**Lab 05 - Create Single Page Application with Spring Boot RESTful API**

# 1. Introduction

Imagine you're a new fresher Spring Boot developer of IT department in a University. Your manager has asked you to develop a BE (RESTful API) and FE (React JS) for Orchid Management to manage just a simple information of an orchid.

Orchid (OrchidID, OrchidName, isNatural, orchidDescription, orchidCategory, isAttractive, orchidURL)

The RESTful API has to support adding, viewing, modifying, and removing information for an orchid - a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

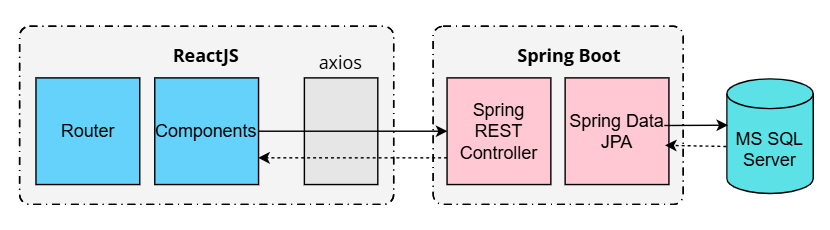
This lab explores

* Create a RESTful API using **Spring Boot** 3 with MS SQL Server. An **SQL Server** **Database** will be created to persist the orchid data (reading and managing) by **Spring Data JPA**
* React Client sends HTTP requests and retrieve HTTP responses using axios, shows data on the components.

# 2. Lab Objectives

In this lab, you will:

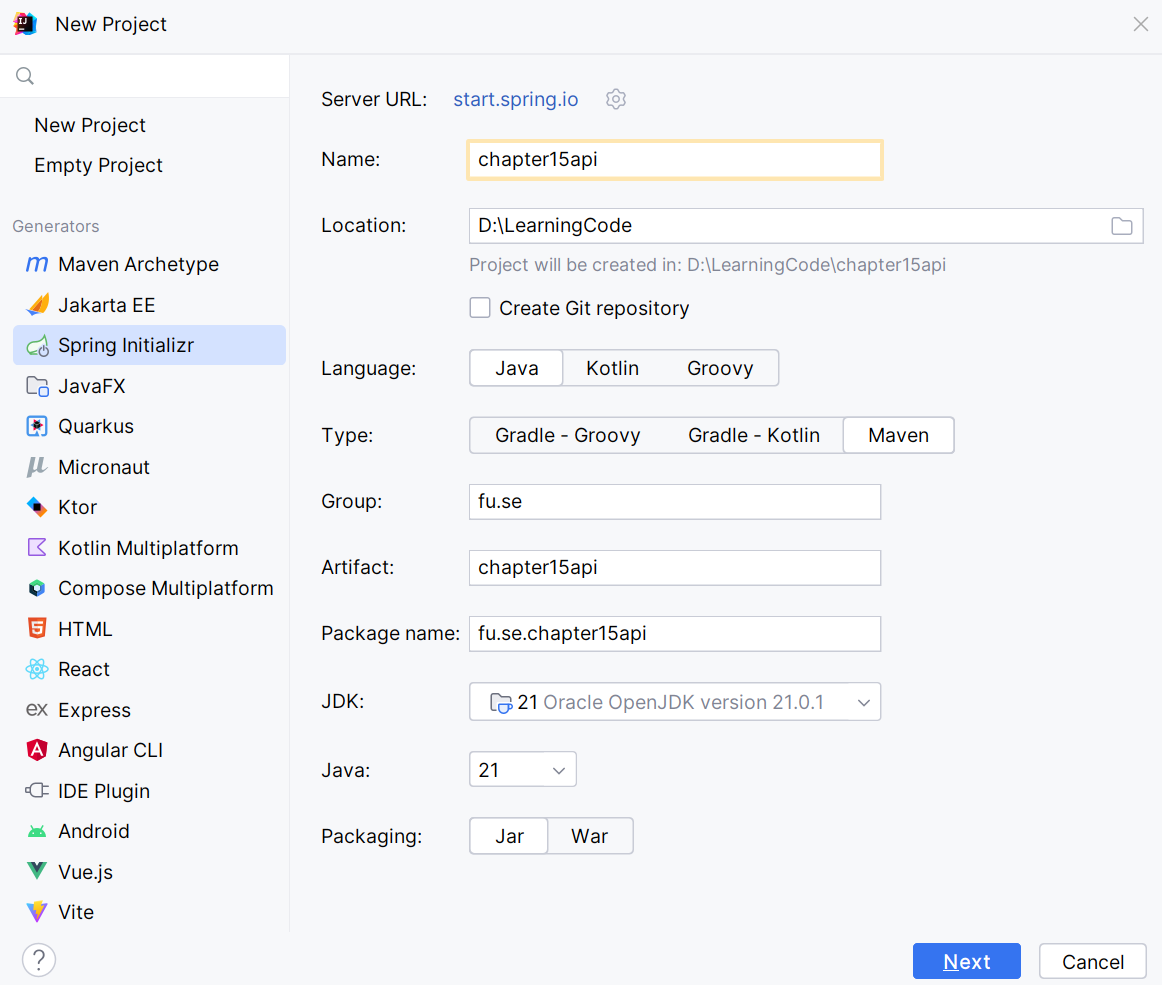
* Use the **Intellij** **IDEA** to create Spring Initializr Project (Maybe you can create the Spring Initializr project in <https://start.spring.io/> then open this project in IntelliJ IDEA.
* Choose **Spring Web** for Building web, including *RESTful*, applications using Spring MVC. Uses Apache Tomcat as the default embedded container. **Spring Data JPA SQL for** persisting data in SQL stores with Java Persistence API using Spring Data and Hibernate. **MS SQL Server Driver SQL -** A JDBC and R2DBC driver that provides access to Microsoft SQL Server and Azure SQL Database from any Java application.
* Use in SQL Server Database as a Data Source
* Develop 3-Layer with Repository Architecture to perform CRUD actions using RESTful API with Spring Boot 3.
* React Client sends HTTP requests and retrieve HTTP responses using axios, shows data on the components.
* Run the project and test the FE, BE interactions.



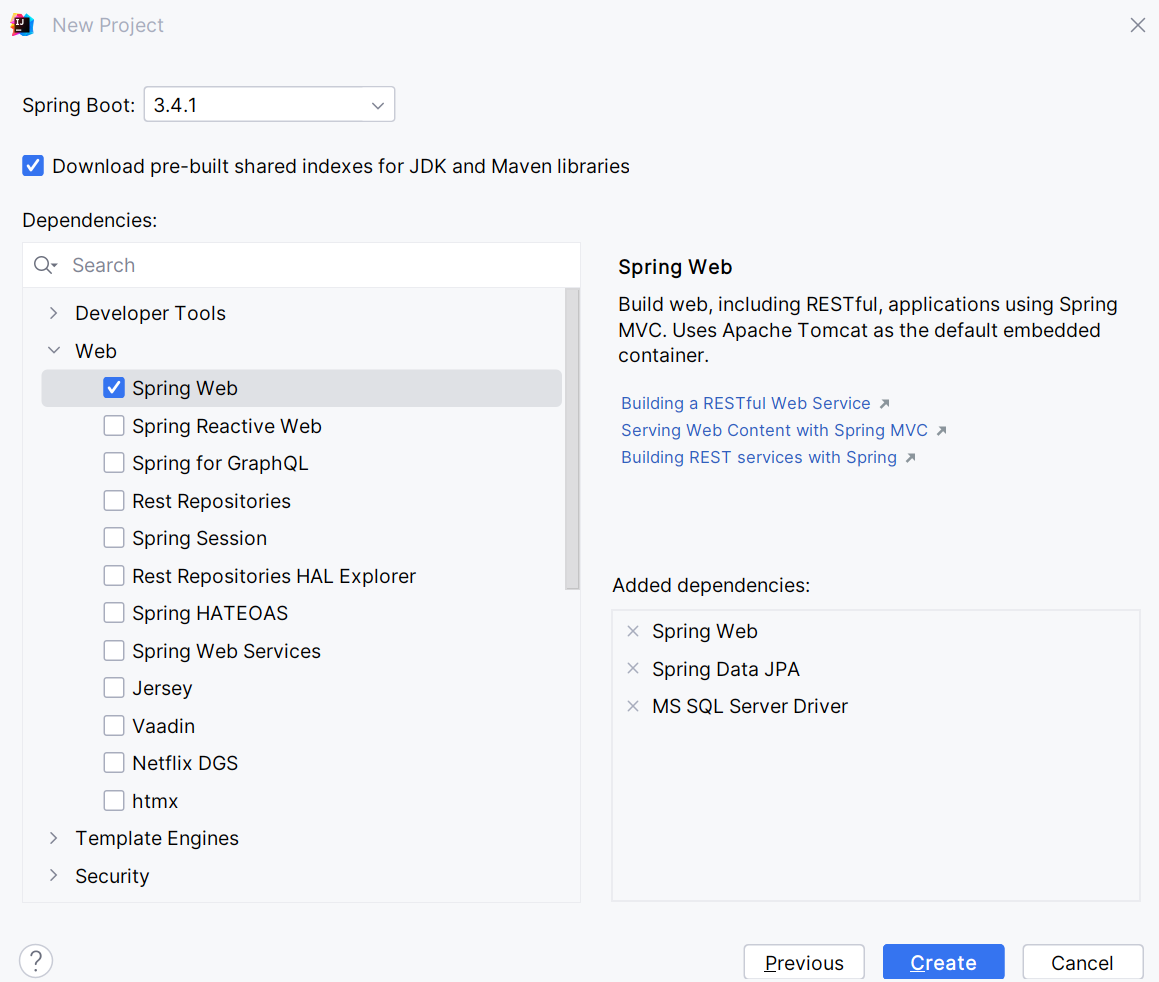
# Activity 01: Design the RESTful API for Orchid Management

**Step 01**. Open IntelliJ IDEA, File | New | Initializr Project

In the left pane of the New Project wizard, select Spring Initializr Project.

