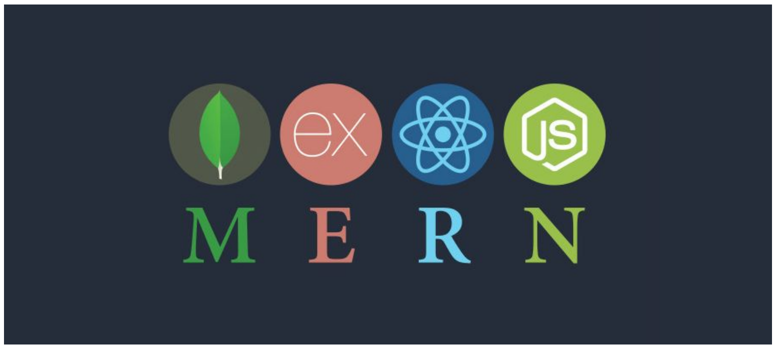
Assembler
School of Software Engineering

TFM

Documentation



By
Alfonso de La Guardia

Table of contents

Project Planning

The project is divided into five different phases, according to the standard division established by the Project Management Body Of Knowledge. They are, in sequential order:

1. **Project Initiation:** This phase relates to what the project is about, they hold meetings to distinguish the different elements into which the project can be divided, and team roles are assigned to each member of the team.
2. **Project Planning:** In this phase, we divide the project into tasks, they assign time estimates, priorities and interdependencies to each task, and gather the tools and resources needed to successfully fulfill the requirements of this project.
3. **Project Execution:** In this phase, all the tasks created in the previous section are executed until its completion.
4. **Project Validation:** parallel to the previous phase, in this phase we do all the tests necessary to validate the project.
5. **Project Closure:** In this phase, the project gets loaded into Heroku.

Tasks corresponding to each phase:

TASK	**Priority**	**Time/ hour**
[P1] Project Initiation		**8h**
[P1-1] Requirements analysis	Medium	1h
[P1-2] Wireframes design	Medium	2h
[P1-3] Uses cases analysis	High	1h
[P1-4] Define Database Design	High	4h

TASK	**Priority**	**Time/ hour**
[P2] Project Planning		**8h**
[P2-1] Project division into phases	Medium	0.5h
[P2-2] Task's delimitation of each phase	Medium	1h
[P2-4] Define the structure of the directories	Medium	0.5h
[P2-5] Define the gitflow	High	0.5h
[P2-6] Initial version of the documentation	High	3h

TASK	**Priority**	**Time/ hour**
[P3] Project Execution		**60h**
[P3-1] Initializing the Project		**3h**
[P3-1-2] Creating the client side on Reactjs	Medium	1 hr

[P3-1-3] Adding SASS to the project	Medium	30 min
[P3-1-4] Adding the Ant Design library to the project	Medium	30 min
[P3-1-5] Installing dependencies	High	30 min
[P3-1-6] Connecting the project to MongoDB	High	30 min
[P3-2] Creating Routes		**5h**
[P3-2-1] Installing React-router-dom	High	30 min
[P3-2-2] Creating basic pages to use with the routes	High	30 min
[P3-2-3] Creating layouts that will divide the user and admin section	Medium	1 hr
[P3-2-4] Configuring routes for the admin panel	Medium	30 min
[P3-2-5] Configuring the route systems for the pages inside Layout for admin	High	1 hr
[P3-2-6] Adding the route configuration for normal users	High	30 min

[P3-2-7] Programming the route system to load the views inside the Basic Layout	Medium	30 min
[P3-2-8] Adding Error404 view	Medium	30 min
[P3-3] Creating the Layout for Admins	Medium	**3h**
[P3-3-1] Creating color variables in SASS to use them throughout the project	Medium	30 min
[P3-3-2] Styling the LayoutAdmin	Medium	30 min
[P3-3-3] Creating the MenuSider component	Medium	30 min
[P3-3-4] Centering content and adding functionality to the MenuSider	High	30 min
[P3-4] Creating the register for new users	High	30 min
[P3-4-1] Creating the model, view and controller for users	High	30 min
[P3-4] Creating the endpoint to add new users		**6h**

[P3-4-1] Creating the basic structure of the SignIn page	Medium	1 hr
[P3-4-2] Creating the structure of the register form in the SignIn page	High	1 hr
[P3-4-3] Adding SASS to the register form	Medium	1 hr
[P3-4-4] Saving the state of the register form with UseState	High	1 hr
[P3-4-5] Creating reusable validation functions	High	30 min
[P3-4-6] Connecting with the Endpoint of Register	High	30 min
[P3-4-7] Resetting the form when the register is finished	Medium	30 min
[P3-4-8] Formatting register email with toLowerCase	Medium	30 min
[P3-5] Creating the User Login		**6h**
[P3-5-1] Creating the service for the creation of Tokens	High	1 hr
[P3-5-2] Creating the Endpoint to do Login	High	1 hr

[P3-5-3] Creating the structure of the login form	Medium	1 hr
[P3-5-4] Saving form data in a Component state	High	30 min
[P3-5-5] Saving the Tokens in localStorage and creating constants to do so	High	30 min
[P3-5-6] Creating functions to obtain access tokens and refresh token	Medium	30 min
[P3-5-7] Creating endpoint to refresh AccessToken	High	15 min
[P3-5-8] Creating function to logout a user	High	15 min
[P3-5-9] Creating a Hook to prove if user is Logged In	Medium	15 min
[P3-5-10] Writing the Auth Provider Login	High	15 min
[P3-5-11] Blocking the Login page for logged in users	Medium	15 min
[P3-5-12] Adding functionality to logout button	High	15 min

