to validate the data received from the payload, parameters and database with Joi. All data are checked following what I defined in the schemas. The documentation uses the Joi validation that I wrote on the schema to write what each request needs and what they give back.

Some parts of the backend were tricky. For instance, the insertion on the table “join” should not have more than one row for a game that what the game expects. If an open game is created for a simple (1 player versus another player), I checked that if an insert is done there is still available space. If there is no space an error is returned. Another tricky part was for the update of a password. When this action happens, the old password mentioned by the user should be compared with the hash stocked in the database, this comparison is made with the function compare of bcrypt. If the old password is right, it returns a successful answer and then the frontend know that it can make the update correctly.

4.5 Frontend

I started the web application by setting up Redux, I created the store in the “index.js” file and put the App component into this store. After this I wrote the Navigation that is encapsulated into the App, I chose to create two different routes: the basic one given by “react-router-dom” and a private one that checks if the user is logged to show the page. If no one is logged, every private route is redirected to the login route. I set it up that way because the app must be usable only if the user is connected. These routes link to the views that encapsulated components.

As I was using Redux to manage the application states, I needed to create the actions and reducers. The actions are used to fetch data from the API previously created and then they dispatch the data to the reducers that will take care of updating the states with what they have. When a state changes in a reducer, I took care of creating a new object that is an exact clone of the last state, and I did the modification into this new state. The method allows the reducer to return to its state in this way: “return newState || state;”. Thereby, with this solution, the old state is returned if a problem occurs during the declaration of the variable.

The SignIn components were inspired with the material-ui SignIn example (SignIn example, 2019a). It calls the API with the receive credential to compare the username and password. When the sign is successful, the rest of the application becomes available and a redirection to the Booking view happens. The Account component shows the information about the authenticated user. I created two dialogs to change the email address and the password.

The Booking components indicates the court’s timetables, depending which sport and data have been chosen. It is possible to get information if a click happens on an event or to open the dialog to create an event if an available time undergoes a click. This dialog allows to configure the game. If the game is private, it will be needed to write the username of each player. To create a game, I first requested the API to create the game and after it returned me the game id. Then, this identifier is used to make the request for creating the link with this game and each player. The timetable was hard to make, I wanted to display each court in the calendar then each one is a resource for the calendar. I also wanted that only a time slot from 7 am to 10 pm could be visible. To make this calendar, I used “react-big-calendar” (react-big-calendar, 2019a) and I formatted it in accordance with my special needs.

Finally, the Connect components allow to choose the sport and date and then displays every open party that fits the arguments. These games are shown into cards and contain information about them. A button can be used to add the connected user to the game. When someone is trying to connect to a game, a request to create a link is done to the API. Depending of the answer, a snackbar appears with an appropriate message.

5 Evaluation

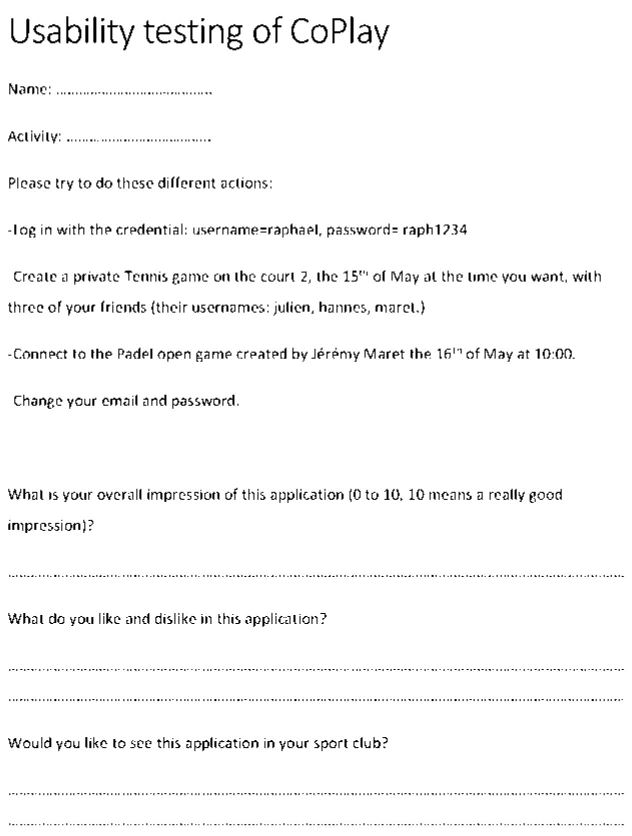
This chapter is about the evaluation of this thesis. It contains a usability testing, where it can be noticed how users are able to use this application. Then the result is confronted with what was expected at the beginning of the thesis.