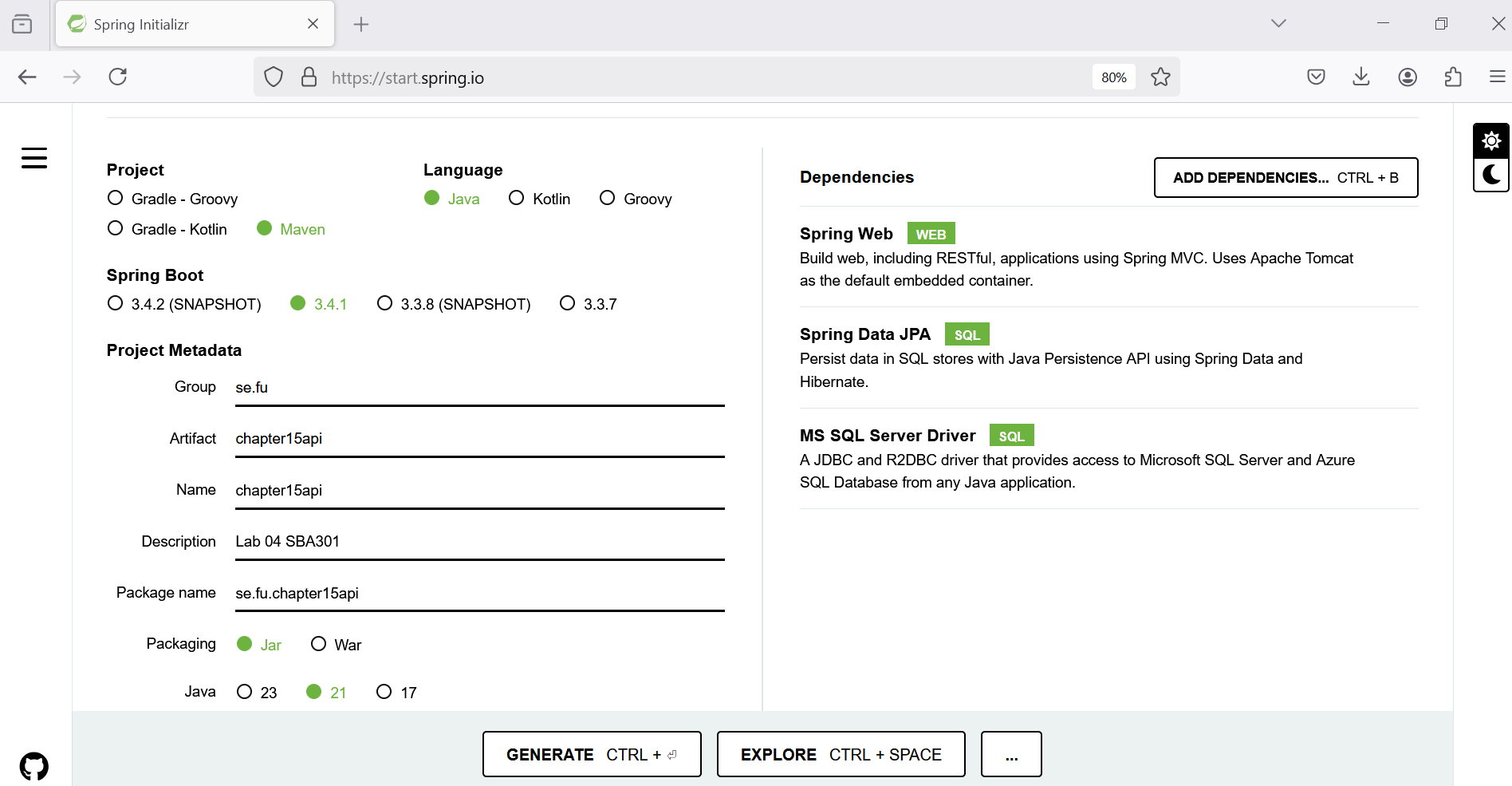


**Step 02.** Fill the information → Next



**Step 03**. Choose the dependency/dependencies → Click Create

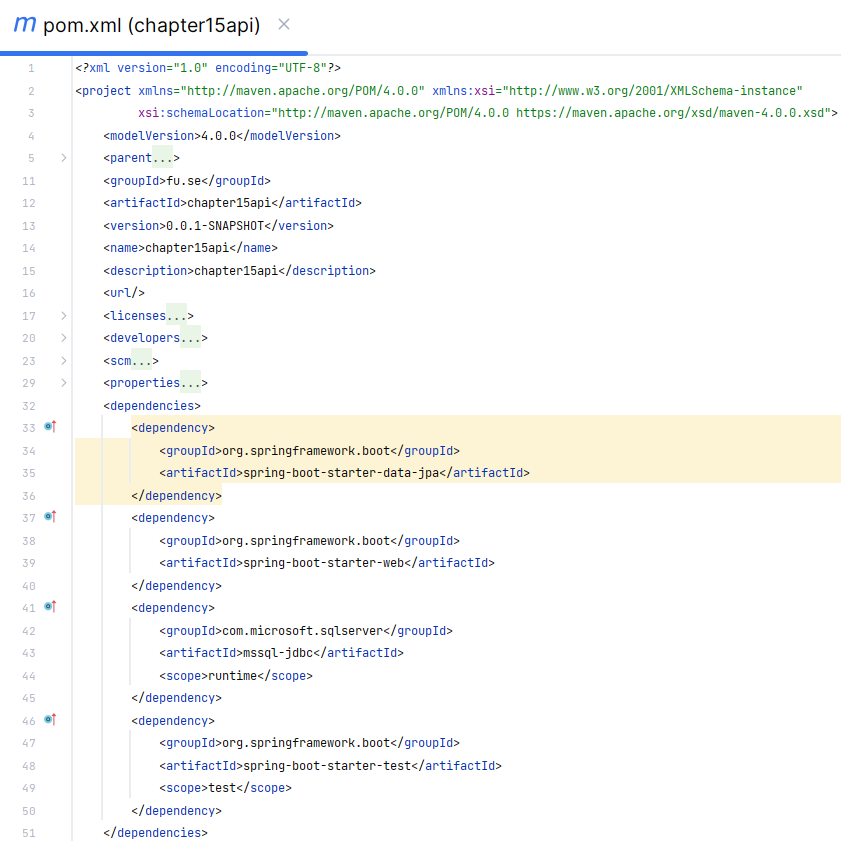
Or you can using Spring Initializr project in <https://start.spring.io/> then open this project in IntelliJ IDEA



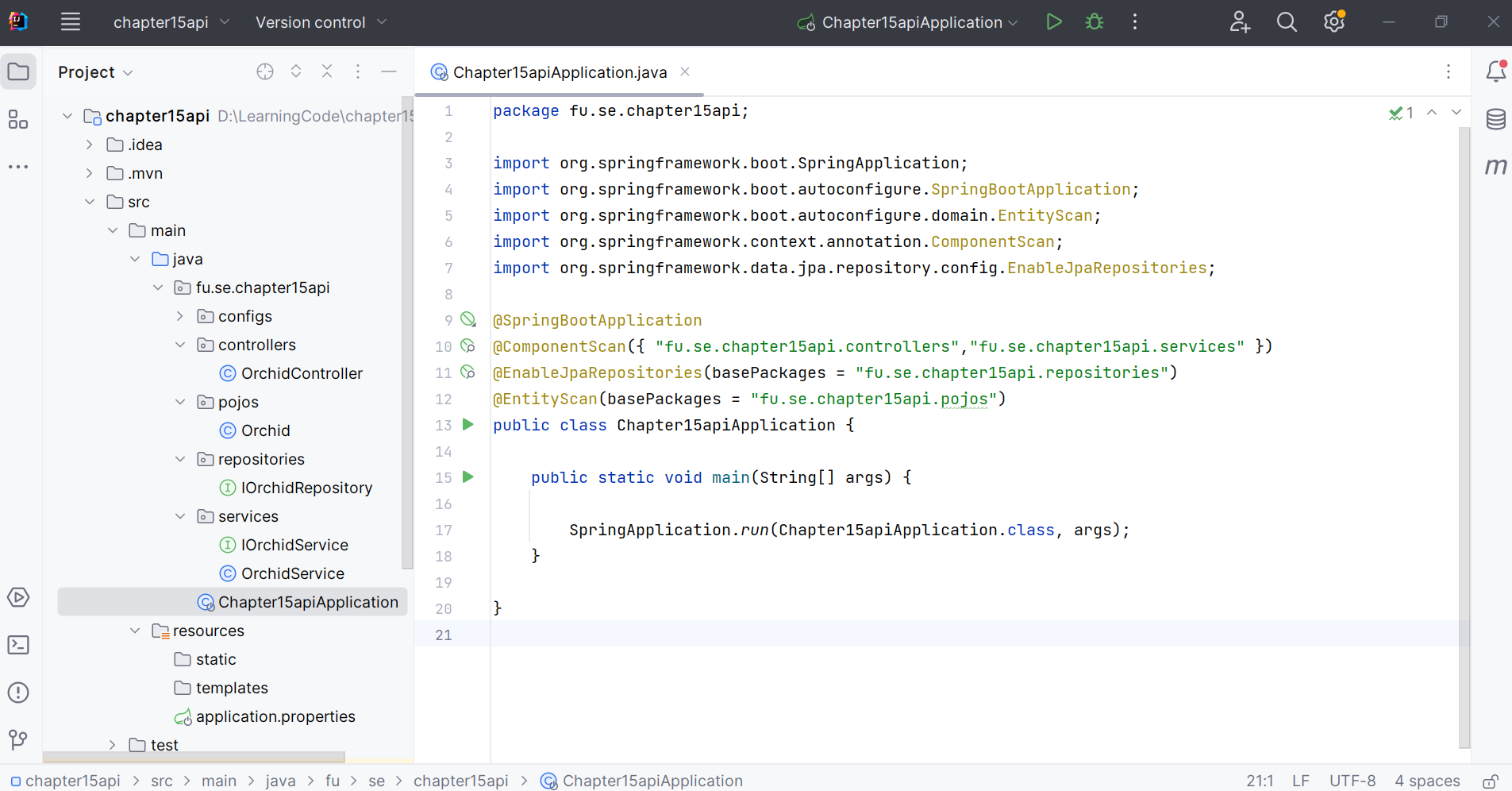
**Step 04**. Edit pom.xml, add the dependencies for