[P3-6] Creating Admin Panel for Users		**10h**
[P3-6-1] Creating the User Menu	High	30 min
[P3-6-2] Creating the endpoint to obtain all users	High	30 min
[P3-6-3] Creating middlewares to block URL to users who are not logged in	Medium	30 min
[P3-6-4] Function to execute Endpoint and obtain all the users	High	30 min
[P3-6-5] Creating a Component to show active and inactive users	High	30 min
[P3-6-6] Creating the form to edit user data	Medium	30 min
[P3-6-7] Updating user data	High	30 min
[P3-6] Creating function for adding new users	High	30 min
[P3-7] Creating Web Menu	Medium	30 min

[P3-7-1] Creating Menu Web section in the Admin Panel	High	30 min
[P3-7-2] Creating the structure of the Menu in	High	30 min
[P3-7-3] Endpoint for creating new menus	Medium	30 min
[P3-7-4] Creating Endpoint to obtain all the menus	High	30 min
[P3-8] Menu Web	High	30 min
[P3-8-1] Creating Menu Web section in the Admin Panel	High	30 min
[P3-8-2] Creating Endpoint to obtain all menus	Medium	30 min
[P3-8-3] Endpoint to activate or deactivate Menus	High	30 min
[P3-8-4] Creating logic for new menus	High	30 min
[P3-8-5] Logic to update the menu	High	30 min
[P3-8-6] Endpoint to be able to eliminate menus	Medium	30 min

[P3-9] Home Page		**3h**
[P3-9-1] Creating the main banner	High	1 hr
[P3-9-2] Component for showing top rated courses	High	1 hr
[P3-9-3] Component to show how courses work	High	30 min
[P3-9-4] Adding review section	High	30 min
[P3-10] Footer and NewsLetter		**3h**
[P3-10-1] Footer structure	Medium	1 hr
[P3-10-1] Navbar in the footer	High	30 min
[P3-10-1] Configuring backend for newsletter	High	30 min
[P3-10-1] Creating structure and form for the Newsletter	Medium	30 min
[P3-10-1] Connecting the form with the endpoint that registers E-mails	High	30 min

[P3-11] Courses Website		**6h**
[P3-11-1] Configuration to create endpoint of the users	High	30 min
[P3-11-2] Endpoint to create courses	High	30 min
[P3-11-3] Endpoint to delete courses	Medium	30 min
[P3-11-4] Endpoint to update users	High	30 min
[P3-11-5] Adding functionality to delete courses	High	30 min
[P3-11-6] Creating structure of the courses site	High	30 min
[P3-11-7] Creating presentation courses	Medium	30 min
[P3-11-8] Showing all courses on screen	High	30 min
[P3-12] Blog		**6h**
[P3-12-1] Creating blog structure	High	1 hr

[P3-12-2] Endpoint for new posts, updating posts and eliminate posts	High	1 hr
[P3-12-3] Adding pagination system	High	1 hr
[P3-12-4] Deleting posts	High	1 hr
[P3-12-5] Showing list of all posts	Medium	1 hr
[P3-13] Deploying the App		**1h**
[P3-13-1] Uploading database to MongoDB Atlas	High	30 min
[P-13-2] Uploading project to Heroku	Medium	30 min

