# Getting started with the front-end code base

In this lesson you will explore the structure of the frontend code base, how to install its dependencies and how to build and run the portal.

## About the front-end project

Before exploring the front-end code base, let me give you a few details about it:

- The front-end is a simple single-page application that demonstrates one way to interact with all the Play Economy microservices. A single-page application is a web application that dynamically rewrites the current web page with new data from a server, instead of constantly loading new pages.
- It is built using React, which is a JavaScript library for building interactive user interfaces. I chose React because at the time of creating the course it was one of the most popular front-end frameworks and is easy to learn, assuming that you have worked with JavaScript before.
- It uses Create React App to simplify how you do local development, how you build and debug the code in your box, and how you package everything in preparation for deployment.
- Finally, it is hosted in a Node.js server, a very popular runtime for JavaScript applications.

## Please keep in mind…

There are a few things that I'd like you to keep in mind as you work with the provided front-end portal in this and the next lessons:

- This is not a course about front-end development and certainly nor about React. As a backend developer myself I only know enough React to build this simple portal. However, I'm probably making lots of beginner mistakes, and I'm probably not following client side best practices in many places. So please don't use this code as a template for a production level application, but more as a basic example of one way you could interact with the microservices from a modern client.
- You won't be coding this front-end in this course. There is both not enough time and I'm not qualified to teach you how to build a front-end step by step. Instead, I'm providing you the finished portal so you can use it AS-IS. It should work with no issues and no changes required, provided you make a small update to the microservices as I'll explain in the next lesson.
- In this lesson I'll give you a quick overview of the structure of the frontend code base, with a focus on the areas that interact with your microservices, as opposed to the many components that handle layout, rendering and a many other things.
- Finally, please check out the links below if you would like to know more about the tech used in the front-end project:
    - https://reactjs.org
    - https://create-react-app.dev
    - https://react-bootstrap.github.io

## (Demo prep)

- Delete .eslintcache

## Exploring the code base

1. Open VS Code terminal and switch to d:\projects

2. Switch to Play.Frontend and:

   code . -r

3. Describe the README file

4. Install the dependencies:
   npm install

5. Start the frontend:
   npm start

6. Browse to the frontend

7. Describe the home page

8. Review the files ( I won't go over every single file ):

| **File** | **Description** |
|---|---|
| README.md | Project summary, prerequisites, build instructions |
| package.json | App version, dependencies, scripts |
| | |
| public (directory) | The root folder served when the site loads |
| index.html | The page template with the root div container where the react components will be injected.<br><br>config.js is also injected here |
| config.js | Defines the urls to microservices and infrastructure services<br>Can be replaced at deployment time (HELM) with PROD urls |
| | |
| src (directory) | Contains all the dynamic React components. The core app logic lives here. |
| index.js | The JavaScript entry point<br>Places the App component into the root div from index.html<br>Imports index.css |
| App.js | The root React component<br>Uses the Layout component to define the layout of the app<br>Uses the Route component to define the client site route to other components<br>Uses App.css |
| | |

| | |
|---|---|
| components (directory) | Contains all the react components (except for App.js) |
| Layout.js | Defines the application layout |
| NavMenu.js | Defines the navigation bar at the top<br>Can be collapsed in smaller screens |
| Footer.js | Defines the footer content at the bottom |
| Home.js | The home page |
| Catalog.js | Defines the Catalog section<br>Has the logic to list Catalog items |
| Inventory.js | Defines the Inventory section |
| | Has the logic to show the inventory bag of a user |
| ItemModal.js | Adds buttons to the list for adding and editing items<br>Adds a Modal component to present ItemForm |
| ItemForm.js | Form to add or edit items |
| GrantItemModal.js | Adds a button to grant items to a user<br>Adds a Modal to show GrantItemForm |
| GrantItemForm.js | A form that allows granting items to a user |
| | |
| .vscode (directory) | VS Code specific files |
| launch.json | Defines how to start the Node.js server and the browser |

9. Start Catalog service

10. Browse to Catalog section in portal

11. Notice catalog items won't load

12. Open Edge dev tools (F12) and notice error in Console

In the next lesson we will learn about Cross-Origin Resource Sharing (CORS) and how it can help us enable the communication between the front-end and our microservices.