

Spring Boot Computer Equipment Web Store
Developed Using Spring Boot Rest API & React JS UI

By

Soner Can Zencirkiran

June 2022

ABSTRACT

This project is an e-commerce website for computer equipment such as headphones, mouses & keyboards (actually you can put any item if you want to). As an admin, you can add, view, update, and delete products, and categories (CRUD operations). For the back-end of the website, I used Spring Boot REST API, for the front-end of the website I used React JS, and to connect the back-end and front-end I used Axios.

USED TECHNOLOGIES

Back-end: Spring (Boot, Data, ModelMapper), JPA/Hibernate, Lombok, node.js

Front-end: React.js(Router-DOM), Axios, BootStrap 4, CSS

Database: MySQL, MySQL WorkBench

API Platform: POSTMAN

IDE: IntelliJ IDEA Ultimate by JetBrains, Visual Studio Code by Microsoft Corporation

Server Build: Maven

Client Build: npm

Spring Boot BootStrapper: Spring Initializr

DETAILS ABOUT TECHNOLOGIES

Spring Boot: Spring Boot is an open-source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase.

Spring Data: Spring Data is a project driven by Spring that aims at providing a consistent data access layer for various data stores, right from relational to NoSQL databases.

Spring ModelMapper: ModelMapper's Spring integration allows for the provisioning of destination objects to be delegated to a Spring BeanFactory during the mapping process.

JPA: The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java objects and relational databases. JPA acts as a bridge between object-oriented domain models and relational database systems.

Hibernate: Hibernate Object-Relational Mapping(ORM) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.

Lombok: The project Lombok is a popular and widely used Java library that is used to minimize or remove the boilerplate code. Never write another getter or equals method again,

with one annotation your class has a fully-featured builder, Automate your logging variables, and much more.

React JS: React is a free and open-source front-end JavaScript library for building user interfaces based on User Interface(UI) components.

React Router-DOM: React Router DOM is an npm package that enables you to implement dynamic routing in a web app. It allows you to display pages and allow users to navigate them. It is a fully-featured client and server-side routing library for React.

Axios: In ReactJS, Axios is a library that serves to create HTTP requests that are present externally. It is evident from the fact that we may sometimes in React applications need to get data from an external source. It is quite difficult to fetch such data so that they can be normally shown on the website.

Bootstrap: Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS, and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

CSS: Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

MySQL & MySQL WB: MySQL is an open-source relational database management system developed by Oracle that is based on structured query language (SQL). MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation, and maintenance into a single integrated development environment for the MySQL database system.

POSTMAN: Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster. With Postman you can test your endpoints.

IntelliJ IDEA: IntelliJ IDEA is an Integrated Development Environment (IDE) for JVM languages designed to maximize developer productivity.

Visual Studio Code: Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux, and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Maven: Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. The tool provides allows developers to build and document the lifecycle framework.

Npm: npm is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js. It consists of a command-line client, also called npm, and an online database of public and paid-for private packages called the npm registry.

Spring Initializr: The Spring Initializr is ultimately a web application that can generate a Spring Boot project structure for you. It doesn't generate any application code, but it will give

you a basic project structure and either a Maven or a Gradle build specification to build your code with.

Node.js: Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

USED TERMS

JSX: JSX is a React extension to the JavaScript language syntax which provides a way to structure component rendering using syntax familiar to many developers. It is similar in appearance to HTML.

HTML: The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

URL: A Uniform Resource Locator, colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier, although many people use the two terms interchangeably.

SQL: SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

EndPoints: An API endpoint is a point at which an API -- the code that allows two software programs to communicate with each other -- connects with the software program. APIs work by sending requests for information from a web application or web server and receiving a response.

JavaScript: JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

Query Parameters: Query parameters are a defined set of parameters attached to the end of a URL. They are extensions of the URL that are used to help define specific content or actions based on the data being passed.

DataBase: In computing, a database is an organized collection of data stored and accessed electronically. Small databases can be stored on a file system, while large databases are hosted on computer clusters or cloud storage.