TASK	**Time/ hour**
[P4] Project Validation	**4h**
[P4-1] Testing	4h

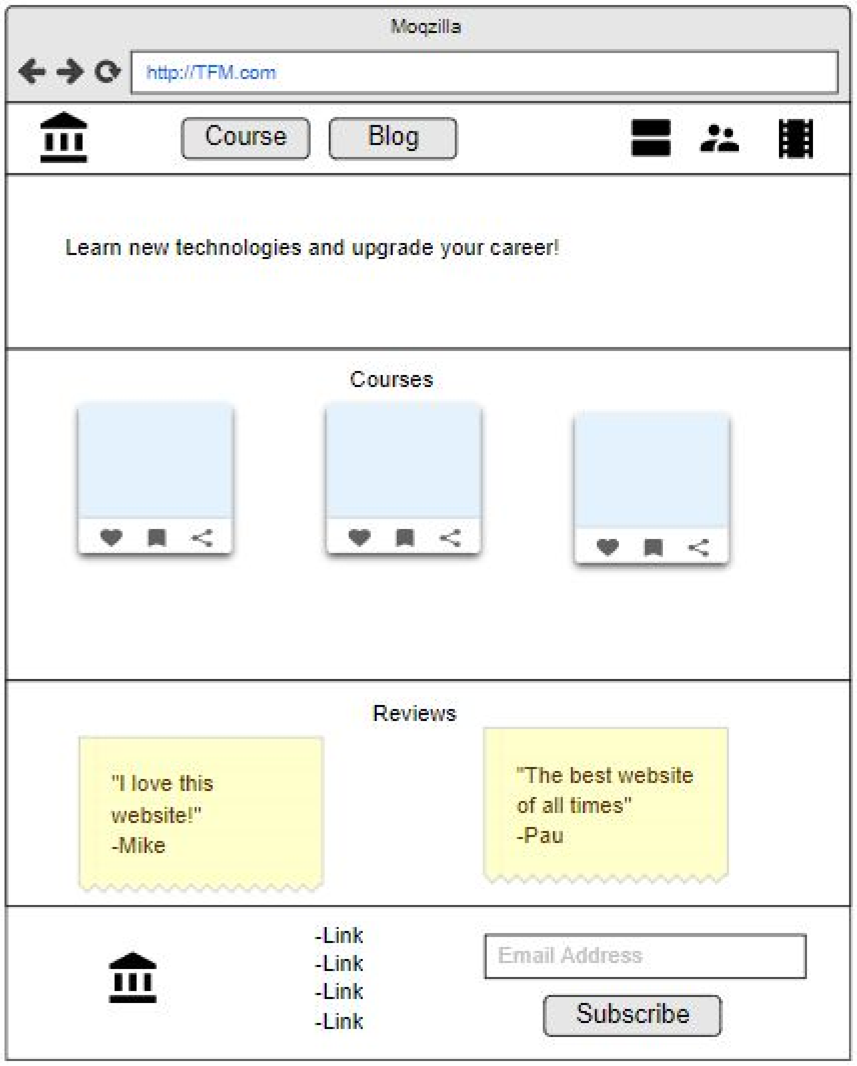
Time division

Project's total hours → 84 h

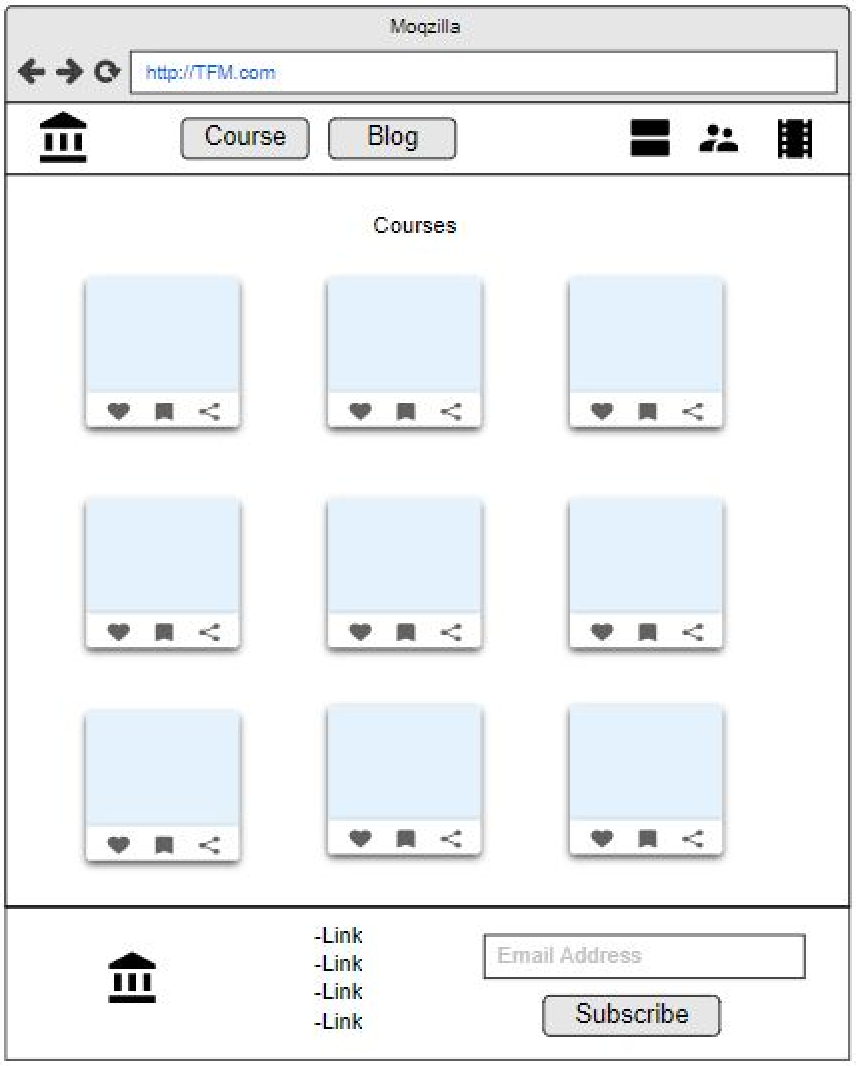
Wireframes

APP: Reactjs

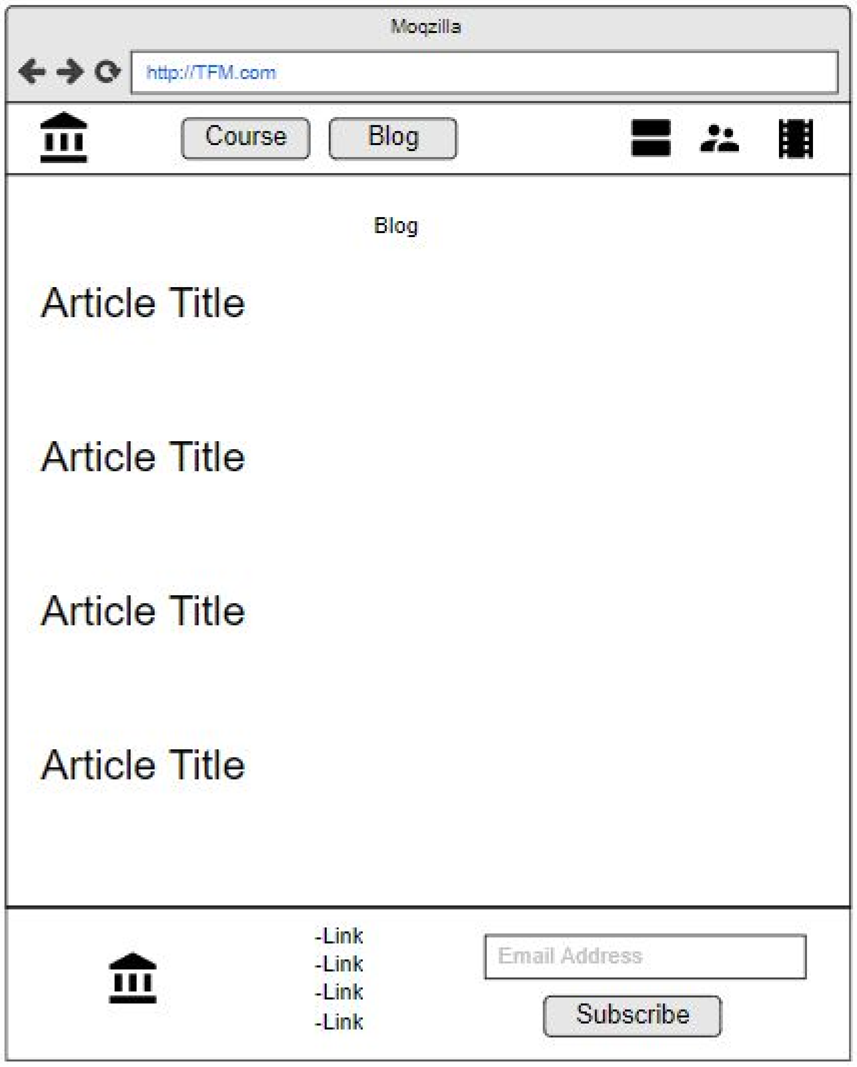
Dashboard


Moqzilla
http://TFM.com
Course
Blog
Learn new technologies and upgrade your career!
Courses
Reviews
"I love this website!" -Mike
