# This is a 2.0 version of Driver application. It has 2 bonus modules: JWT authentication and demo of Angular FrontEnd site.

# 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.

Authentication and Security

Authentication is based on JWT library and Spring Security. To use this version of Driver system please run roles.sql script located in “/driversmanagement/src/main/resources”.

When a HTTP request comes it will go through a chain of filters for authentication and authorization purposes. If request has no JWT TOKEN, it will not reach the API but prompted to SignUp. After registration, user receive a role - an authorization parameter storing in DB table ROLE. Each user receive encrypted password to Log on and enter permitted API for his role (for there is only 2 roles – admin and user).

# Architecture and Functionality

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

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

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

DriverRepository extends JpaReposirory to provide for application a persistence layer. 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.

# How to deploy and run the application?

1. The project was built as Maven project, and can be simply imported to your IDE (POM.xml located in the root folder).
2. After you import the project, you need to start Apachy Derby DB by running “startNetworkServer.bat” file in “/driversmanagement/db-derby-10.14.2.0-bin/bin” folder.
3. After importing and building the project you can start it by running “Driverscore.java” class.
4. Please add roles to Database by running roles.sql or by running those queries:

**INSERT** **INTO** roles(name) **VALUES**('ROLE\_USER');

**INSERT** **INTO** roles(name) **VALUES**('ROLE\_ADMIN');

The Backend application is ready.

Angular application deployment:

To deploy the Angular application you will need a Node.js and angular-http-server installed on your computer.

1. Install Node.js <https://nodejs.org/en/>.
2. Open cmd or terminal and run npm install -g angular-http-server

cd to folder with the project “…/bringoz\_drivers/dist/bringoz\_drivers”.

1. Run angular-http-serverdrivers
2. Browse to localhost:8080;

GitHub repository:

https://github.com/YuriSokolenko/DriversManagment.git