Shirts Shop Documentation

Overview

Shirts Shop is a web application built using Java Spring, Angular, and PostgreSQL. It provides an online platform for users to create accounts, log in, browse and search for products, view product details, edit their profile information, change their passwords, and add products to their cart. The application has a secure backend API that requires authentication using JWT tokens. User data is protected, and users can only access their own data unless they have admin privileges. The passwords are securely hashed using Bcrypt and stored in the database. The frontend Angular application stores the access token in local storage and includes it in requests that require authentication. It provides a user-friendly interface for users. It features a simple design that makes it easier for users to browse and interact with the application. Additionally, it includes alerts to provide feedback to the user regarding the status of their requests.

Technologies

- Java
- Spring Framework
- JSON Web Token (JWT)
- Angular
- PostgreSQL
- Maven
- Docker

Installation

1. Clone the repository from GitHub:
   https://github.com/KirolosSawiris/e-commerce-clothing-store-web-app.git

2. Backend Setup:
   - Install Java Development Kit (JDK) 17.
   - Install IntelliJ IDEA.
   - Install Docker and ensure it's running.
   - Run a PostgreSQL container from the official Image.
   - Update the backend application's configuration file to connect to the Dockerized PostgreSQL database.
   - Build and run the backend API using IntelliJ.

3. Frontend Setup:
   - Navigate to the frontend application directory.
   - Make sure the Node.js is already installed in your system.
   - Install the required dependencies by running the following command in the directory terminal: **npm install**.
   - Build the frontend application by running **npm run start**.

Database

The application uses PostgreSQL as the database management system.

Database Schema

1. **Users Table:**
   - **The** Users **table stores information about registered users in the application.**
   - **Each user has a unique** id **as the primary key.**
   - **The** username **column stores the username of the user, which is required and must be unique.**
   - **The** firstName **and** lastName **columns store the first and last names of the user, respectively.**

- **The** email **column stores the email address of the user, which is required and must be unique.**
- **The** password **column stores the hashed password of the user.**
- **The** address**,** country**,** region**, and** postcode **columns store the user's address information.**

2. **Carts Table:**
   - **The** Carts **table represents the shopping carts associated with users.**
   - **Each cart has a unique** id **as the primary key.**
   - **The** user_id **column references the** id **of the user in the** Users **table, establishing a one-to-one relationship between a user and a cart.**
   - **The** cartTotal **column stores the total value of items in the cart.**
3. **CartItems Table:**
   - **The** CartItems **table stores the items added to the shopping cart.**
   - **Each cart item has a unique** id **as the primary key.**
   - **The** cart_id **column references the** id **of the cart in the** Carts **table, establishing a one-to-many relationship between a cart and its items.**
   - **The** product_id **column references the** id **of the product in the** Products **table, indicating which product is associated with the cart item.**
   - **The** quantity **column stores the quantity of the product in the cart item.**
4. **Products Table:**
   - **The** Products **table represents the shirt products available in the application.**
   - **Each product has a unique** id **as the primary key.**
   - **The** title **column stores the title or name of the product.**
   - **The** price **column stores the price of the product.**
   - **The** quantity **column stores the available quantity of the product.**
   - **The** color, size, and description **columns store the product details.**
   - **The** image **column stores the URL or path to the image of the product.**
   - **The** category_id **column references the** id **of the category in the** Categories **table, establishing a one-to-many relationship between a category and its products.**
5. **Authorities Table (Roles):**
   - **The** Authorities **table stores the roles or authorities in the application.**
   - **Each role has a unique** id **as the primary key.**
   - **The** `name` **column stores the name of the role.**
6. **Categories Table:**
   - **The** `Categories` **table represents the categories for grouping shirt products.**
   - **Each category has a unique** `id` **as the primary key.**
   - **The** `name` **column stores the name of the category.**

Backend Application

The backend has the model classes that have been used to generate the database schema. Also, it has its JPA repositories and controllers.

Controllers

Each entity has its API URL and a controller class.
The controllers provide the Rest API for each entity, each control class has its GET, POST, and PUT functions.
The user controller also has a security layer, it makes sure that the user who requested the user is the same as the user authenticated through the access token, and if yes it provides the requested data if not it throws an Access denied exception.

Services

We have the UserService class that for example that encrypts the user password and saves it to the database, we also have CartService that is used to add or remove items from the cart

Security

We use security to make sure that only the user can access his/her own data. So when the user login it performs a filter in the security layer to make user that the username and password match the username and password in the database if yes it sends an access token that the user can use to access and modify his/her own data. The other entities' APIs are locked only for admins.

Backend API

Products

- */api/v1/products/ -* GET to get the products it doesn’t need authentication.
- */api/v1/products/ - POST* Only admins can post products.

Login

- */login/ - POST* You need to post the username and the password in order to get the access token no authentication is needed here.

Register

- */api/v1/users/Register/ - POST* To create a new user in the database, also no authentication is needed here.

User

- */api/v1/users/{username}* - GET to get the user by its username and please note that you need an authentication token for this request. Only the token created for this user will work unless it’s an admin token.
- */api/v1/users/{username} - PUT* to edit the user details and you need authentication for this one as well
- */api/v1/users/addCartItem/{username}- PUT* to add an Item to the user cart and you need authentication for this one as well
- */api/v1/users/removeCartItem/{username} - PUT* o remove an Item from the user cart and you need authentication for this one as well

Frontend Application

The front end is a modern and user-friendly e-commerce platform that allows customers to explore and purchase a wide range of products. Built using Angular. My application provides a seamless shopping experience with robust features such as user authentication, product listings, shopping cart management, profile management, and search between products.

Key Features

Login and Registration: Users can create accounts or login to existing accounts, enabling personalized experiences and order tracking.
Product Listings: Browse and search through an extensive collection of products with detailed descriptions, images, and pricing information.
Product Details: Get in-depth information about a specific product, including specifications, customer reviews, and related items.
Shopping Cart: Add products to a cart, and proceed to checkout for a streamlined purchasing process.
Profile Management: Users can update their personal information, manage shipping addresses, and change passwords.

Search: users can search between the products.

Architecture

Besides the app component, the application has 6 other components and 3 services.

Components

LoginComponent: This component has the login page and holds the username and password in a form and sends the request by calling the API service.
RegisterComponent: This component has the registration info and sends them by requesting the apiService as well.
CartComponent: This component holds the user's cart information and sends the requested order like adding an item to the cart or removing the item from the cart and that is done using the authService.
User-profileComponent: This component holds and shows user information and makes the user able to change their data after confirming the password.

ProductComponent: This Component makes the user able to see the product details and also add this product to the cart with any quantity.
HomeComponent: This Component shows the list of products to the user and makes them able to add any of them to the cart.

Services

ApiService: this Service is responsible for any API request.
AuthServie: this Service is responsible for the request that needs authentication to perform.
FilterService: this Service filters the products if the user enters a text in the search and clicked search.

Summary

The Shirts Shop is a web application built using Java Spring, Angular, and PostgreSQL. It provides an online platform for users to browse and purchase shirt products. The application features a secure backend API that requires authentication using JWT tokens. Users can create accounts, log in, and manage their profile information. The frontend UI has a simple design. It supports features such as adding products to the cart and viewing product details. With its intuitive interface and secure authentication mechanism, the Shirts Shop offers users a seamless and secure shopping experience.