* MS SQL Server Driver **mssql-jdbc**
* Spring Web - RESTful API: **spring-boot-starter-web**
* Spring Data JPA for Paging and Sort data: **spring-data-jpa**
* Testing **spring-boot-starter-test**

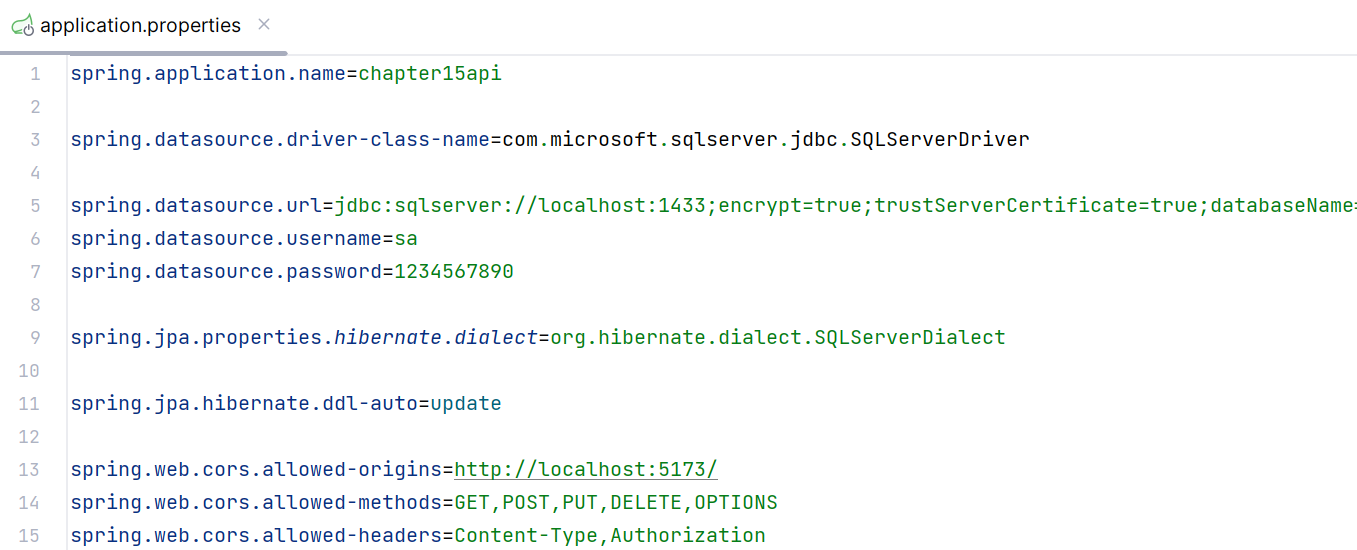
<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>**spring-boot-starter-data-jpa**</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>**spring-boot-starter-web**</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.microsoft.sqlserver</groupId>  
 <artifactId>**mssql-jdbc**</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>**spring-boot-starter-test**</artifactId>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

****

**Step 05**. The structure of the Lab project

****

**Step 06**. The application.properties file



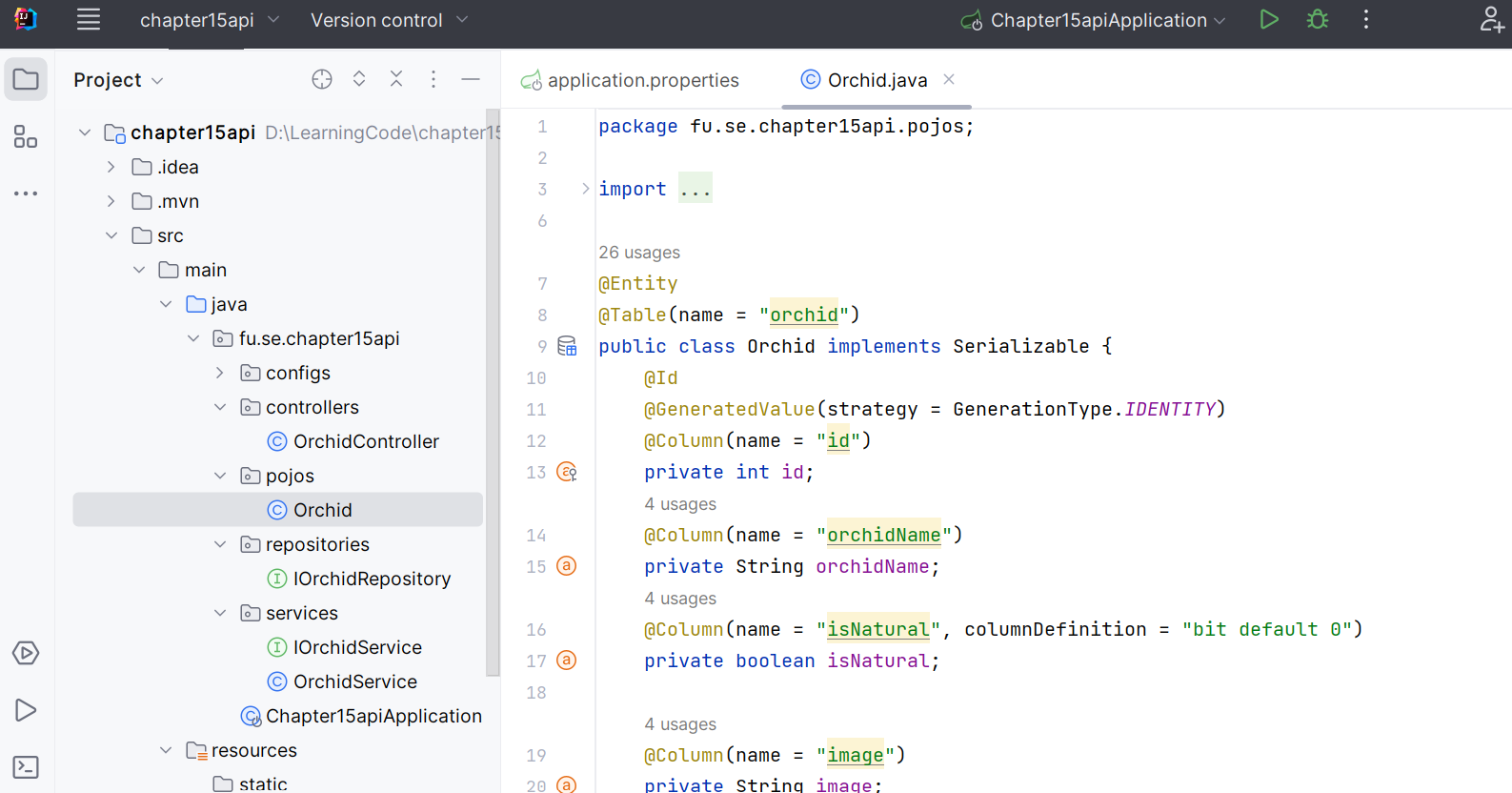
# Activity 02: Implement RESTful API Orchid Management with CRUD functions

**Step 01**. Create **pojos** package, then create Orchid Entity

Entity Class: Represents a database table

Annotations:

* @Entity: Marks the class as a JPA entity
* @Id: Specifies the primary key field
* @GeneratedValue: Strategy for generating primary key values
* @Column: Maps a field to a database column



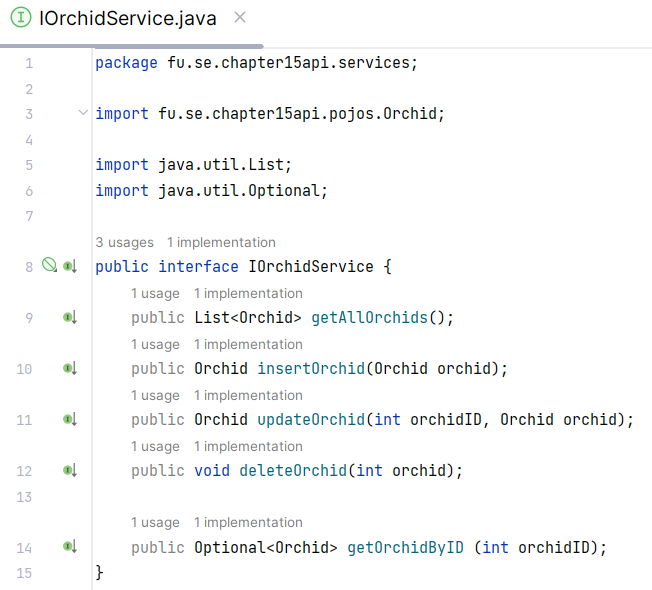
**Step 02**. Implement Repository by creating **repositories** package, IOrchidRepository interface