TABLE OF CONTENTS

| | |
|---|----------------|
| ABSTRACT | 2 |
| USED TECHNOLOGIES | 3 |
| DETAILS ABOUT TECHNOLOGIES | 4, 5, 6 |
| USED TERMS | 7 |
| WORKING PROCESSES | 8,9 |
| 1 Creating a Spring Boot Project With Spring Initilazr | 10 |
| 1.1 Pom.xml | 10 |
| 1.2 Application.properties File | |
| 11 | |
| 2 Creating MySQL DataBase | 11 |
| 2.1 Installing MySQL | 11 |
| 3 Connecting Spring Boot Application to MySQL Database | 12 |
| 3.1 Starting Application for the First Time | 12 |
| 4 Layers | 13 |
| 4.1 Entity | 13 |
| 4.2 Repository | 14 |

| | |
|---------------------------------------|-----------|
| | 8 |
| 4.3 Service | 14 |
| 4.4 DTO | 15 |
| 4.4 Mapper | 15 |
| 4.6 Controller | 15 |
| 5 PostMan | 16 |
| 5.1 Testing Endpoints | 16 |
| 6 Starting a front-end project | 17 |
| 6.1 React file structure | 17 |
| 6.2 Defining source address to react | 17 |
| 6.3 Downloading BootStrap4 | 18 |
| 6.3 Creating React Service | 18 |
| 6.4 React Components | 18 |
| 6.5 React Hooks | 19 |
| 7. RESULTS AND DISCUSSION | 20 |
| 8. CONCLUSION | 20 |
| RESUME | |

1 Creating A Spring Boot Project With Spring Initilazr

With Spring Initilazr I generated my spring boot project quickly, all I had to do is select my projects' features, dependency, and click generate.

The screenshot shows the Spring Initilazr web interface with the following configuration options:

- Project:**
 - ☒ Maven Project
 - ☐ Gradle Project
- Language:**
 - ☒ Java
 - ☐ Kotlin
 - ☐ Groovy
- Spring Boot:**
 - ☐ 3.0.0 (SNAPSHOT)
 - ☐ 3.0.0 (M3)
 - ☐ 2.7.1 (SNAPSHOT)
 - ☒ 2.7.0
 - ☐ 2.6.9 (SNAPSHOT)
 - ☐ 2.6.8
- Project Metadata:**
 - Group:
 - Artifact:
 - Name:
 - Description:
 - Package name:
 - Packaging: ☒ Jar ☐ War
 - Java: ☐ 18 ☒ 17 ☐ 11 ☐ 8
- Dependencies:**
 - Spring Boot DevTools** (DEVELOPER TOOLS): Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
 - Lombok** (DEVELOPER TOOLS): Java annotation library which helps to reduce boilerplate code.
 - Spring Web** (WEB): Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Data JPA** (SQL): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
 - MySQL Driver** (SQL): MySQL JDBC and R2DBC driver.

Buttons at the bottom: GENERATE (CTRL + G), EXPLORE (CTRL + SPACE), SHARE...

Now, I have a spring boot project. After generating the spring boot project I had to check the file structure of the spring bootstrap to be able to use it.

1.2 Pom.xml

A Spring boot project has two important files to work on at the beginning one of them is in the target file called "pom.xml" which is a maven-based service that keeps all information about your dependencies and transmits functions, data etc. from dependencies to your project. Every dependency is kept in the XML file as shown;

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
```

1.3 Application.properties file

Another Important file is in the resources file called “application.properties”. With this file you can configure your application, for example, you can change your application server.port, and this file is where you connect your application to a database or databases. And all of the packages and classes that you will create will go into a package called (in this project it is defined with default name by spring initilazr) “demo (com.example.demo)”.

In the demo package there is an executable java class called “DemoApplication”, this is the heart of the project because it has the main method. At the start if you try to run “DemoApplication” it occurs an error is shown;

```
*****
APPLICATION FAILED TO START
*****
Description:
Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.
Reason: Failed to determine a suitable driver class
```

The reason of this error is Spring Application can't find any database to connect to. Now I have to create a database (For this project I will use a MySQL server) and connect it to my spring boot application within “application.properties” file.

2 Creating MySQL DataBase

Firstly, I installed MySQL Installer from official MySQL web page. In MySQL Installer there is an Available products part where you select MySQL server version and MySQL applications. For this project, I needed latest MySQL Server which is 8.0 and from applications part I needed MySQL WorkBench and MySQL Shell. After selection part I chose my server.port (default: 3306) and created a MySQL Account with root name and password and finished installation. Now I have a MySQL server and it automatically creates one database with my root name. Now I got to connect my spring boot application to this database.

3 Connecting Spring Application to MySQL

As I said before to start the “DemoApplication” I have to connect my Spring boot application to a database within “application.properties” file. To do that you have to define your MySQL server.port url for Spring data source and mine is localhost:3306 after that you have to give informations of owner of the server such as username and password. Then I had to mention that this is a MySQL server to program able to use MySQL driver dependency that we added