"The best website of all times" -Pau
-Link
-Link
-Link
-Link
Email Address
Subscribe

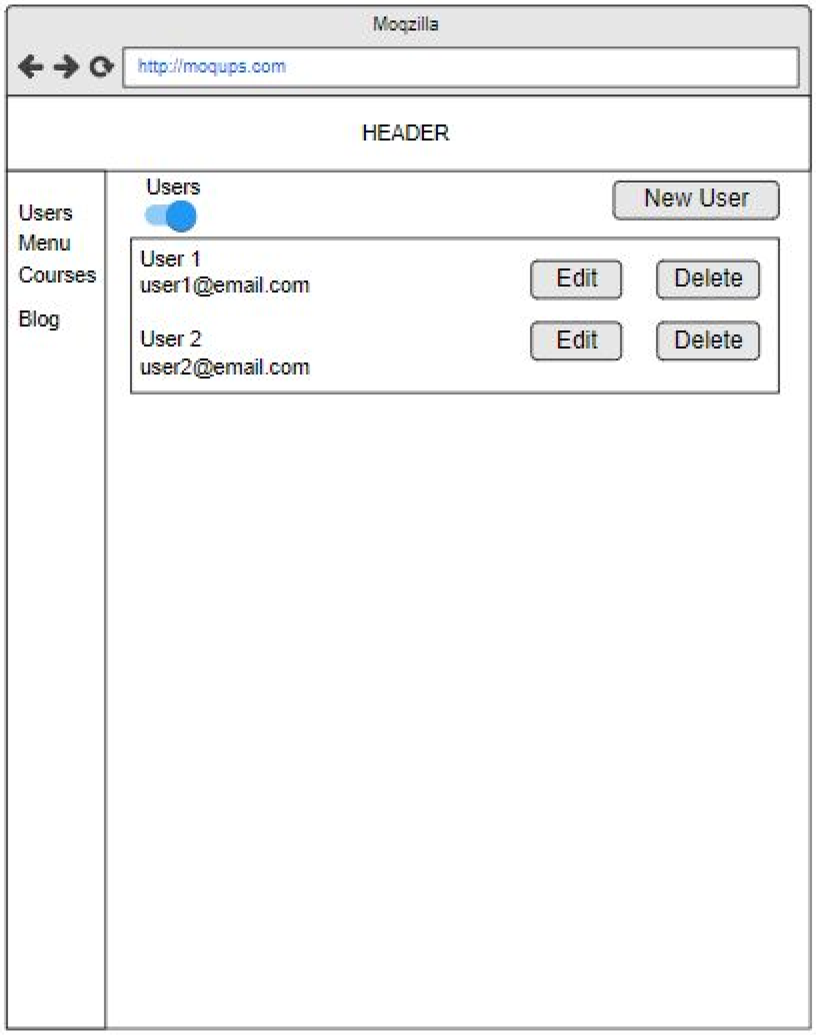
Courses


Moqzilla
http://TFM.com
Course
Blog
Courses
-Link
-Link
-Link
-Link
Email Address
Subscribe

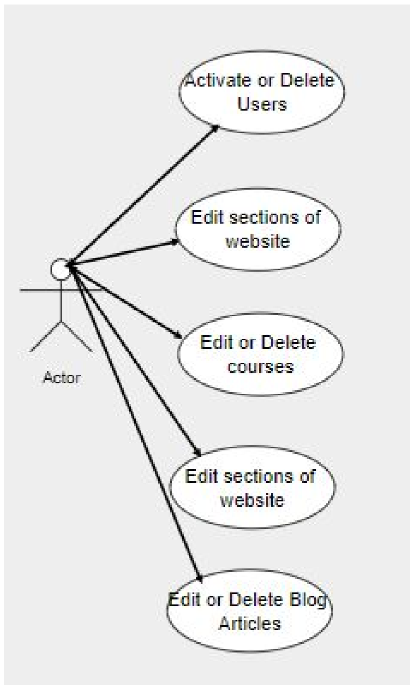
Blog


Moqzilla
http://TFM.com
Course
Blog
Blog
Article Title
Article Title
Article Title
Article Title
-Link
-Link
-Link
-Link
Email Address
Subscribe

