Lab 4 Full Stack Spring Boot with ReactJS

Updated: 2024-11-06.

Introduction to the lab

Learning outcomes

- Develop web projects with Spring Boot (backend) and ReactJS (frontend). Create and persist entities into a relational database through web interfaces.
- Deploy ReactJS application in Docker.

Submission

This is a week lab. You may submit as you go but be sure to complete and submitted by the deadline outlined in the class plan (see slides 10 and 11 from the first class).
Remember to create the "lab4" folder in your Git repository. Be sure to create and update the lab "notebook, e.g. in 🗁lab4/README.md . [The notebook is where you take notes of quick reference info, links and visuals, so you can study from this content later; it is an import part of your submission.]

4.1 First Dummy App

ReactJS depends on Node.js; therefore, ensure you have installed a recent version of Node.js. If you already have Node.js installed, you should update it and use version v22.11.0 and npm version 10.9.0.

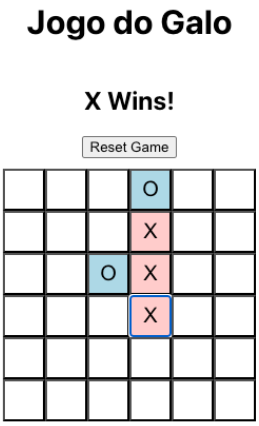
a) Create your first ReactJS App using the Create React App command. There are other React-based Frameworks available, and these can be used in the group project, but for this guide, we will just use ReactJS.
b) Complete the guided exercise available from "Quick Start".

4.2 Props and States

In React, **props** and **state** play key roles in managing data and behavior across components. **Props** enable data sharing between components by passing information from a parent to its child components. Though similar to HTML attributes, props are far more flexible—they can carry any JavaScript value, including objects, arrays, and functions, allowing you to customize child components' behavior and content. **State**, on the other hand, is used to store dynamic data within a component. While state can hold any JavaScript value, including objects, directly modifying objects in state is discouraged, as it may cause unexpected issues in your application. To update an object in state, create a new object (or copy the existing one) and set the state to reference this updated version. This method ensures changes are properly detected, maintaining React's reactivity and efficient rendering process.

a) Work through the guided exercise on "Passing Props to a Component" to understand how to share data between components using props.
b) Complete the guided exercise on "Updating Objects in State" to learn best practices for modifying state objects in React without directly mutating them.
c) Follow the steps in the guided exercise under "Passing Data Deeply with Context".

d) Using the information provided in the Quick Start guide and the knowledge acquired in Software Design Patterns from the previous semester, create a Tic Tac Toe game (Jogo do Galo) with a 6 x 6 grid. You can follow this guide to obtain something similar to the example below:



4.3 Consume REST API services

React Hooks are a powerful feature that enable developers to use state and other React features without writing class components. The two most commonly used hooks are useState and useEffect. The useState hook was already explored in the previous exercises. The useEffect hook, on the other hand, allows functional components to perform side effects, such as fetching data, updating the DOM, or setting up subscriptions. It replaces lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount found in class components.

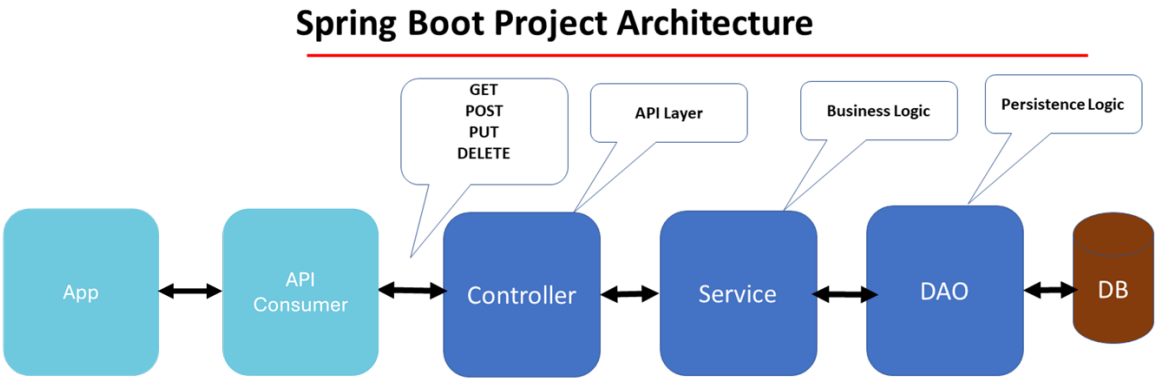
a) See the guide for "Connecting to an external system".

b) Proceed with the guided exercises that demonstrate how to Consume REST APIs in React.

4.4 Wrapping-up and integrating concepts

Consider the last exercise from the previous lab (Exercise 3.3).
Let us replace the Postman Client by a ReactJS interface to consume the services:

a) Separate the "boundary" of the problem (API Consumer) from the components, by adding an intermediary component.



b) Deploy the application to a Docker container. Consider using docker compose (related example).