5.1 Usability testing

Usability testing is the action to watch users using an application in live to rate its usability and ease of use. It is one of the best ways to see if a web application is well developed as you can see directly how the users interact with every element of the website. You can obtain a lot of information about how usable an application is (Babich, 2017). To evaluate if my application was user friendly, I chose to make a usability testing with four people. This chapter describes how it was made and the result.

5.1.1 How the test was done?

Beforehand, I chose four people and asked them if they were available to test my application. I selected three people that have already used a sport club booking system and one that had never used one. I gave them the sheet that you can see in the Figure 17, with instructions to do the four actions. When they tried to do these actions, I stayed next to them without giving any help to watch how they did. After they used my application, I asked them three questions to evaluate their impressions.



Usability testing of CoPlay

Name: ……………………………………

Activity: ………………………………

Please try to do these different actions:

-log in with the credential: username=raphael, password= raph1234

Create a private Tennis game on the court 2, the 15th of May at the time you want, with three of your friends (their usernames: julien, hannes, maret.)

-Connect to the Padel open game created by Jérémy Maret the 16th of May at 10:00.

Change your email and password.

What is your overall impression of this application (0 to 10, 10 means a really good impression)?

………

What do you like and dislike in this application?

………

………

Would you like to see this application in your sport club?

………

………

Figure 17: Usability testing instructions

They had to first log in with the given credential into the application. Then, they had to create a private party on a specific date to see if they were able to choose the sport correctly and navigate between dates. The third instruction was to find the open party I had created before and join it. Finally, I asked them to change their email and password. I could then see if they were able to cope with the errors if some happened and if they were able to understand how everything worked.

5.1.2 Results

The first test was done by Raphaël Azzoug who is used to sports clubs booking system. He was able to log in correctly and he directly understood how the booking page worked, so he succeeded this second action easily. For the connect action, he chose the right sport and the right date and time, but he stayed it in the Booking page. He clicked on the wanted event and did not understand why he could not join the game. I had to explain to him and after this test I changed the message, so that it will be possible to understand how to join an open game display on the “Booking” page. When I told him to go on the Connect page, he connected to the game without any problem. Finally, he intuitively changed his email and password. After this test, he gave the grade of eight and said that he liked the design and simplicity but did not really liked the information message he got from the booking page. He ended up by stating that he finds interesting to have this kind of application in a sports club.

The second test was done by Kevin Berret who is not used to sport clubs booking systems. He was able to log in and book the game perfectly. For connecting to the game, he also stayed in the booking page, but when he clicked on the game, he understood with the new message what he needed to do so, he succeeded to join the game and to change his information. After this test, he gave the grade of nine and said that he liked the user interface and the rapidity of the application. He also liked the message I had implemented from the last test and he added that he found that the date picker did not really fit with the rest of the design. He concluded by telling me that even if he does not go to sports clubs, he thought that it was a good idea that could work well.

The third test was done by Clément Vasser who is used to sport clubs booking systems. He was able to log in but took some time to book the game as he did not directly understand how to navigate. He succeeded to join the game easily and to change his information. After this test, he gave the grade of eight and said that he finds the application easy to use and to handle but he needed some time to understand how it works. For the last question, he answered by telling me that he would like to have this application in his club.