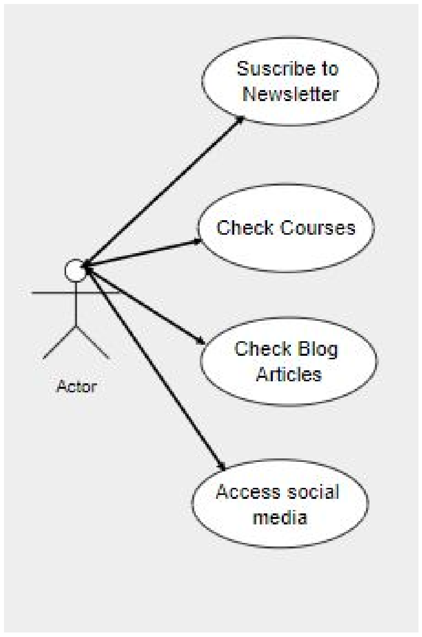
Admin Layout



Use Case Diagram for Admins



Use Case Diagram for Users



Database Design and Structure in MongoDB: COLLECTIONS

Courses

Courses	
id Course:	type: Number,
	unique: true,
	required: true
link:	String,
coupon:	String,
price:	Number,
order:	Number

Menu

Menu	
title:	String,
url:	String,

order:	Number,
active:	Number

Posts

Post	
title:	String,
url:	type: String,
	unique: true
description:	String,
date:	Date

NewsLetter

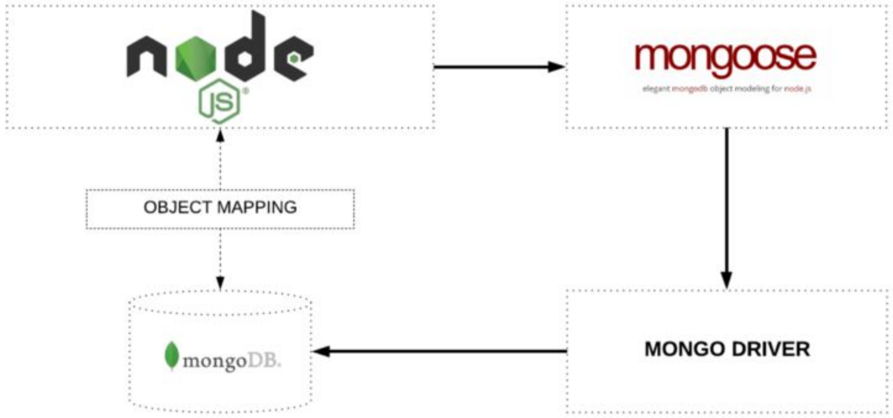
NewsLetter	
E-mail:	Type: string
	unique: true

Courses	
name:	String,

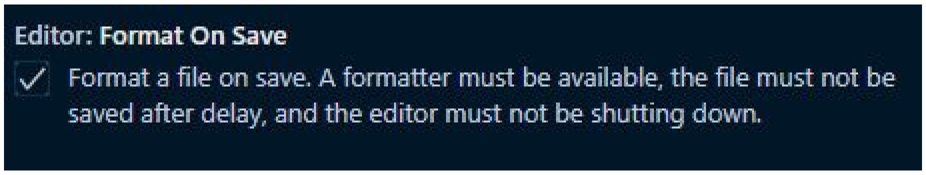
lastname:	String,
email:	type: String,
	unique: true
password:	String,
role:	String,
active:	Boolean,
avatar:	String

Learned lessons

- Sass will help you reduce the styling code for your product, because you can use reusable variables to store stylings.
- Node-sass dependency allows you to natively compile .scss files to css at incredible speed and automatically via a connected middleware.
- Ant design is a design system for enterprise-level products. It allows you to create an efficient and enjoyable work experience.
- The bcrypt-nodejs dependency is installed to encrypt passwords in the server.
- The body-parser dependency installed in the server folder is used to pass information in the body in http requests when using Express.
- The connect-multiparty dependency installed in the server folder is a middleware used to upload images to our server.
- JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.
- Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.
- The database runs in a different port than the database.

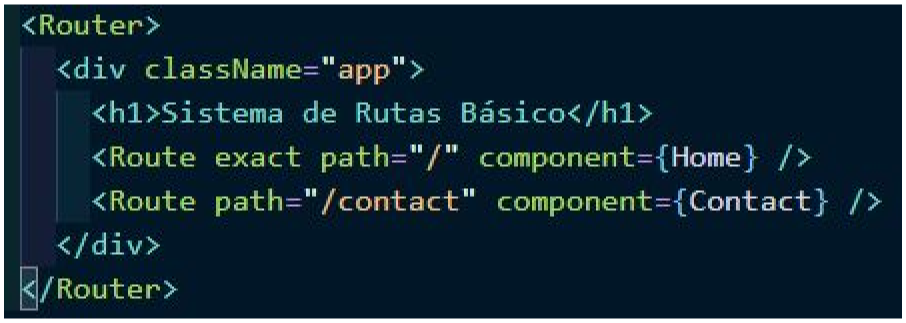


- To make the project format itself every time you “save”, go to File->Preferences->Settings search in the searchbar for the word “save” and click on the button that says “Editor: format on save”:



Editor: **Format On Save**
✓ Format a file on save. A formatter must be available, the file must not be saved after delay, and the editor must not be shutting down.