* Create JPA repository interface, this interface extending JpaRepository
* Provides CRUD operations and more
* No implementation needed (Spring Data JPA provides it)



**Step 03**. Implement Service Layer includes: IOrchidService interface, OrchidService class in **services** package.

* Define service interfaces: Create interfaces to define the business logic for application.
* Implement service classes: Implement the service interfaces and use the repositories to interact with the database.
* Use the IOrchidRepository to interact with the Repository (Data Layer)





**Step 04**. Implement REST Controller

@RequestMapping - Maps HTTP requests to specific controller methods based on the request path, HTTP method, and other criteria. It can be used at both the class and method level. Attributes of @RequestMapping:

* value: Specifies the request path.
* method: Specifies the HTTP method (e.g., RequestMethod.GET, RequestMethod.POST).
* params: Specifies request parameters that must be present.
* headers: Specifies request headers that must be present.
* consumes: Specifies the content type of the request body.
* produces: Specifies the content type of the response body.

The annotations are shortcuts for @RequestMapping with specific HTTP methods. They simplify the code and improve readability.

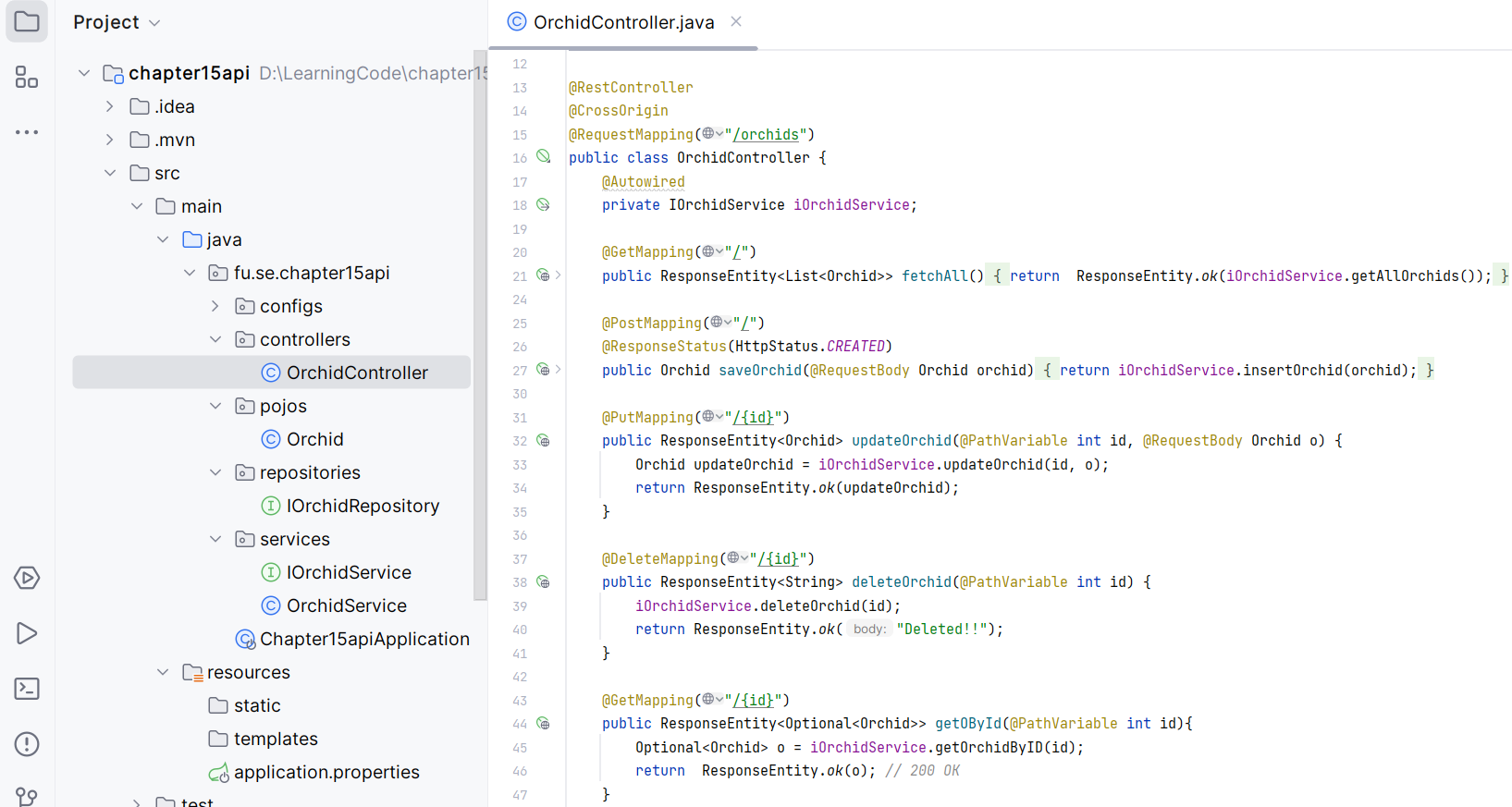
* @GetMapping
* @PostMapping
* @PutMapping
* @DeleteMapping
* @PatchMapping

@PathVariable: Extracts values from URI path variables and binds them to method parameters.

@RequestParam - Extracts values from query parameters and binds them to method parameters.

@RequestParam attributes:

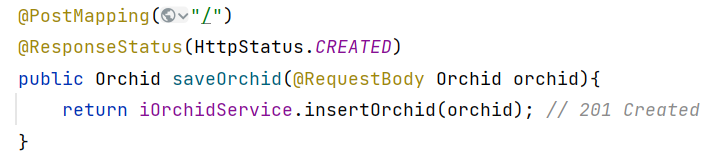
* value: Specifies the name of the query parameter.
* required: Indicates whether the parameter is required.
* defaultValue: Specifies a default value if the parameter is not present.

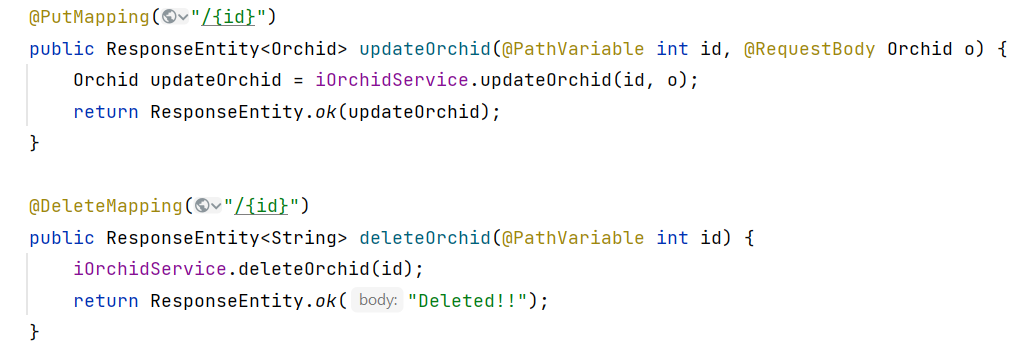


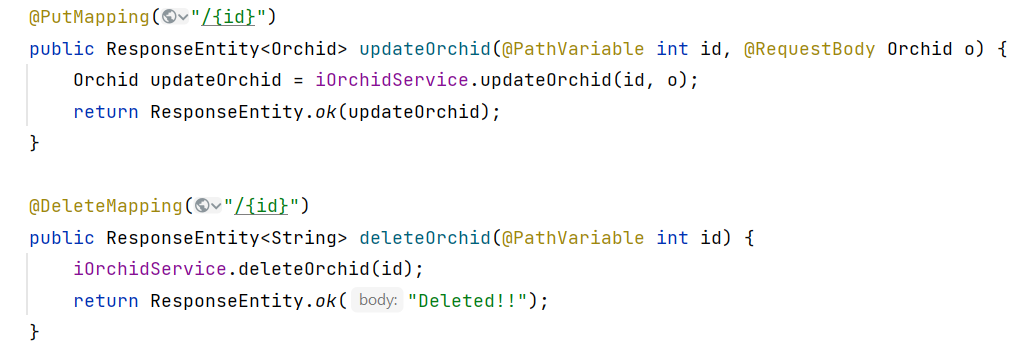
The details of action methods in REST Controller

****

****

****

****

****

**Step 05**. Configure CORS

Configuring CORS (Cross-Origin Resource Sharing) involves specifying which domains or origins are permitted to access resources hosted on a server. CORS is essential for web applications where frontend code (running in a browser) needs to interact with a backend server hosted on a different domain.