The fourth and last test was done by Julien Dubuis who is used to sport clubs booking systems. He was able to do the four steps without problem. For the questions, he gave the grade of ten and said that the application is really intuitive and easy to use and did not find anything he did not like. He then said that he would really appreciate to have an application like this one in his club as he currently struggles to find partners.

With these tests, I saw that the application is quite easy to use as they all succeeded to do the requested actions. They often tried first to connect to a game directly on the booking page, I should then find a way to make this clearer. Another solution could be to create a link directly when an open party is clicked on the booking page. Overall, they had a really good impression of the application and they all got really interested by the possibility to connect. When I saw them using my application, it helped me to detect bugs and what could be add to the application.

5.2 Results against goals

The created application follows quite well the goals expected at the beginning of this thesis. A booking system that allows to see the current reservations and to create new ones has been built. The possibility of joining a game has been correctly implemented as well. It is possible to see all open parties and booked a desired one. Finally, the application is more or less easy to use as the usability test has shown.

5.3 Method

The methods and tools that I used to create this application have been well chosen as it allowed me to successfully accomplish what I wanted. React gave me the possibility to easily do the frontend. However, I think I could have had a similar result if I had used Vue.js. Node.js gave me rapidity for the application and it will be really beneficial when the application will reach a production version. Spring or Laravel could have been an alternative but Node fits better with this king of project.

5.4 Validity of results

The application works correctly and can be used to make reservations and connections as it aimed to do.

5.5 Own learning

During this thesis, I had to learn various things as I lacked some skills needed to make this application. I had never developed an application of this magnitude alone, therefore I had to learn how to manage my work with the time I had. I finally understood Redux more deeply because I had not completely understood how it worked before starting this thesis. I learned a lot more about Node.js, React and all the other tools I used.

6 Conclusion

This thesis was intended to create a first version of an application with which it is possible to book a game to search for players and to join a game. A web application has been created that incorporates the objectives of the thesis well. This application has been tested with a usability test and it revealed that the application was quite easy to use and to understand. The usability test made it easier to understand how users understand the application. Some points were very clear for me but with the usability test, I quickly understood that it can be puzzling when you have not developed the application. This first version brings a solid base to make a productive version in the future. This solution aims at helping people to practice sport by bringing it more attractive. With this creation, it would be possible to make new meetings and there is no longer the need to find a player to practice as the application can find someone for you.

6.1 Personal conclusion

As I have had this idea since a little over a year, it was really good for me to finally create a first version of it. This opportunity gave me the possibility to better know if this idea was feasible or not. Now, I have every key to go further with this project and it will be easier to find people to cooperate with. I am thinking even more about creating a start-up now. This thesis gave me more confidence in my future carrier as I really pushed my skills further. When I started with this idea, I thought I did not have the capacity to create something like this, and now I know I can go way further than I originally believed. It was really interesting for me to do this thesis, a real pleasure.

6.2 Further development

This first version let me know if the application can work and can be interesting for a real use. This application will need to undergo several improvements and new features will have to be implemented to be able to set it up in a sport club. As I am really involved in this project, I want to go further and make a productive version. I am already thinking of further developments that will have to be done for the next versions of this project.

First, the security has to be thought. The API must be secure when it will be accessible on internet. Tokens could be used to identify the user of the API. The React will need to be secure as well. The password policies have to be stronger, captcha could be added to the login page to avoid brute force attack and some securities for DDOS attacks could be implemented. A library to handle some automated test of the full application should be considerate in the future. The error will have to be manage in a better way, by given better information to the user.

An admin part will have to be constructed to manage users, clubs and playgrounds. There, it will be possible to specify the memberships of users and control if they are still in a valid subscription. A payment method could be interesting and so everything about the membership could be made into the application. After this last point, a registration form could be interesting and the possibility to pay for only a few reservations instead of a membership.