- If you have a basic routing system like this:

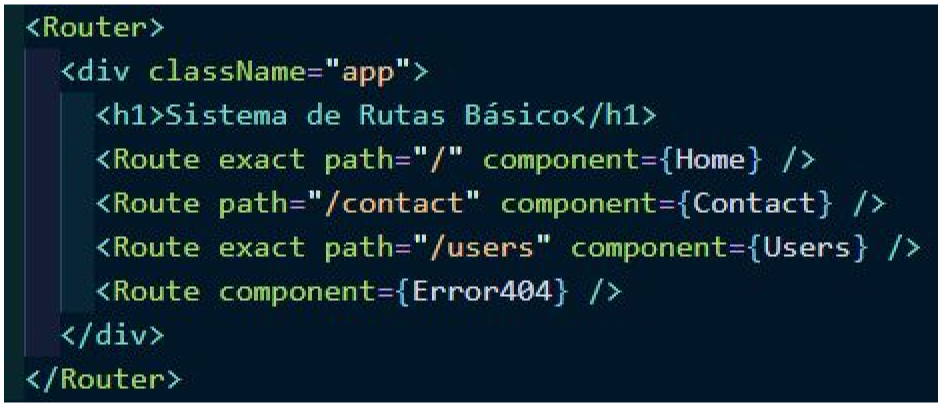


```
<Router>
  <div className="app">
    <h1>Sistema de Rutas Básico</h1>
    <Route exact path="/" component={Home} />
    <Route path="/contact" component={Contact} />
  </div>
</Router>
```

The exact path will render the Home component ONLY when the path is /. In other words, if you delete the “exact” keyword at the Route, React will render both Home and Contact components.

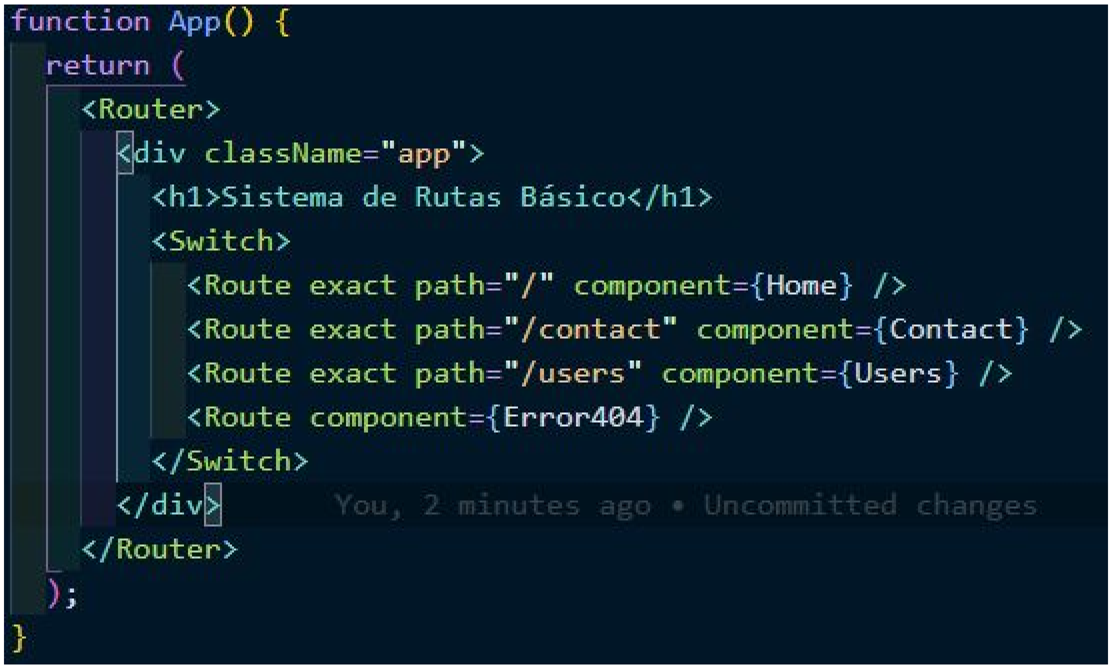
In this case, if you go to http://localhost:3000/contact you will see the contact component rendered in the browser. HOWEVER, if you add something else to the path, like http://localhost:3000/contact/users, it will STILL render the contact component, even though the path “contact/**users**” doesn’t exist. The way to fix this is to add in the contact <Route> the “exact” keyword as well.

- If you have a route like this:



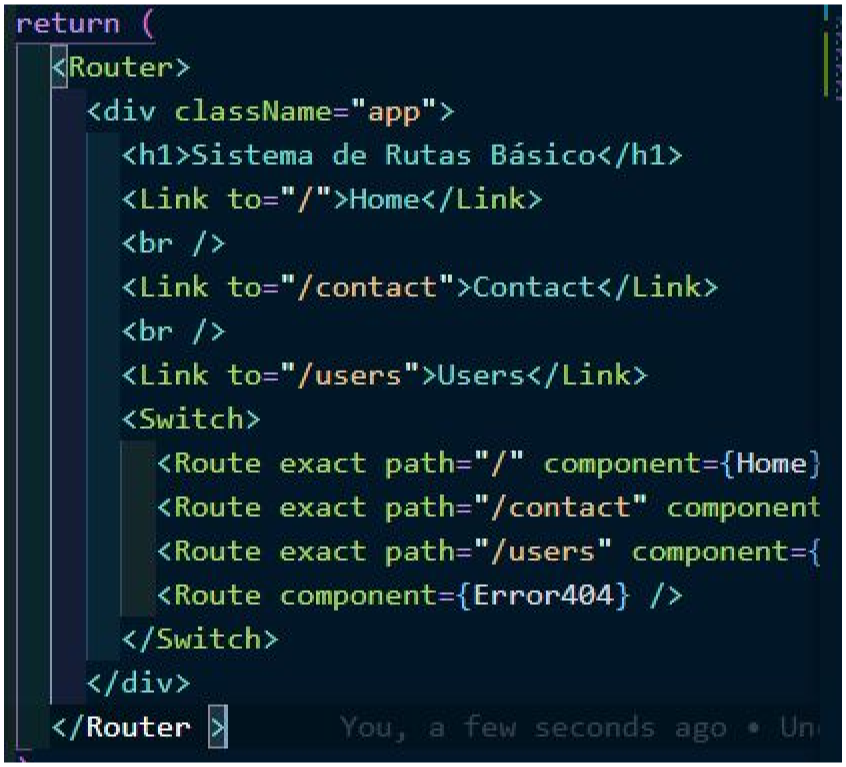
```
<Router>
  <div className="app">
    <h1>Sistema de Rutas Básico</h1>
    <Route exact path="/" component={Home} />
    <Route path="/contact" component={Contact} />
    <Route exact path="/users" component={Users} />
    <Route component={Error404} />
  </div>
</Router>
```

