# API Reference

The Driver Management API is organized around [REST](http://en.wikipedia.org/wiki/Representational_State_Transfer). It has predictable resource-oriented URLs, accepts [form-encoded](https://en.wikipedia.org/wiki/POST_(HTTP)#Use_for_submitting_web_forms) request bodies, returns [JSON-encoded](http://www.json.org/) responses, and uses standard HTTP response codes.

Driver Management system based on Spring Boot framework with JPA and Hibernate core dependencies to provide basic CRUD operations. Native Queries and JPQL was used for more complicated queries.

All data stores in Apachy Derby DB. You can find “clientDerby”.jar (JDBC) library in the project and Derby folder (“db-derby-10.14.2.0-bin”) with all binaries.

The project was built as Maven project, and can be simply imported to your IDE (POM.xml located in the root folder). After importing and building the project you can start it by running “Driverscore.java” class.

# Architecture and Functionality

Driver Management system divided to layers. Each layer responsible for one kind of operations only.

Rest Controller (http path: "/driver-service/") receive http requests, process parameters and JSON objects (Jackson library is used ), call relevant function and sends JSON-encoded responses with http code.

Service layer (DriverServiceImpl.java) processes method calls, ifNull checks and (if needed) throws Exception.

DriverRepository extends JpaRepository to provide a persistence layer for an application. It has CRUD method called from super class and 4 custom methods with JPQL annotations for more complicated queries.

The application automatically creates DB and table Driver by annotations in POJO class Driver.java each time you start it. You can change spring.jpa.hibernate.ddl-auto=create-drop to update to disable this behavior. Enum DriverStatus used to process 3 statuses of driver (ACTIVE, INACTIVE, DELIVERING). Java 8 LocalTime library is used to simplify JSON parsing to SQL time, to provide start and end of working time for each driver. The Id member was added to provide a Primary Key for Driver table and ensure uniqueness of each DB record.

# TESTING

In test folder (“/src/test/java”) you can find tests cases for REST API.

All the test cases are based on Junit 4, MockMvc and Mockito libraries. In addition to written test, the API was tested using PostMan.

GitHub repository:

<https://github.com/YuriSokolenko/Bringoz-Driver-Managment--Core.git>