****

**Step 06**. The main entry point for Spring Boot application

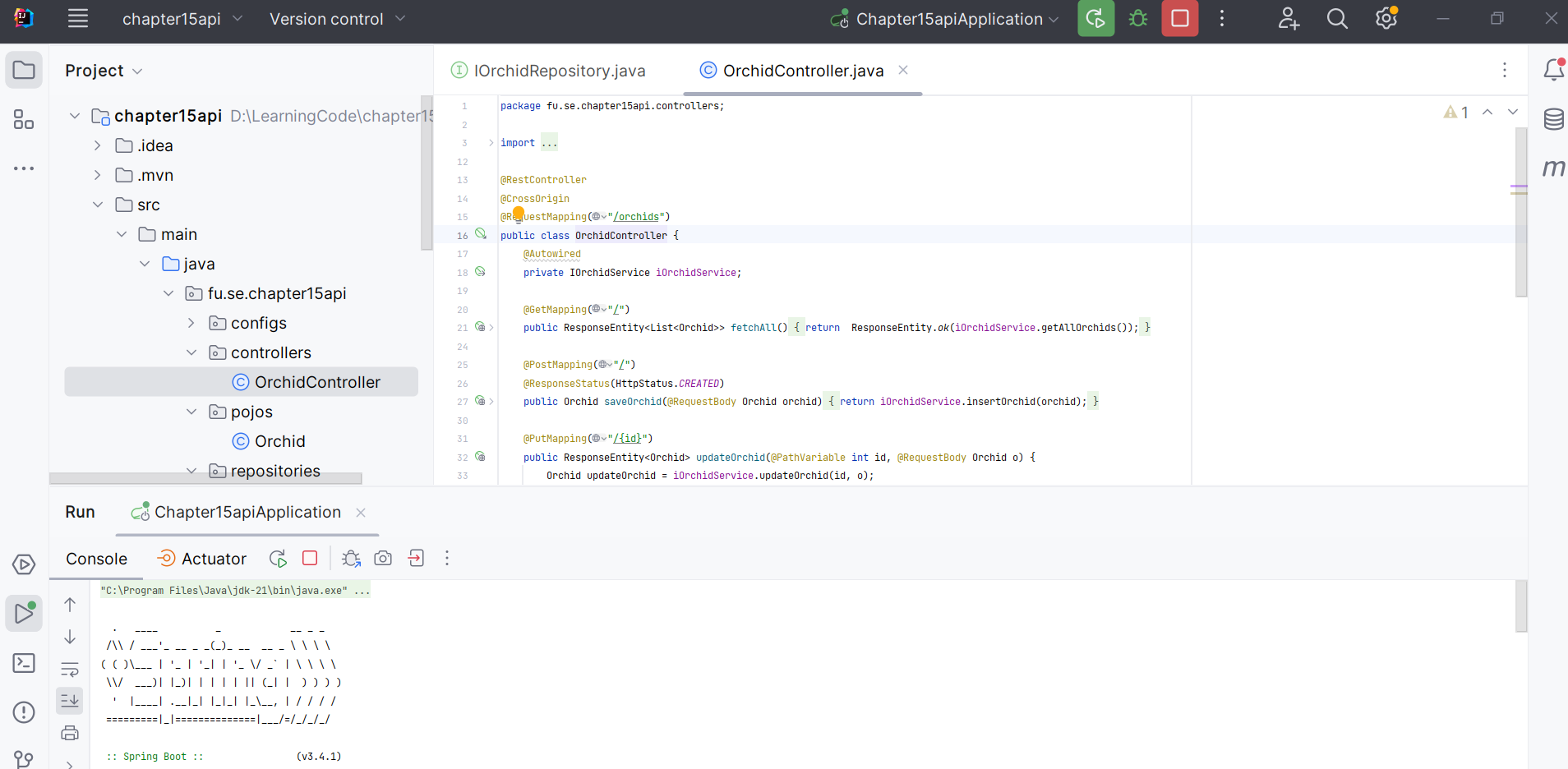
The @SpringBootApplication annotation is a convenience annotation in Spring Boot that combines several annotations to simplify the configuration and initialization of a Spring Boot application.

When you use @SpringBootApplication, it automatically includes the following annotations:

* **@EnableAutoConfiguration:** This annotation tells Spring Boot to automatically configure your application based on the libraries on the classpath. For example, if Spring Boot detects that you have a database driver in your project, it will automatically configure a datasource.
* **@ComponentScan**: This tells Spring to scan the current package and its sub-packages for annotated components (such as @Component, @Service, @Repository, and @Controller) to register them as Spring beans.
* **@Configuration**: This marks the class as a source of bean definitions for the application context, meaning that you can define @Bean methods in this class.

****

**Step 07.** Run the RESTful API Project



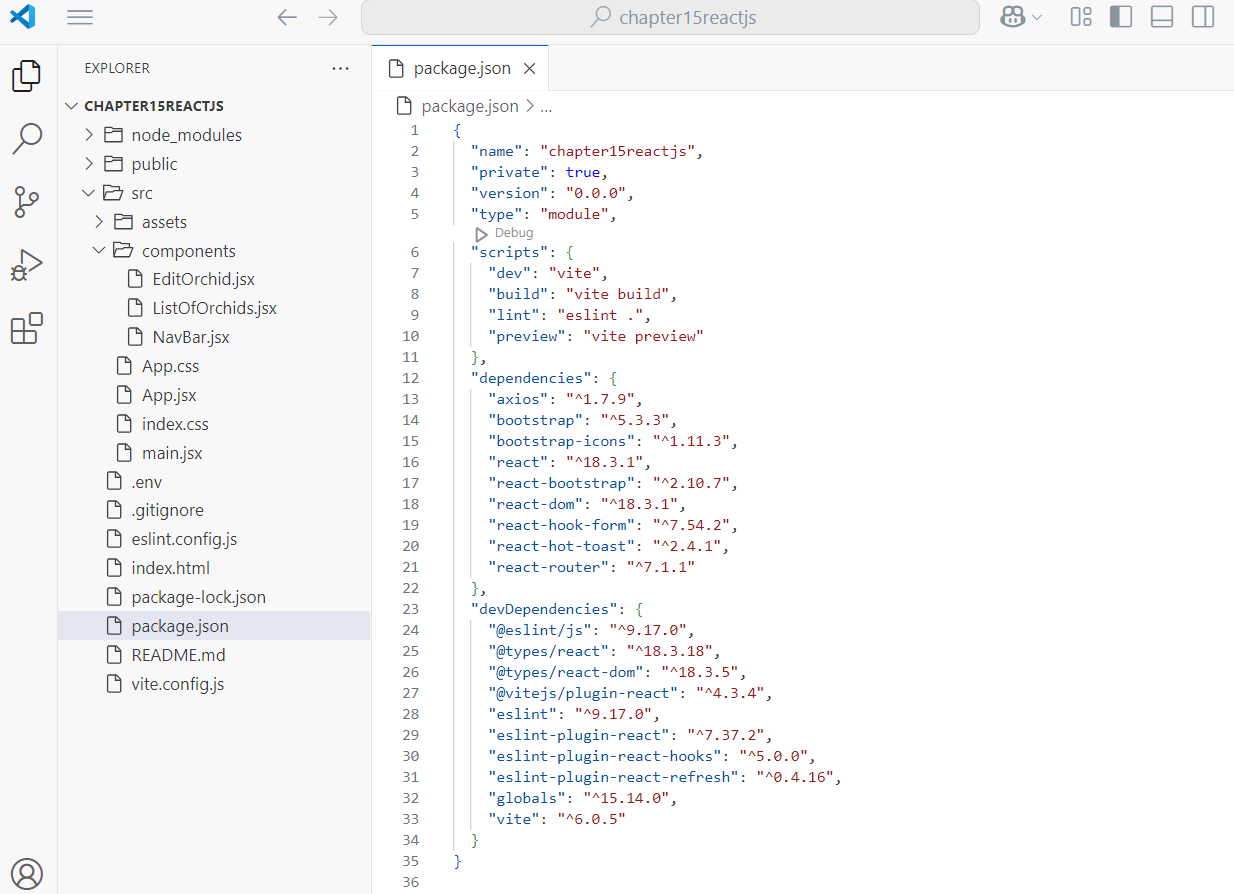
**Step 07.** Test the RESTful API using Postman

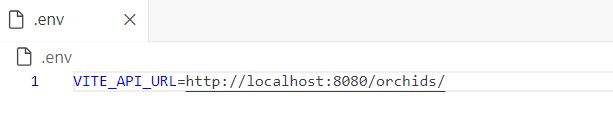
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | API name | HTTP Method | Path | Status Code |
| 1 | GET Orchids | GET | *http://localhost:8080/orchids/* | 200 OK |
| 2 | POST Orchid | POST | *http://localhost:8080/orchids/* | 201 Created |
| 3 | DELETE Orchid | DELETE | *http://localhost:8080/orchids/{id}* | 204 No Content |
| 4 | PUT Orchid | PUT | *http://localhost:8080/orchids/{id}* | 200 OK |
| 5 | GET Orchid | GET | *http://localhost:8080/orchids/{id}* | 20 OK |