It can be interesting to get information about the players that are in a game, so you will know more about your potential opponents. Following this last idea, the other players could be added into a friend list. With this it could be possible to make more options when an open game is created, for example allowed only the people of your friends list to join. A historiy of the previous games and a view of your next games with the opportunity to leave them have to be created. Finally, a possibility to switch the application in different languages could be done.

References

Atlassian. (2019a). Retrieved 2019, from Atlassian: https://www.atlassian.com/git/tutorials/

Babich, N. (2017, 02 23). *Adobe blog.* Retrieved from The Top 5 User Testing Methods: https://theblog.adobe.com/the-top-5-user-testing-methods/

Bhabesh. (n.d.). *Git Tutorial (Beginner): Using GitLab & Source Tree.*

Bulat, R. (2018, 10 27). Retrieved from https://medium.com/@rossbulat/joi-for-node-exploring-javascript-object-schema-validation-50dd4b8e1b0f

Burnett, M. (2017, 10 02). Retrieved from https://www.sitepoint.com/an-introduction-to-jsx/

Charruel, M. (2019a). Retrieved 2019, from https://openclassrooms.com/fr/courses/4902061-developpez-une-application-mobile-react-native/5046311-decouvrez-redux

create-react-app. (2019a). Retrieved 2019, from Github Facebook: https://facebook.github.io/create-react-app/

Cromwell, V. (2016, 11 3). *Evan You*. Retrieved from web.archive.org: https://web.archive.org/web/20170603052649/https:/betweenthewires.org/2016/11/03/evan-you/

Daityari, S. (2019, April 27). Retrieved from https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#part-2-community-and-development

Daityari, S. (2019, April 27). Retrieved from https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#part-2-community-and-development

Daityari, S. (2019, April 27). *angular-vs-vue-vs-react.* Retrieved from Codeinwp: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

Dionne, M. (2017, NOVEMBER 23). Retrieved from https://snipcart.com/blog/javascript-nodejs-backend-development

Emmanuel, O. (2018, June 01). Retrieved from https://medium.freecodecamp.org/understanding-redux-the-worlds-easiest-guide-to-beginning-redux-c695f45546f6

Front-end Frameworks - Overview. (2018). Retrieved from Stateofjs: https://2018.stateofjs.com/front-end-frameworks/overview/

Getting started. (2019a). Retrieved 2019, from Material-ui: https://material-ui.com/getting-started/installation/

Git branch. (2019a). Retrieved 2019, from atlassian: https://atlassian.com/git/tutorials/using-branches

Git Merge. (2019a). Retrieved 2019, from atlassian: https://atlassian.com/git/tutorials/using-branches/git-merge

Git-scm. (2019a). *Git Branching - Branches in a Nutshell*. Retrieved 2019, from Git-scm: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

Haider, M. (2018, 01 05). Retrieved from https://simpleprogrammer.com/introduction-hapijs/

Hamedani, M. (2018, 11 5). *react-vs-angular*. Retrieved from programmingwithmosh.com: https://programmingwithmosh.com/react/react-vs-angular/

Hammer, E. (2018, Novembre 28). *Why You Should Consider hapi.* Retrieved from Hueniverse: https://hueniverse.com/why-you-should-consider-hapi-6163689bd7c2

Hapi Swagger. (2019a). Retrieved 2019, from Github: https://github.com/glennjones/hapi-swagger

JSX in depth. (2019a). Retrieved 2019, from Reactjs: https://reactjs.org/docs/jsx-in-depth.html

Knex.js. (2019a). Retrieved 2019, from Knexjs: https://knexjs.org/

Knex.js. (2019a). Retrieved 2019, from Github: https://github.com/tgriesser/knex

Kuizinas, G. (2019, 02 12). Retrieved from https://medium.com/@gajus/stop-using-knex-js-and-earn-30-bf410349856c

Kumar, S. (2015, May 25). *Difference Between Library and Framework*. Retrieved from c-sharpcorner: https://www.c-sharpcorner.com/UploadFile/a85b23/framework-vs-library/

Lerner, A. (2019a). Retrieved 2019, from https://www.fullstackreact.com/30-days-of-react/day-2/

Maret, J. (2019). *README.md.* Retrieved from Github.

