**Airports Management System**

**Overview**

The Airports Management System is a Spring Boot application designed to manage information about airports and flights. It provides REST APIs for storing and retrieving data about airports and flights, as well as perfoming some statistics-related funcionalities.

**Project Structure**

The project is structured as follows:

**src:** Source code directory

**main**

**java:** Java source files

**Controller**: Classes handling HTTP requests

**Model:** Entity classes representing Airport Flight

**Repository:** Data access and classes using Spring Data JPA

**Service:** Business logic and service classes

**resources:** Configuration files, static files, and templates

**test:** Test source files

**target:** Complied bytecode and packaged JAR file

**flights.csv:**CSV file containing flight data

**airports.csv:** CSV file containing airport data

**pom.xml:** Maven project configuration file

Setup Instructions

1. **Clone the Repository:**

git clone

1. **Build the Project:**

mvn clean install

1. **Run the Application:**

java - jar target/AirportsManagement -1.0-SNAPSHOT.jar

1. **API Endpoints:**

**Airports:**

* /airports’:GET, POST, PUT, DELETE operations on airports
* /airports/country/{country}: GET airports by country
* /airports/city/{city} :GET airports by city
* /airports/name/{name}: GET airports by name
* /airports/minRunways/{minRunways}: GET airports with minimum runways

**Flights:**

* /flights: GET, POST, PUT, DELETE operations on flights
* /flight/departure/{departureAirportCode}:GET flights by departute airport code
* /flight/destination/{destinationAirportCode}: GET flights by destination airport code

1. **Testing**

Run unit tests using:

mvn test

1. **Docker Setup:**

Build a Docke image from project root:

docker build -t airmanager .

Run the Docker container:

Docker run -p 8080:8080 airmanager

The Airports Management System is now set up and can be accessed through the provided API endpoints.