# Activity 03: Create FE Application using ReactJS

**Step 01.** Set up the React project with Vite

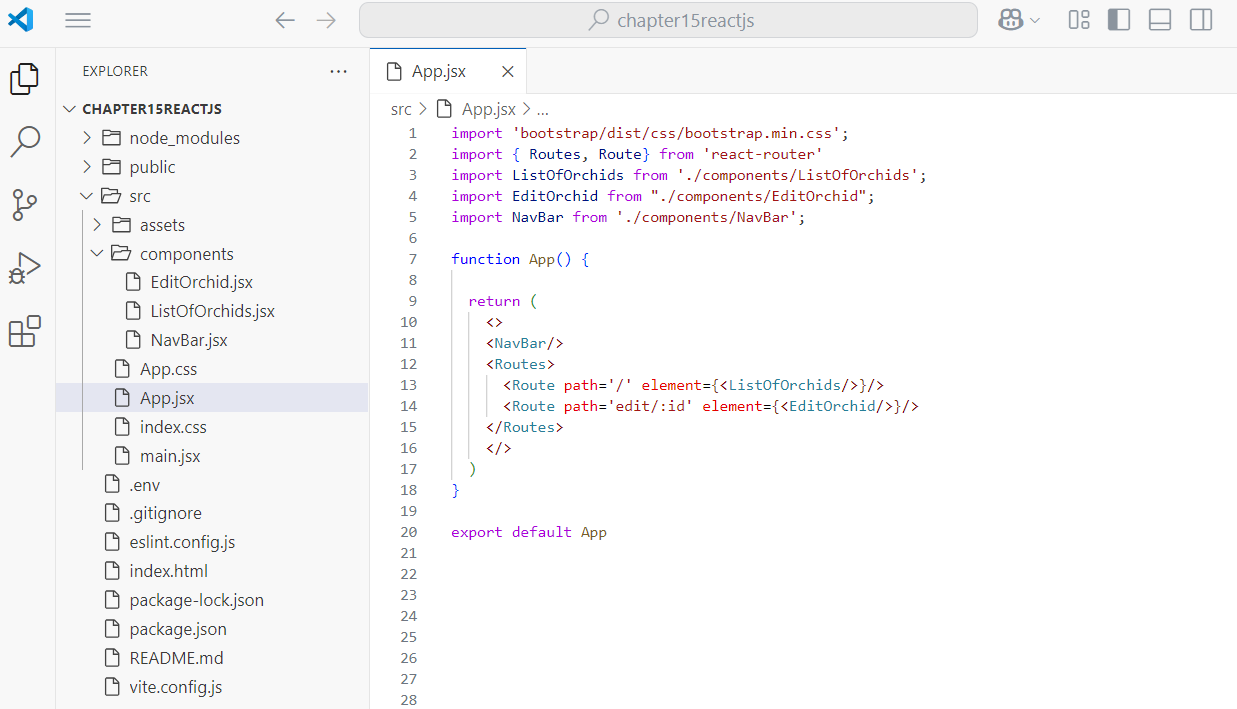
* Open the terminal and run “*npm create vite@latest chapter15reactjs*”
* Choose "React" framework and "JavaScript" variant.
* Install dependencies: *npm install*
* Create a text file that contains environment variables

****

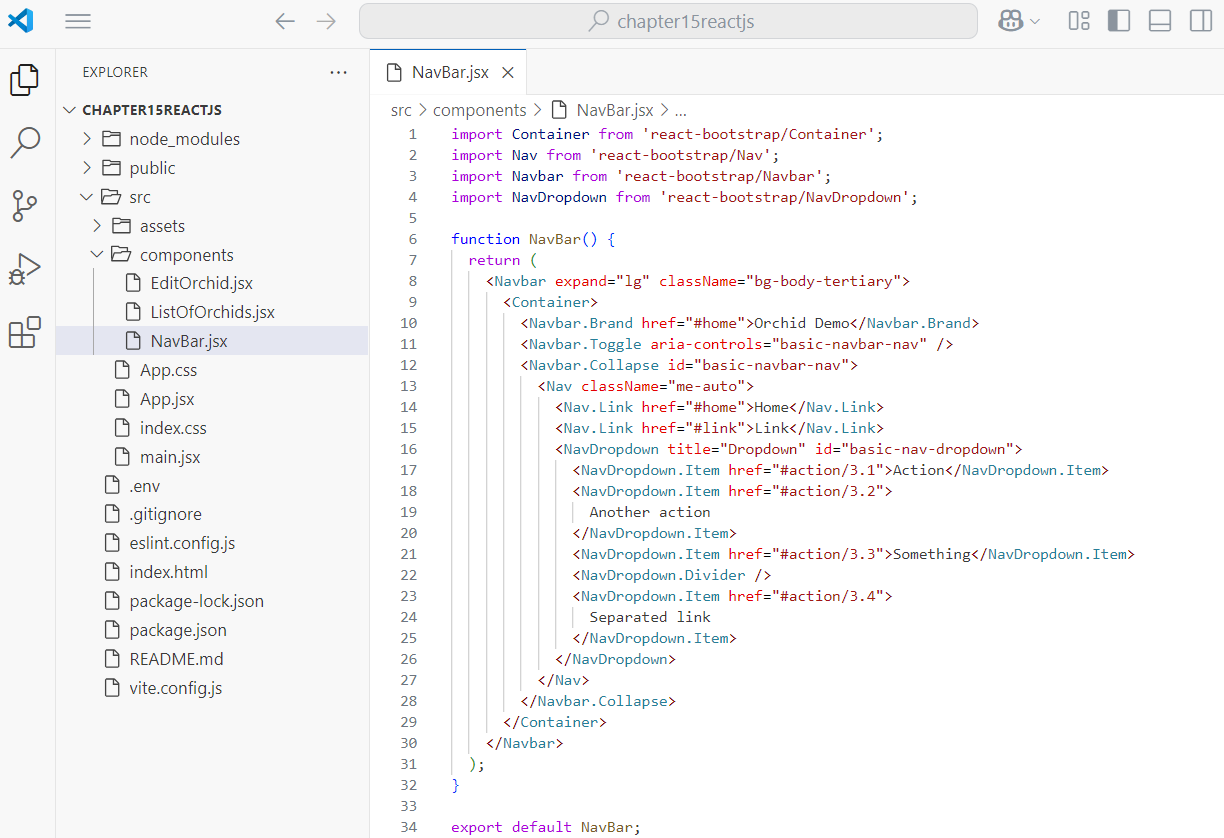
****

**Step 02.** Change the root component App.jsx

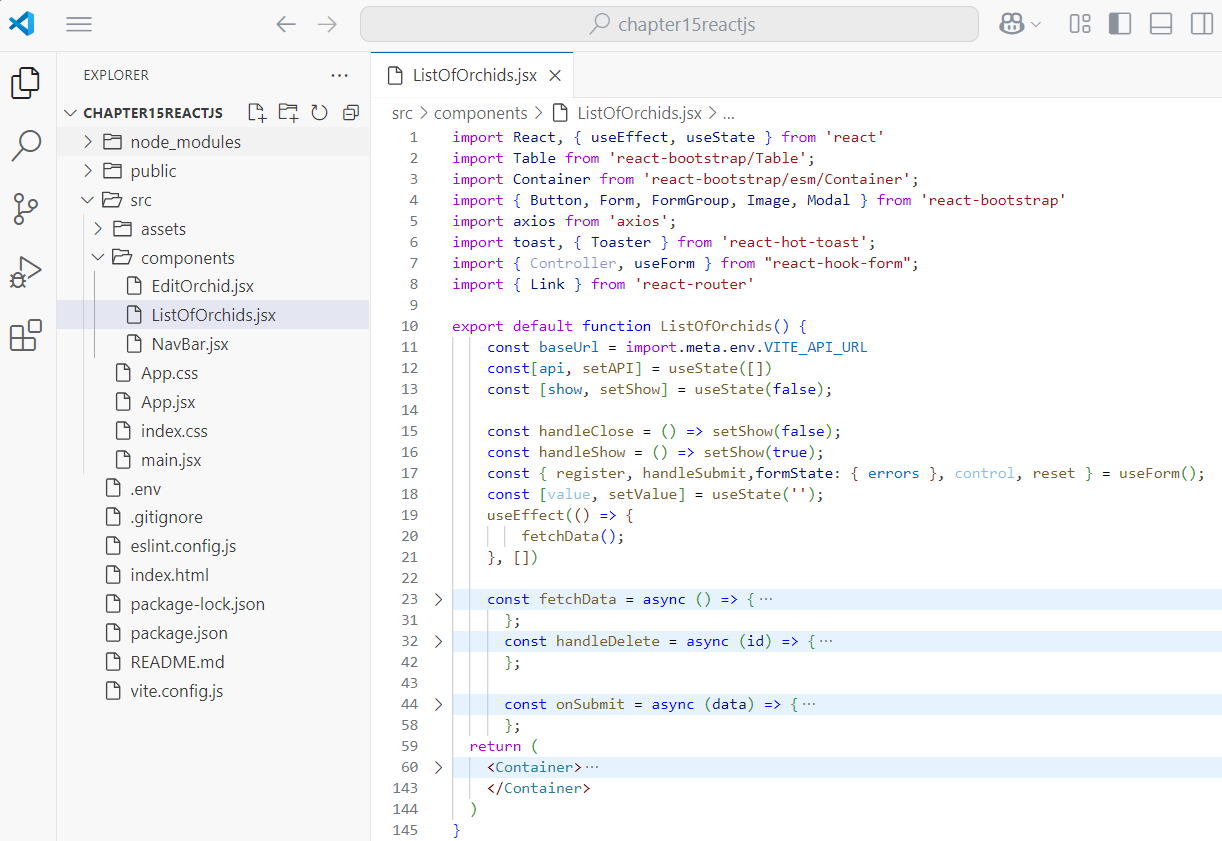
* App.jsx is the root component of React application.
* App.jsx acts as the container for all other components and defines the overall structure of user interface.
* App.jsx imports and renders other components, creating a hierarchy of components that form the application.

