

Road Tax Management System REST API - Design and Implementation Report

Introduction

This report discusses the design and implementation of a REST API for a Road Tax Management System. The API is developed using **Spring Boot**, and the backend utilizes a **MySQL database**. The report covers the app design and implementation choices, app architecture, the relationship between components, security configuration, the authentication service using access and refresh tokens, and the business domain.

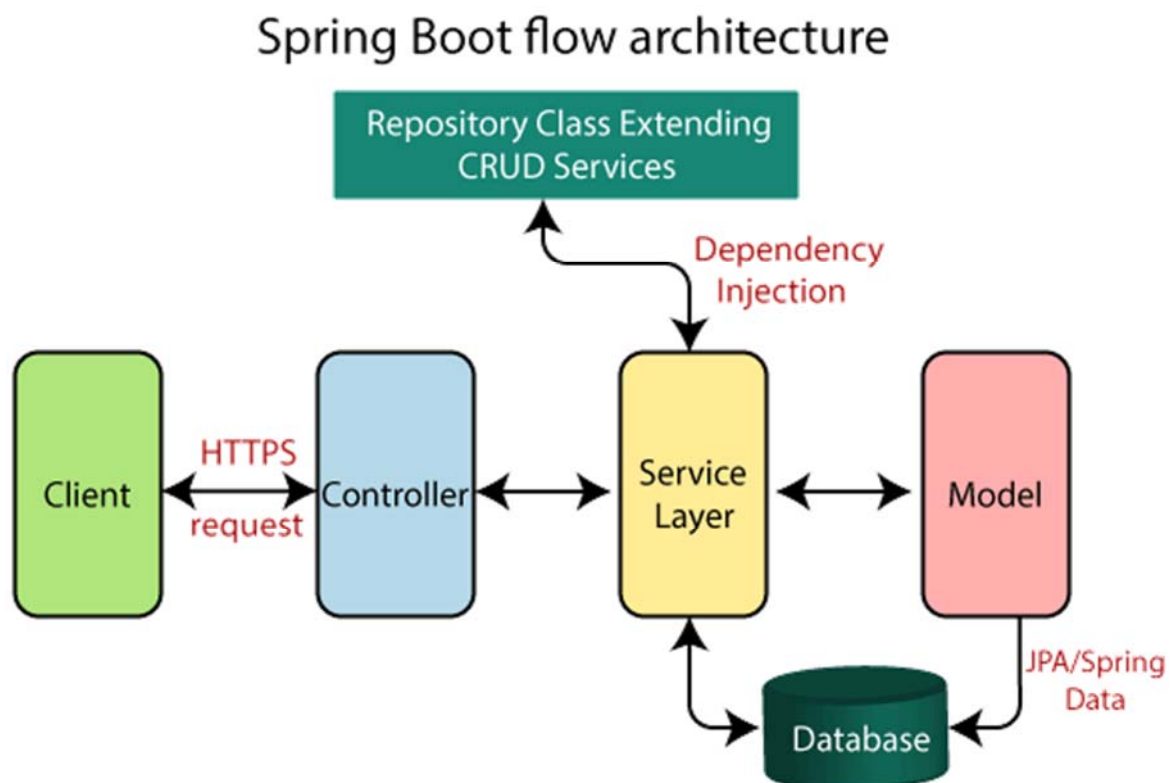
App Design and Implementation Choices

The Road Tax Management System API is designed using the Spring Boot framework, which provides a lightweight and efficient solution for building robust Java-based applications. Spring Boot offers various features such as auto-configuration, embedded servers, and easy dependency management, enabling rapid development and deployment.

The choice of MySQL as the database management system provides a reliable and scalable solution for storing and retrieving data. MySQL offers strong support for ACID transactions and provides a wide range of features to ensure data integrity and security.

App Architecture

The app architecture follows a **layered approach**, separating concerns into different components. The key components include:



1. Controllers: The controllers handle incoming HTTP requests, map them to appropriate methods, and return HTTP responses. They serve as the entry points for the REST API endpoints.

2. Services: The services encapsulate the **business logic** and perform operations on the data. They interact with the repositories to retrieve and manipulate data from the database.

3. Repositories: The repositories provide the interface for performing CRUD operations on the database. They utilize Spring Data JPA, which simplifies database interactions by generating SQL queries based on method names.

4. Entities: The entities represent the data models of the application. They are mapped to database tables using JPA annotations.

Entity Design

1. Car Registration:

The car registration entity represents the details of a registered car. It includes attributes like registration number, make, model, year of manufacture, and owner. The relationship between car registration and car owner is one-to-one, as each car has a single owner.

2. Car Owner:

The car owner entity stores information about the car owner, such as name, address, contact details, and tax information. It has a one-to-many relationship with car registration, as one owner can have multiple cars.