```
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce?useSSL=false&allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=Rasyss01*
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
logging.level.org.springframework.web=DEBUG
```

in our pom.xml. And that is all it is just a few line of code as shown;

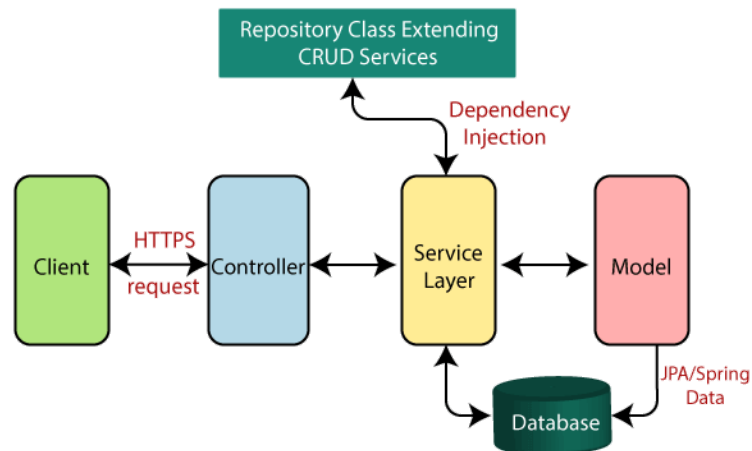
3.1 Starting Application for the First Time

After connecting to the database I am able to run the project. After execution in console there is line says “Tomcat started on port(s): 8080 (http) with context path “ (8080 is the default port for spring boot applications). When I visit localhost:8080 it give an “Whitelabel Error Page” because we have not defined any endpoint to this address.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

4 Layers



Every Spring boot application has these layers as shown above.

4.1 Entity

Model(Entity) layer is where we define our entities such as product, category, order etc. When we are creating an entity we use 2 common annotation called @Entity and @Table. With @Entity annotation, we are mentioning that this is an entity class and with @Table annotation, we are mentioning that every variable defined in this class is should be automatically created and attached to the table we define (we give the name of the table).

This is how we define it and get it on database;

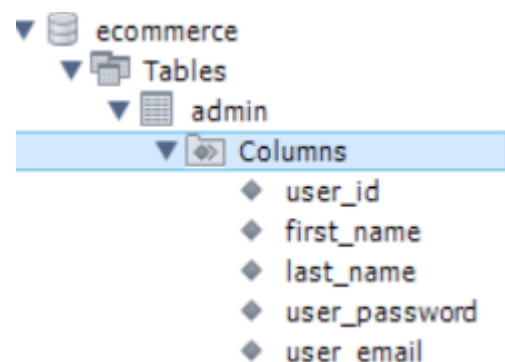
```

@Entity
@Data
@Table(name = "admin")
public class Admin {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long user_id;

    @Column(
        name = "first_name", nullable = false
    )
    private String firstName;
}
  
```

4.2



Repository

In Repository layer first thing we do is use `@Repository` annotation to mention spring that this is a repository. we create a repository interface for every entity and extend it to another interface called `JpaRepository`. In `JpaRepository` there are functions such as `findAll`, `findById`, `save` etc. to be able to control database tables from application. After creating entity's repository we pass its repository to the Service Layer where we will use functions from the `JpaRepository`. This is how we create a repository for an entity;

```
public interface AdminRepository extends JpaRepository<Admin, Long> {
}
```

4.3 Service Layer

In Service Layer we use `@Service` annotation to mention spring that this is a service. In this layer we define our functions that we will use on entities such as "findAllProducts". As the name suggest this function should return every product that we have on our database. For that we call our "ProductRepository" and call "findAll" function that defined in `JpaRepository`. It is shown below;

```
private final ProductRepository productRepository;