The “Error404” component will always render in screen. So if you go to the path “/” or the path “/contact” or “/users” you will still see the Error404 component. The way to render the “Error404” component when you go to a path that DOESN’T exist, is to use SWITCH react component and encapsulate all the routes inside it, like this:



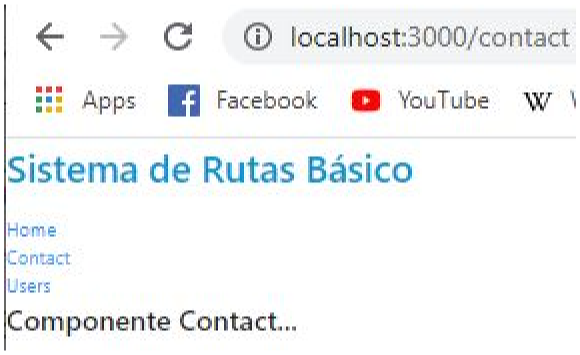
```
function App() {
  return (
    <Router>
      <div className="app">
        <h1>Sistema de Rutas Básico</h1>
        <Switch>
          <Route exact path="/" component={Home} />
          <Route exact path="/contact" component={Contact} />
          <Route exact path="/users" component={Users} />
          <Route component={Error404} />
        </Switch>
      </div>        You, 2 minutes ago • Uncommitted changes
    </Router>
  );
}
```

- With the <Link> react component you can create a basic navigation system, like so:



```
return (
  <Router>
    <div className="app">
      <h1>Sistema de Rutas Básico</h1>
      <Link to="/">Home</Link>
      <br />
      <Link to="/contact">Contact</Link>
      <br />
      <Link to="/users">Users</Link>
      <Switch>
        <Route exact path="/" component={Home}
        <Route exact path="/contact" component
        <Route exact path="/users" component={
        <Route component={Error404} />
      </Switch>
    </div>
  </Router>          You, a few seconds ago • Un
```

It will render like this:



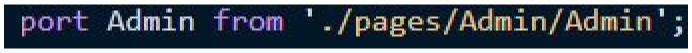
```
localhost:3000/contact
Apps  Facebook  YouTube  W
```

Sistema de Rutas Básico

Home
Contact
Users
Componente Contact...

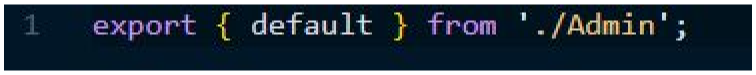
So if you click on the `<Link to="/contact">Contact</Link>`
It will take you to the component “Contact” found in the Contact path

- To avoid repeating having paths like this:



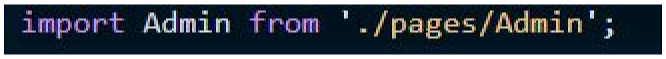
```
port Admin from './pages/Admin/Admin';
```

Inside the /Admin folder, add an index.js file and use this:



```
export { default } from './Admin';
```

This way, you can now use paths like this:



```
import Admin from './pages/Admin';
```

which look better.

- A common pattern in React is for a component to return multiple elements. Fragments let you group a list of children without adding extra nodes to the DOM.
- Overview of Components in Ant Design:
  -Layout: The layout wrapper, in which Header Sider Content Footer or Layout itself can be nested, and can be placed in any parent container.
  -Header: The top layout with the default style, in which any element can be nested, and must be placed in Layout.
  -Sider: The sidebar with default style and basic functions, in which any element can be nested, and must be placed in Layout.
  -Content: The content layout with the default style, in which any element can be nested, and must be placed in Layout.
  -Footer: The bottom layout with the default style, in which any element can be nested, and must be placed in Layout.
- Tabs in Ant Design (source: https://ant.design/components/tabs/)

Tabs make it easy to switch between different views.

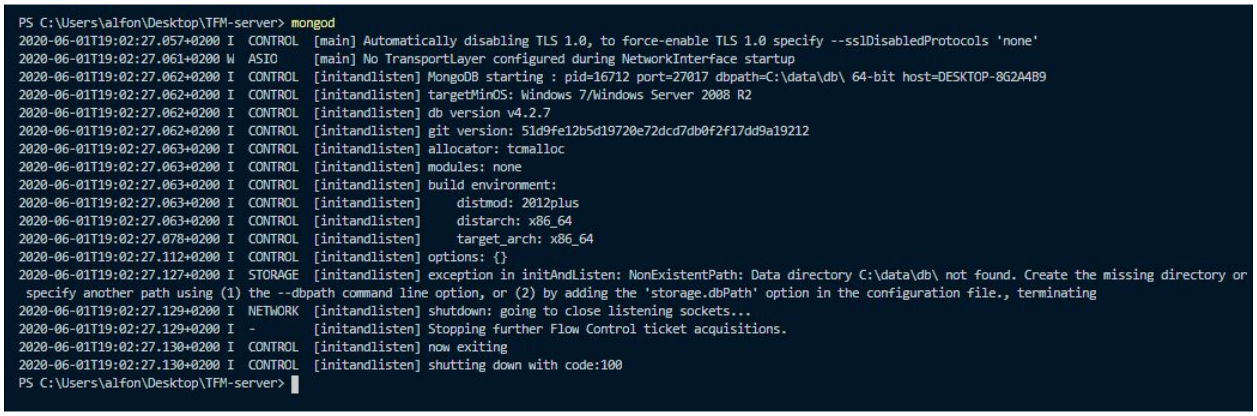
Ant Design has 3 types of Tabs for different situations.