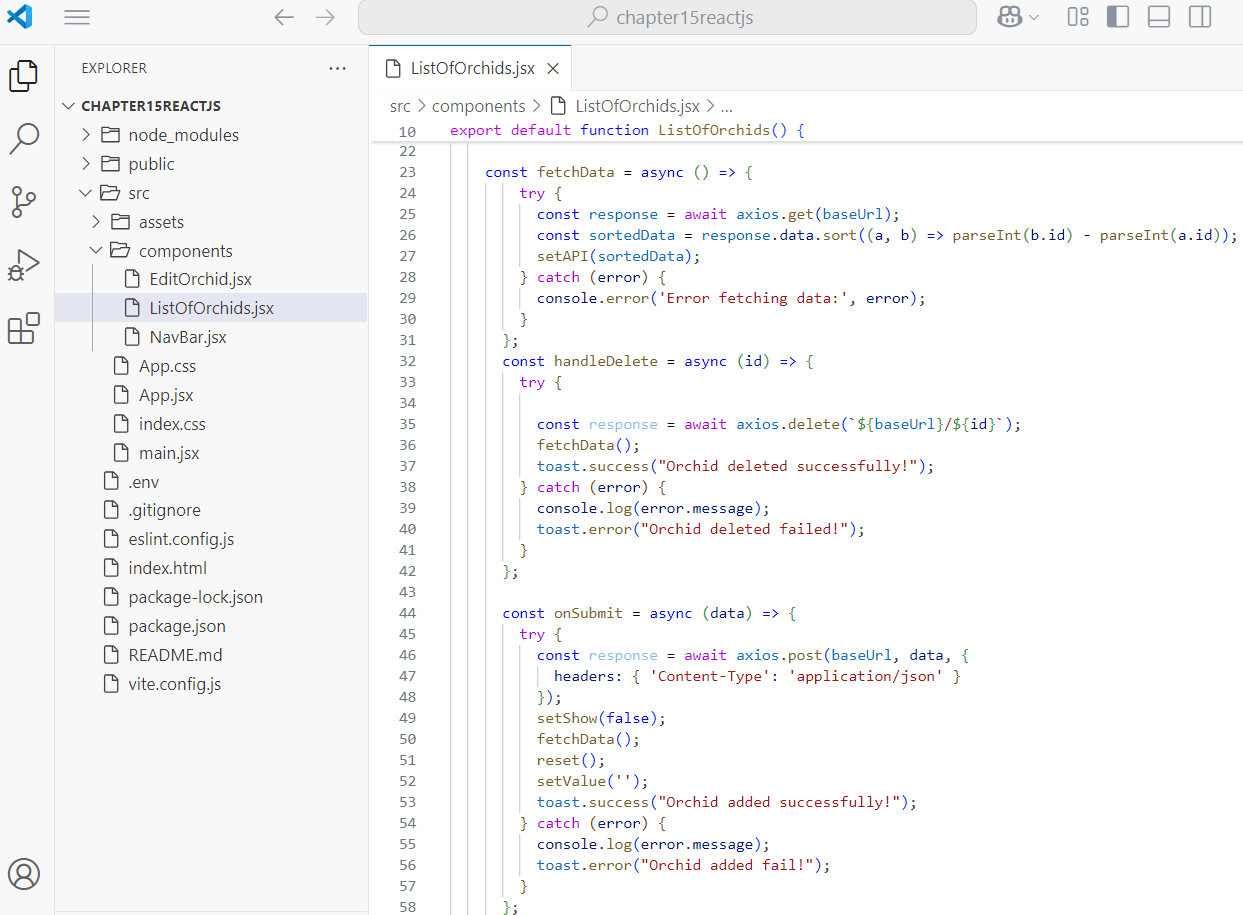


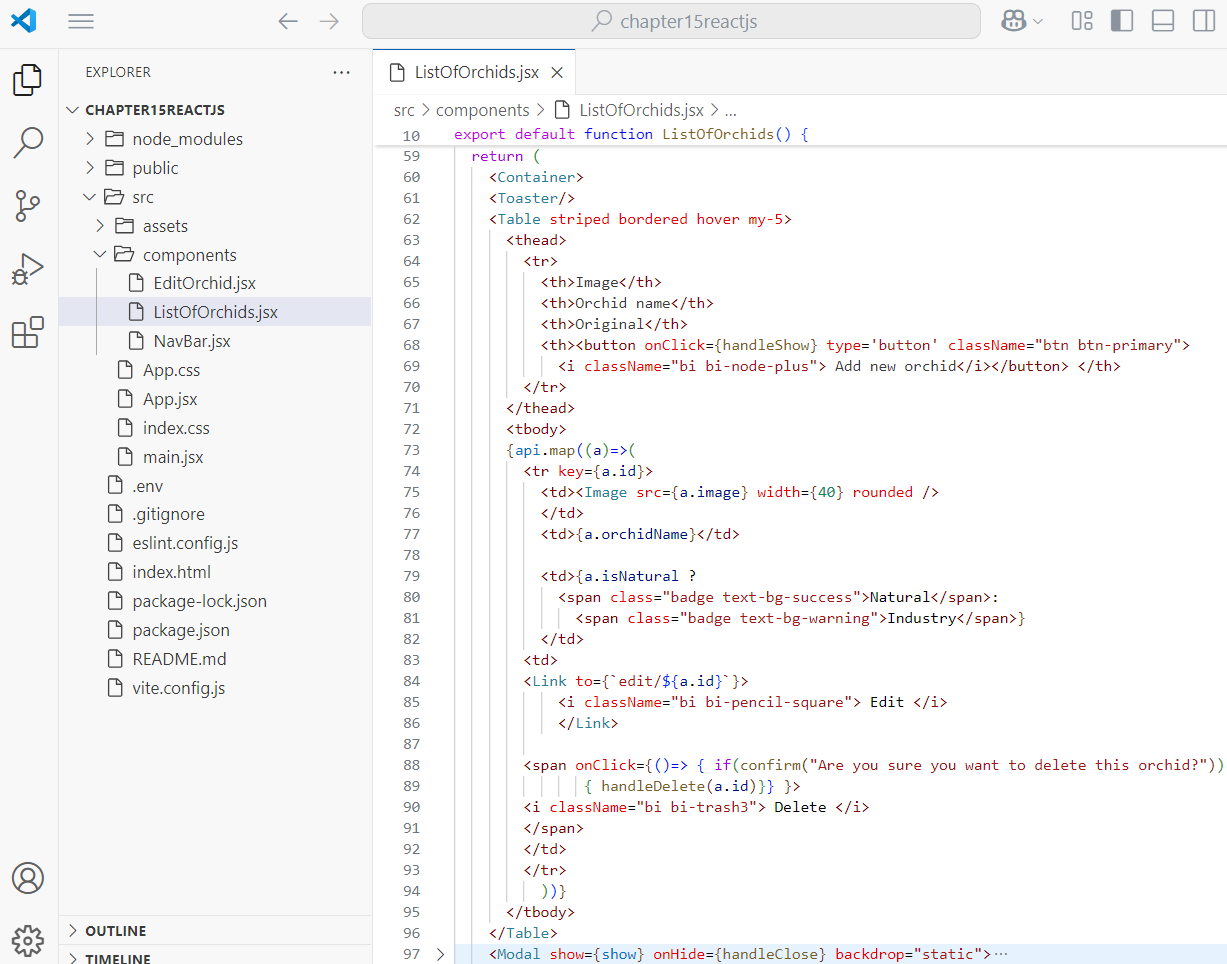
**Step 03.** Work with navigation bar NavBar.jsx