@Override
public List<Product> findAllProducts(){
    return productRepository.findAll();
}
```

In Service Layer you do not have to use defined functions from `JpaRepository` you can also create custom functions.

4.4 DTO

In DTO Layer we are defining that which of the features of an entity will be send between layers or what will be printed to the screen of a client such as userName, userEmail etc. You can think it like you are roaming on a social media page on a computer and you click on another users profile and opened inspect with your browser and saw this user password, id, number etc. This could be the situation if we did not used DTO, with DTO we avoid private informations to be transmit between layers. But to do that we need mapper.

4.5 Mapper

This is where we convert an entity to a response. If we had 3 variable in our entity such as userId, userName, and password and we had 1 variable in our DTO response such as userName. This is where it only takes userName because it is the one defined in response and leaves other variables. And in mapper classes we use @Component annotation to mention that this is a mapper class.

4.6 Controller Layer

In this layer we define our endpoints. As I mention before I said in service we define our functions for our entities. In this layer we call his functions after we send them to mapper. Let's say we want to use getAllProducts function and make it return our products in controller we define an endpoint and call this function inside of it. It is used as it shown below;

```
@GetMapping("/products")
public ResponseEntity<List<ProductResponse>> getAllProducts(){
    return ResponseEntity.ok(productMapper.findAllProducts());
}
```

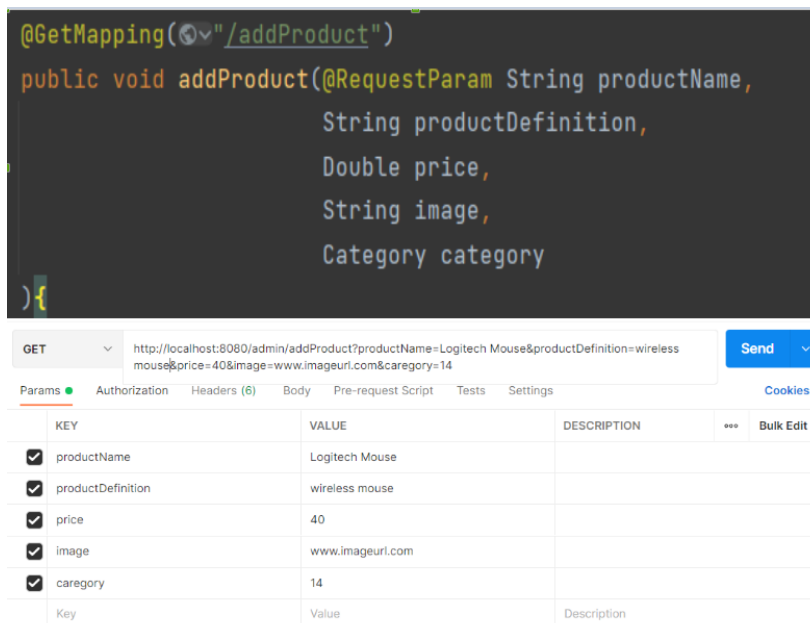
In this layer we use @Controller and @RequestMapping annotation and give it a base url if we want to.

5 POSTMAN

On a website every page, every Id, every function etc. change the url of the page. For example, users shown with “test:1000/users” and a specific user shown with “test:1000/users/45”. So with Postman we can check our endpoints to see if they are working or not.

5.1 Testing Endpoints

In spring boot there are two annotations called `@RequestParam` and `@PathVariable`. With `@RequestParam` we use to extract query parameters with “?” and we mention that I want to give every parameter one by one in my url such as;