  - Card Tabs: for managing too many closeable views.
  - Normal Tabs: for functional aspects of a page.
  - Radio.Button: for secondary tabs.

- Ant design has its own classes as well, which come inherently with the items you use from Ant Design.
- The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.
- If you want to add global stylings (which can work for all components) do it in App.scss.
- Tokens are pieces of information that allow the authorization process to be performed. Whether this information is readable or parsable by the client (or any party other than the authorization server) is defined by the implementation. The important thing is: the client gets this information, and then uses it to get access to a resource. The JSON Web Token (JWT) spec defines a way in which common token information may be represented by an implementation.
- The useEffect hook tells react that your component needs to do something after render. React will remember the function you passed (we'll refer to it as our "effect"), and call it later after performing the DOM updates.
- To create an endpoint:
  -Create a function for the component in the server, with req and res, put a console.log inside to test if it's working
  -Go to the route folder and create the route
  -Test with postman the route, and if it's right, the console.log will execute in the terminal of the server project.

Unplanned changes and unforeseen events

- In the beginning I wasn't sure if I should create a repo for client and server, or put the same two folders in the same project
- I wasted around 3 hours because of an error that mongodb was giving me. I was unable to make the connection with mongoDB.
- I wasted a lot of time when using Ant Design forms, because you cannot use "onSubmit" in these <Forms>. You must use "onFinish" and delete the ePreventDefault() function.



```
PS C:\Users\alfon\Desktop\TFM-server> mongod
2020-06-01T19:02:27.057+0200 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-06-01T19:02:27.061+0200 W  ASIO     [main] No TransportLayer configured during NetworkInterface startup
2020-06-01T19:02:27.062+0200 I  CONTROL  [initandlisten] MongoDB starting : pid=16712 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-8G2A4B9
2020-06-01T19:02:27.062+0200 I  CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-06-01T19:02:27.062+0200 I  CONTROL  [initandlisten] db version v4.2.7
2020-06-01T19:02:27.062+0200 I  CONTROL  [initandlisten] git version: 51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
2020-06-01T19:02:27.063+0200 I  CONTROL  [initandlisten] allocator: tcmalloc
2020-06-01T19:02:27.063+0200 I  CONTROL  [initandlisten] modules: none
2020-06-01T19:02:27.063+0200 I  CONTROL  [initandlisten] build environment:
2020-06-01T19:02:27.063+0200 I  CONTROL  [initandlisten]     distmod: 2012plus
2020-06-01T19:02:27.063+0200 I  CONTROL  [initandlisten]     distarch: x86_64
2020-06-01T19:02:27.078+0200 I  CONTROL  [initandlisten]     target_arch: x86_64
2020-06-01T19:02:27.112+0200 I  CONTROL  [initandlisten] options: {}
2020-06-01T19:02:27.127+0200 I  STORAGE  [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found. Create the missing directory or
 specify another path using (1) the --dbpath command line option, or (2) by adding the 'storage.dbPath' option in the configuration file., terminating
2020-06-01T19:02:27.129+0200 I  NETWORK  [initandlisten] shutdown: going to close listening sockets...
2020-06-01T19:02:27.129+0200 I  -        [initandlisten] Stopping further Flow Control ticket acquisitions.
2020-06-01T19:02:27.130+0200 I  CONTROL  [initandlisten] now exiting
2020-06-01T19:02:27.130+0200 I  CONTROL  [initandlisten] shutting down with code:100
PS C:\Users\alfon\Desktop\TFM-server>
```

The issue had seemingly no reason, and it fixed itself randomly, for no reason as well.

- I'm facing a huge difficulty when using tokens for authorization and persmissions.

Conclusions of the project and your evolution respect your knowledge before starting the master

The Master was literally a life-changing experience for me. I got into a new World I knew nothing about and learned so much in the process...and I loved it! I learned a lot about deep life lessons in the process, such as patience, not comparing yourself to others and the importance of study and hard work. I also noticed a great improvement in my understanding of everything in the last project. I could find errors and solve bugs faster than before

About Product

- Swot Analysis:

Strengths:	**Weaknesses:**
● The website has a newsletter that keeps the users interested. ● The website is easy to use and provides quality information for the users	● There are other websites like Udemy that offer far more courses ● There a other courses websites that offer more features
Opportunities: ● The COVID crisis can make the website more popular as people will want to learn from home. ● Not many courses websites offer a Blog, this will attract more users.	**Threats:** ● The website can lose popularity quickly after the Covid crisis is done. ● It is difficult to stand from the rest if you are new in the market.

- **Competition Analysis**:
  The main competitors in this case are websites like Udemy, Coursera, Khan Academy, among others. These websites have a long time in the market. They offer different kinds of courses. However almost none of the websites like this offer a Blog and a Newsletter. If the people don't come to our websites for its courses, perhaps they will obtain value by the Blog articles and the newsletter. This can make our website stand out from the rest.

- **Cost of developing the website:**
  This is a small project, fit for a junior developer. Assuming a Junior developer works 40 hours a week, for 4 weeks in a month, it would be roughly 160 hours of work. Assuming that on average a Developer can earn around 15 euros per hour, and that the total hours of the project is 84, that would mean that the project could cost around 1260 euros in total, approximately.