Technical Documentation for URL Shortener API

Overview

The URL Shortener API is a service designed to take long URLs and convert them into shorter, more manageable links. This API is implemented using Spring Boot and offers endpoints for generating short URLs, retrieving original URLs.
The document outlines the structure, endpoints, and behavior of the API without including code implementation.

API Features

The URL Shortener API includes the following key features:

- **Create Short URLs**: Accepts a long URL and returns an object with shorter version.
- **URL Redirection**: When the short URL is accessed, it redirects to the original long URL.
- **Delete short URL**: Delete short URL object for a given short ID.

Minimum System Requirements

- **Java Version**: Java 11 or higher
- **Spring Boot**: 3.x
- **Database**: PostgreSQL,HR2 for unit testing
- **Maven**: For dependency management
- **JPA**: For ORM (Object-Relational Mapping)
- **SLF4J**: For logging
- **Mockito**: For unit testing

Architecture Overview

The URL Shortener API follows a simple layered architecture:

1. **Controller Layer**: Handles HTTP requests and responses.
2. **Service Layer**: Contains the business logic to create and retrieve URLs.
3. **Repository Layer**: Manages persistence using JPA.
4. **Entity Layer**: Represents the URL object with fields such as longUrl, shortUrl, expiryTime and createdAt

Data Flow

1. **Request**: The client sends a long URL with ttl and shortId as option data to the API.
2. **Processing**: The API generates a unique short URL if you don't provide, stores it in the database, and returns object.
3. **Redirection**: When the client accesses the short URL, the API retrieves the corresponding long URL and redirects to it.
4. **Delete short URL**: if don't need short URL any more you can delete using short ID

Endpoints

All endpoints I provide in swagger ui through this link after running application

- Create short URL
- URL Redirection by short ID
- Delete short Url

Security

- **Data Validation**: Validate URLs before storing them to prevent malicious inputs.
- **Rate Limiting**: Implement rate limiting to prevent abuse of the API.

Error Handling

The API should return meaningful error messages in the following cases:

- **Invalid URL**: When the user submits an invalid URL.
    - Status code: 400 Bad Request
- **URL Not Found**: When trying to retrieve or redirect to a URL that doesn't exist.
    - Status code: 404 Not Found
- **Expired UR**L: When accessing a short URL that has expired.
    - Status code: 410 Gone
- **URL Already in use**: When client enters a URL in use
    - Status code: 409 Conflict
- **Internal Server error**: For unexpected server-side errors
    - Status code: 500 Internal server error

Logging

- **DEBUG Logging**: Capture logs for troubleshooting process.
- **INFO Logging**: Log all information with details for used to Log an INFO message.

All logs stored in logs/application.log in root directory

Conclusion

This document outlines the steps to build a URL shortener API using Spring Boot. The API supports generating shortened URLs, redirecting to the original URLs. You can extend the API further by implementing additional features such gathering statistics and front end.