```
@GetMapping("/addProduct")
public void addProduct(@RequestParam String productName,
                       String productDefinition,
                       Double price,
                       String image,
                       Category category)
```

The screenshot shows a Postman interface for a GET request to `http://localhost:8080/admin/addProduct?productName=Logitech Mouse&productDefinition=wireless mouse&price=40&image=www.imageurl.com&category=14`. The 'Params' tab is active, displaying a table of query parameters.

| KEY | VALUE | DESCRIPTION |
|---|------------------|-------------|
| <input checked="" type="checkbox"/> productName | Logitech Mouse | |
| <input checked="" type="checkbox"/> productDefinition | wireless mouse | |
| <input checked="" type="checkbox"/> price | 40 | |
| <input checked="" type="checkbox"/> image | www.imageurl.com | |
| <input checked="" type="checkbox"/> category | 14 | |
| Key | Value | Description |

With `@PathVariable` we extract data from url. It is a better option when there are crypted parameters and it is better for RESTful web services that have long urls such as “test:1000/order/024852365623”.

In PostMan you can select that if you want to use GET method that you defined in Spring with @GetMapping or DELETE etc. After the test all endpoints works fine now it is time to start front-end.

6 Starting a front-end project

To create a react project you need node.js installed on your computer. Other thing you need an IDE, most of the software developers recommends to write your back-end and front-end on different IDEs. In this situation Visual Studio Code will be mine front-end IDE. After that you need to download react and react-snippets (to write code easily) from Visual Studio Code extensions part. Now we are ready to create a react project, all you got to do is go to the terminal and type these lines;

1- `npx create-react-app react-frontend`

2- `cd react-frontend`

3- `npm start`

It will automatically get opened on your default browser.

6.2 React File Structure

After creating react there are 3 important file/package that we will work on. One of them is index.html. This is actually where all of our real a specific URL. You can add your bootstrap, font-syle addresses etc. here. There is a package called "src" this is all of our components go. Under the "src" file there is "app.js", this is where we define our "routes" and call our components."app.js" is the second important file in our project. The other important file is "index.css" this is where we write our styling code.

6.2 Defining source address to React

React should get information from back-end, to be able to do that we need to download axios and define where react will get information. We need to give it our back-end server.port (Base.URL) address with axios.create method. To download axios all you got to do is go to the terminal and type;

`npm install axios`

And it will be automatically added to your “package.json” where all your dependencies in.

6.3 Downloading Bootstrap4

To make our website look better we need to write css but sometimes it takes a lot of time, so that is why developers created bootstrap for us. Bootstrap is a CSS framework that allows you to design your elements just with classname attribute. For example you have a button and you want to make it look better, so all you gotta do go into button tag and type `className=“btn”` and it will automatically get its css from bootstrap. To download bootstrap all you gotta do go into terminal and type this line;

`npm install bootstrap (you can choose version with @“version”)`

And it will be shown in your package.json file

6.4 React Components

React Components are codes that is not dependent to any class, you can call them anytime you want again and again because they are reusable. But they are not same with JavaScript functions because in return they return HTML (If a JavaScript file returns HTML it is called JSX). There are two types of React Components. One of them is Class Component, a Class Components should be extended to `React.Component` class. And it has a method that returns HTML called “render” method. In Class Components you cant use React Hooks. React Hooks allow you to use state and other React features without writing a class. Other Component type is Function Component, in Function Component you can use React Hooks and you dont need a “render” method because it is a function it can return HTML directly. Best thing about components is that you can call as much as component in one page.

6.5 React Hooks

As I mentioned before it is a feature that helps you to use react fetures without creating class. There are a lot of React Hooks but most importants are “useState”, “useNavigate” and “useParams”. useState allows you to have state variables, you can call this variables and change them in a function.

UseNavigate allows you to navigate from an URL to another easily, all you got to do is create a function that navigates user to another URL and add it on a button. useParams allows you to get parameter from URL address. For example, you want to delete a post on social media, this post is open at “test:1000/posts/39” useParams takes 39 from your URL and adds it on delete function on back-end.

7. RESULTS AND DISCUSSION

This project has Spring Boot REST API back-end and React.JS front-end. On back-end it has multi-layered structure of spring boot. On front-end it has react. In front-end I used a lot of component to avoid website to be slow. My application has 5 entity and they are Admin, Category, Order, Product and User. For most of them I wrote service that has CRUD operations, mapping, repository, controller and DTO. Every information about this classes given in other pages.

8. CONCLUSION

In this project we are able to buy a product that is available on website. User can select a category or can review all of the products. After buying a product (with credit card (No API just prototype)) it is saved on database as order. As admin you can add, update, view and delete a product and category. This project is not at the same level with other professional e commerce website because it is made by one person and in short time. But its have the base structure of a professional e commerce website you can see it as a prototype. Yet, you can still buy products and get them. With this project I understood the spring boot structure so well and how to use react efficiently, what is react hooks and how to work with components.

RESUME

Soner Can Zencirkıran was born in Novermber 12, 1998, in Adana/Yüreğir. He is 4th grade Computer Enginnering student at Cukurova University. He is interested in web applications

that are designed with Java and JavaScripts. He learned Java, JavaScript, Spring boot, C, C++, R, Python,, HTML, CSS, PHP, Bootstrap, SQL in his studying period.