Full-stack Web Shop
MySQL, Spring Boot, Vue.js

Introduction

This documentation provides a comprehensive overview of the "Bike X Optimum" web shop, a collaborative project developed by a team of three students. The web shop is built on the MySQL database management system, utilizing the Spring Boot framework for the backend and Vue.js for the frontend development.

Team members:

- Filip Stanojković – Singidunum University Faculty of Informatics and Computing (Факултет информатике и рачунарства Униврезитета Сингидунум)
- Dimitrije Rankov – University of Belgrade Faculty of Organizational Sciences (Факултет организационих наука Универзитета у Београду)
- Igor Stanković - College of Information and Communication Technology (Висока школа за информационе и комуникационе технологије)

09/06/2025

System Architecture

The system architecture of the web shop is structured as follows:

Frontend Components:

The frontend is built using Vue.js (Vue 2), providing a responsive and dynamic user interface. It supports full CRUD operations for user account management, including registration, account deletion, password updates, and bike purchasing workflows. As of recent, it also features JWT access and refresh tokens, as well as HTTPS.

Backend Components:

The backend is developed with Spring Boot, following a layered architecture that separates concerns into controllers, services, entities, and repositories. JWT tokens are managed using the io.jsonwebtoken library to handle secure authentication and authorization.

Controllers:

Controllers serve as the interface between the frontend and backend logic. They handle incoming HTTPS requests, process business logic via services, and return appropriate responses. In this web shop, controllers manage user authentication and profile actions, bike catalogue retrieval, and transaction handling.

Entities:

Entities model the core business objects of the application and directly map to database tables. Each entity encapsulates relevant attributes and behaviours. Key entities include User, Bike, and Transaction, representing users, available products, and purchase records respectively.

Services:

Services encapsulate the core business rules and workflows. They orchestrate operations such as user authentication, order processing, and inventory management. By isolating business logic here, controllers remain focused on request handling, improving modularity and maintainability.

Repositories:

Repositories abstract data persistence, providing methods to query, insert, update, and delete records from the database. They serve as the data access layer, isolating database operations from business logic. In this project, repositories manage CRUD operations for users, bikes, and transactions.

Database Schema

Overview:

The database contains three main tables: bikes, transactions, and users. These tables store information about available bikes, transaction history, and user accounts, respectively.

Tables:

1. ***bikes:***

- bike_id: INT (Primary key; Unique)
- model: VARCHAR(255) (Name or model of the bike)
- price: DOUBLE (Price of the bike)
- img_path: VARCHAR(255) (Path to the image of the bike)

2. ***transactions:***

- transaction_id: INT (Primary key; Unique)
- buyer: INT (Foreign key referencing the user_id of the buyer)
- product: INT (Foreign key referencing the bike_id of the purchased bike)
- datetime: TIMESTAMP (Date and time of the transaction)

3. ***users:***

- user_id: INT (Primary key; Unique identifier for each user)
- email: VARCHAR(255) (Email address of the user)
- password: VARCHAR(64) (Encrypted password of the user)
- balance: DOUBLE (User's balance or available funds)
- role: VARCHAR(10) (Role of the user; e.g., 'admin', 'user')

Relationships:

The transactions table has foreign key constraints referencing both the users and bikes tables. This ensures referential integrity, i.e., each transaction is associated with valid user and bike IDs. On user deletion, it changes user_id to null.

- The buyer column in the transactions table references the user_id column in the users table, linking each transaction to the respective buyer (@ManyToOne).
- The product column in the transactions table references the bike_id column in the bikes table, indicating the bike purchased in each transaction (@ManyToOne).

API Endpoints

Bike Controller:

- *GET /bikes*

Retrieves a list of all bikes.

- *GET /bike/id/{id}*

Retrieves a bike by its unique identifier.

- *GET /bike/model/{model}*

Retrieves a bike by its model.

Transaction Controller:

- *POST /addTransaction*

Creates a new transaction.

- *GET /transactions*

Retrieves a list of all transactions.

- *GET /transaction/id/{id}*

Retrieves a transaction by its unique identifier.

User Controller:

- *POST /addUser*

Creates a new user.

- *GET /users*

Retrieves a list of all users.

- *GET /user/id/{id}*

Retrieves a user by their unique identifier.

- *GET /user/email/{email}*

Retrieves a user by their email address.

- *PUT /update*

Updates an existing user.

- *DELETE /delete/id/{id}*

Deletes a user by their unique identifier.

- *POST /login*

Accepts user credentials and returns HTTP 200/401.

- *POST /check-password*

Validates user password and returns HTTP 200/404.

- *POST /refresh*

Accepts a refresh token and returns a new access token if valid.

Refer to each respective Controller code for detailed information on request and response formats for each endpoint.

Deployment

GitHub Repository:

https://github.com/44filip/bike-x-optimum

Database

Prerequisites are:
MySQL Server (latest)
MySQL Workbench (latest)

Import the data found in /db_dump to your relational database of choice (project choice is MySQL).

Frontend

Prerequisites are:
Node.js version 20.x or below (min 8.x)
Python 3.x (latest)

Through a terminal, navigate to .\frontend\ by typing “cd .\frontend” then execute the following commands in order:

```
npm install
npm run serve
```

Backend

Prerequisites are:
JDK version 17.x (latest)
IntelliJ IDEA (latest)

Run BackendApplication.java found in “.\backend\src\main\java\rs\ac\singidunum\backend\BackendApplication.java” either through a Java environment or through the terminal (the execution will vary from system to system).

Browser

Navigate to https://localhost:8080 in the address bar.
Using https may trigger a security warning due to the certificate in place.
If you're having login issues, navigate to https://localhost:8443/login and accept the security risk.
For testing purposes, JWT token expiry will automatically log the user out after thirty seconds.
You may change REFRESH_TOKEN_VALIDITY in “.\backend\src\main\java\rs\ac\singidunum\backend\util\JwtUtil.java”

Thank you for reading through the Bike X Optimum documentation.