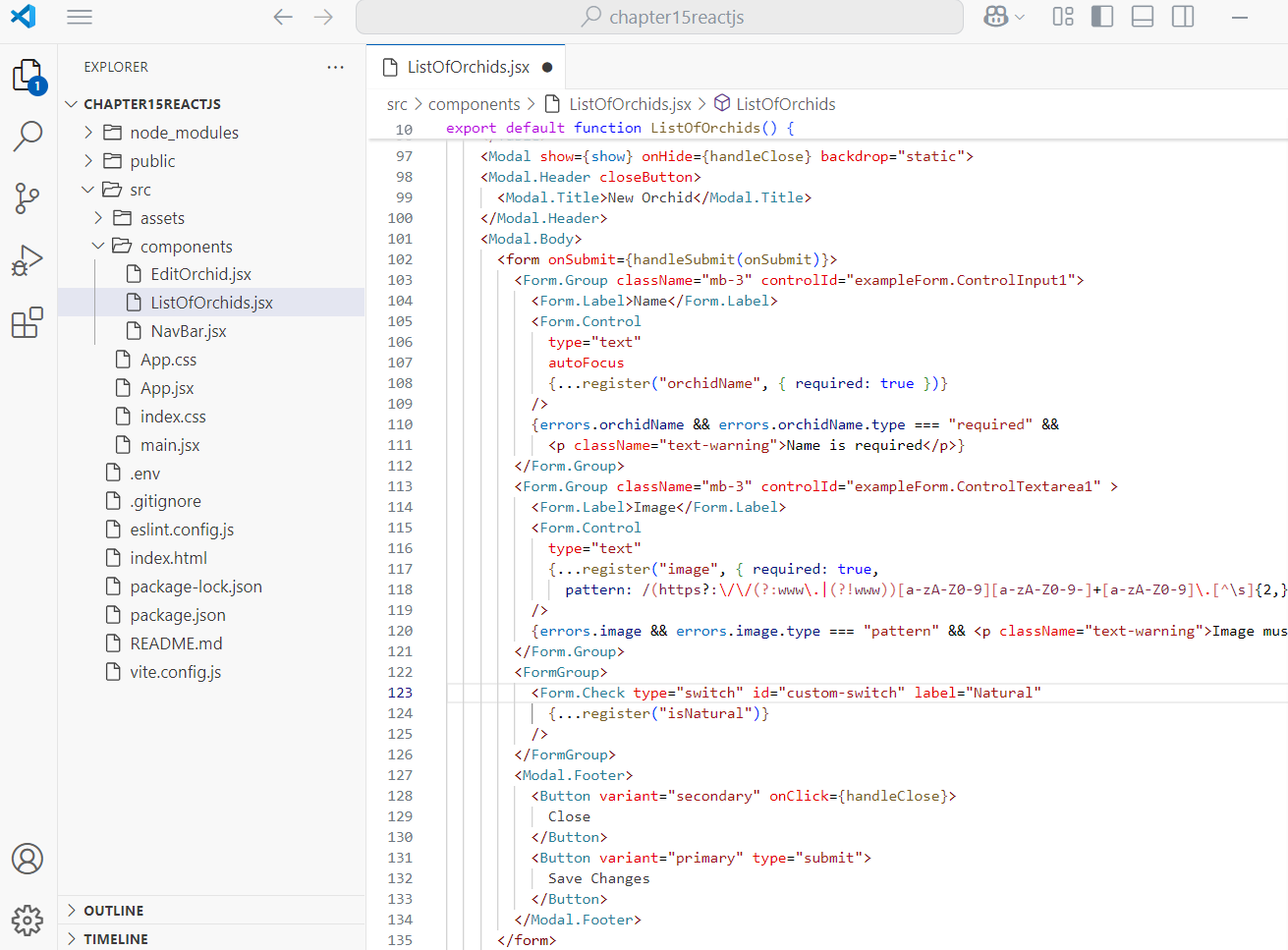


**Step 04.** Work with ListOfOrchids.jsx

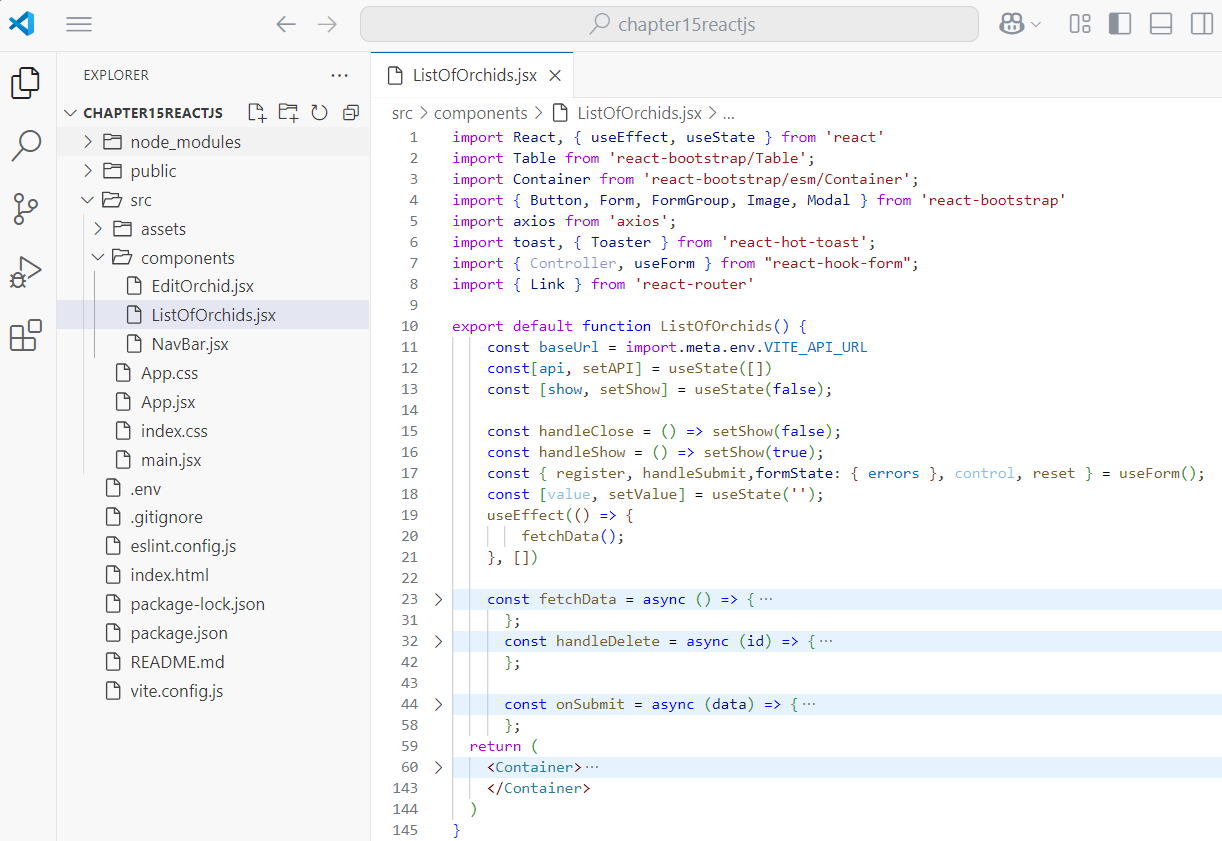
****

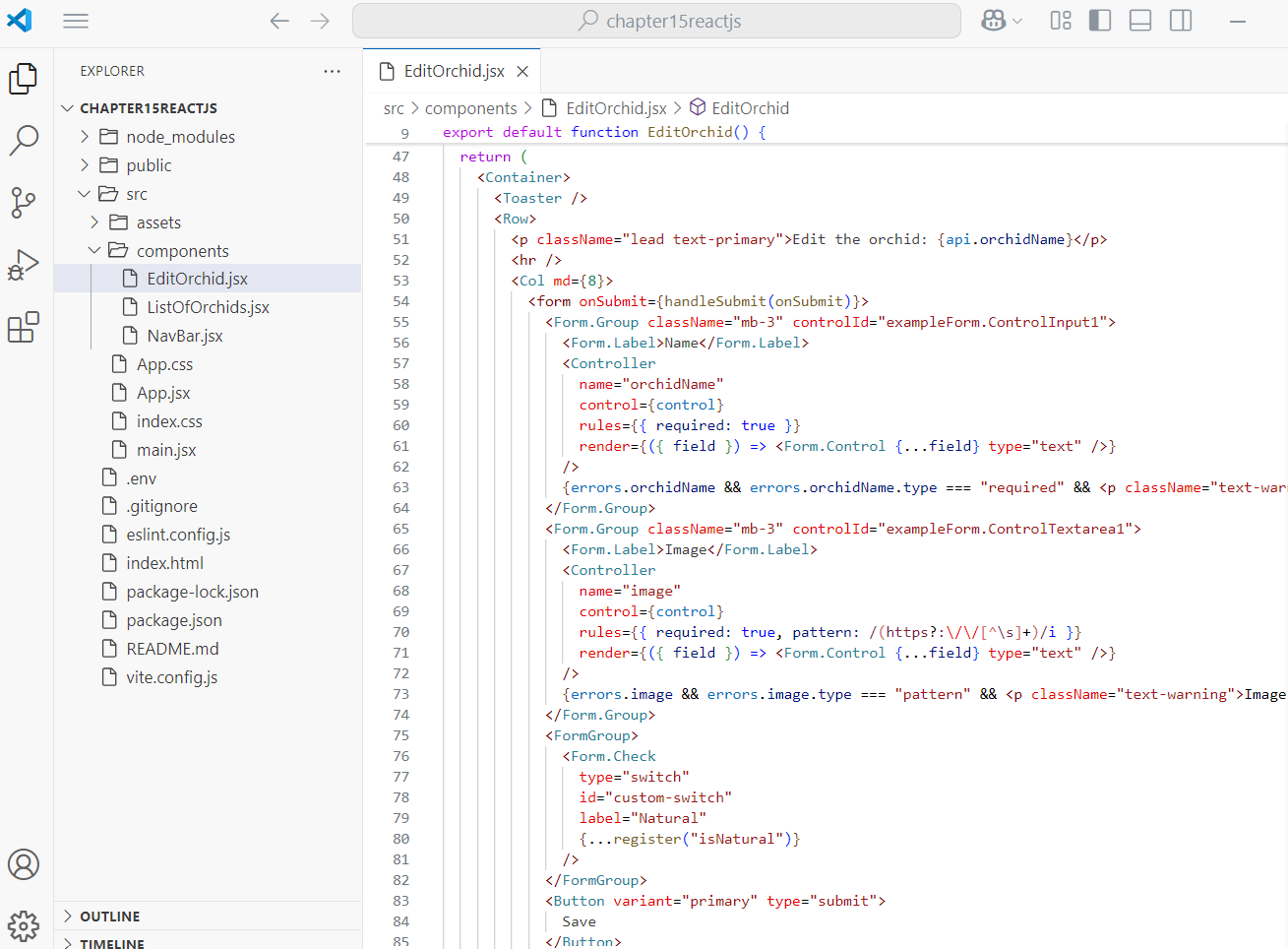
****

****

****

**Step 05.** Work with EditOrchid.jsx





**Step 06.** Run the ReactJS application

* Start the development server: npm run dev
* Open browser to http://localhost:5173/