Neagoie, A. (2018, 11 08). *React vs Angular vs Vue: Who wins in 2019?* Retrieved from Medium: https://medium.com/zerotomastery/react-vs-angular-vs-vue-who-wins-in-2019-5d9acd0843e8

Nebra, M. (2018, 01 23). Retrieved from https://openclassrooms.com/fr/courses/2504541-ultra-fast-applications-using-node-js/2504696-node-js-what-is-it-for-exactly

Nguyen, T.-Q. (2019a). Retrieved 2019, from https://openclassrooms.com/fr/courses/5511091-organisez-votre-application-react-avec-la-logique-redux

Pandit, N. (2018, November 14). *What Is ReactJS and Why Should We Use It?* Retrieved from c-sharpcorner: https://www.c-sharpcorner.com/article/what-and-why-reactjs/

Petrosyan, M. (2018, 02 07). Retrieved from https://itnext.io/angular-5-vs-react-vs-vue-6b976a3f9172

Private route. (2019a). Retrieved 2019, from reacttraining.com: https://reacttraining.com/react-router/web/example/auth-workflow

React - README.md. (2019a). Retrieved 2019, from Github: https://github.com/facebook/react

React Native. (2019a). Retrieved 2019, from Wikipedia: https://en.wikipedia.org/wiki/React_Native

react-big-calendar. (2019a). Retrieved 2019, from Github: https://github.com/intljusticemission/react-big-calendar

react-router. (2019a). Retrieved 2019, from reacttraining.com: https://reacttraining.com/react-router/

Redux. (2019a). Retrieved 2019, from Redux: https://redux.js.org/

Reese, E. (2019a). Retrieved 2019, from Openclassroom: https://openclassrooms.com/fr/courses/3432056-build-your-web-projects-with-rest-apis/3432063-what-is-an-api

SignIn example. (2019a). Retrieved 2019, from GitHub: https://github.com/mui-org/material-ui/blob/master/docs/src/pages/getting-started/page-layout-examples/sign-in/SignIn.js

Sridhar, A. (2018, August 12). *An introduction to Git: what it is, and how to use it*. Retrieved from Freecodecamp - Medium: https://medium.freecodecamp.org/-what-is-git-and-how-to-use-itc341b049ae61

Suzdalnitski, I. (2018, September 04). *React.js: a better introduction to the most powerful UI library ever created.* Retrieved from hackernoon.com: https://hackernoon.com/react-js-a-better-introduction-to-the-most-powerful-ui-library-ever-created-ecd96e8f4621

Szabo, G. (2019a). *Reading a file with nodejs.* Retrieved 2019, from Code maven: https://code-maven.com/reading-a-file-with-nodejs

Torvalds, L. (2007). Tech Talk: Linus Torvalds on git. *Tech Talk: Linus Torvalds on git.* Google.

What is git ? (2019a). Retrieved 2019, from atlassian: https://www.atlassian.com/git/tutorials/what-is-git

What is REST. (2019a). Retrieved 2019, from restfulapi.net/: https://restfulapi.net/

YoruNoHikage. (2016, 04 07). Retrieved from https://zestedesavoir.com/tutoriels/299/la-theorie-rest-restful-et-hateoas/

You, E. (2016, November 3). Evan You. (V. Cromwell, Interviewer)

Yusufu, E. (2017, 12 23). Retrieved from https://medium.freecodecamp.org/beginners-guide-to-react-router-4-8959ceb3ad58