3. Tax Information:

The tax information entity contains details about the tax associated with a car. It includes attributes like tax amount, due date, payment status, and payment history. The relationship between tax information and car registration is one-to-one, as each car has tax information associated with it.

4. Tax Payments:

The tax payments entity represents the payments made by the car owner for the tax. It includes attributes like payment amount, payment date, and reference number. The relationship between tax payments and tax information is one-to-many, as there can be multiple payments for a single tax.

Data Transfer Objects (DTOs)

To facilitate data transfer between the client and the server, Data Transfer Objects (DTOs) are created:

- AuthenticationRequest
- AuthenticationResponse
- RegisterRequest
- CarRegistrationDTO
- TaxInformationDTO
- TaxPaymentDTO

API Documentation

To generate automated documentation for the RESTful APIs, the OpenAPI specification is used. The springfox-swagger2 and springfox-swagger-ui libraries are integrated into the project to generate interactive API documentation.

Relationship Between Components

The components in the Road Tax Management System API are interconnected to achieve the desired functionality. The controllers interact with the services, which in turn utilize the repositories to access the database. The entities define the structure of the data and are persisted in the MySQL database using JPA.

The controllers receive HTTP requests, validate input, and invoke the appropriate service methods. The services implement the business logic, perform data processing, and interact with the repositories to persist or retrieve data. The repositories handle the database operations and provide an abstraction layer for data access.

The relationship between components ensures the separation of concerns and facilitates modularity, making the codebase more maintainable and extensible.

Security Configuration & Authentication Service

To secure the API and manage user authentication, the Road Tax Management System API implements an access token and refresh token-based authentication mechanism.

1. Access Token: Upon successful authentication, the user receives an access token. The access token is a digitally signed JWT (JSON Web Token) containing user details and permissions. It is used to authorize subsequent API requests.

2. Refresh Token: In addition to the access token, the user also receives a refresh token. The refresh token is a long-lived token with a larger expiration time. It is used to obtain a new access token when the current access token expires. The refresh token is securely stored on the client-side, typically as an HTTP-only cookie.

3. Authentication Service: The authentication service handles user authentication and token generation. It verifies user credentials, generates access and refresh tokens, and associates them with the user. It also validates the tokens during subsequent API requests to ensure the user's identity and authorization. Also, there is a logout functionality to remove the access and refresh tokens.

4. Security Configuration: The API employs Spring Security for secure authentication and authorization. The security configuration includes defining security rules, enabling CORS (Cross-Origin Resource Sharing), configuring token-based authentication, and protecting API endpoints.

The access and refresh token-based authentication mechanism provides secure access to the API prevents unauthorized access and ensures a seamless user experience with token refresh functionality.

MySQL Database

The Road Tax Management System API uses a MySQL database to store and retrieve data related to car registrations, car owners, tax information, and tax payments. MySQL is a robust and widely adopted relational database management system known for its performance, scalability, and reliability.

The database schema is designed to represent the relationships between entities. For example, the `CarRegistration` entity has a one-to-one relationship with the `CarOwner` entity, while the `TaxInformation` entity has a one-to-one relationship with the `CarRegistration` entity. The `TaxPayment` entity, on the other hand, has a many-to-one relationship with the `TaxInformation` entity.

The use of MySQL ensures data consistency, provides ACID transaction support, and allows efficient querying and retrieval of data for the Road Tax Management System.

Automated Testing

To ensure the reliability and correctness of the system, automated tests are implemented. The following types of tests are created:

1. Unit Tests: These tests focus on testing individual components, such as service classes, repository classes, and controller methods.

2. Integration Tests: These tests verify the integration between different components of the system, such as testing the interaction between the controller, service, and repository layers.

3. API Tests: These tests validate the functionality of the RESTful APIs by sending HTTP requests and asserting the responses.

Dockerization

To simplify the deployment process, a Dockerfile, and a docker-compose file are created. The Dockerfile includes the necessary steps to package the application as a Docker image and run it in a containerized environment. The docker-compose file will allow running the app besides to the database management system(MySQL). This allows for easy deployment and scalability.

Conclusion

The Road Tax Management System REST API, developed using Spring Boot and MySQL, demonstrates an effective approach to building a robust backend for managing road tax-related operations. The app design and implementation choices, including the layered architecture and separation of concerns, ensure modularity and maintainability. The security configuration and authentication service employing access and refresh tokens enhance the API's security and user experience. The MySQL database provides a reliable and scalable storage solution for the application's data.

The Road Tax Management System API serves as a solid foundation for further enhancements, such as additional business logic, integration with front-end applications, and the implementation of additional features to support road tax management operations.