**DOTSUB DEMO**

The deliverable is compound by two folders:

* server. Contains the sources for the backend server application.
* frontend. Contains the sources for the frontend application.

**Server**

**Description**:

The backend was developed using Java 11 + Spring Boot. Maven is used as a dependency management tool. The tests are executed with JUnit 5 Jupiter engine. Other spring plugins used:

* Spring Plugins: Spring MVC, Spring Boot, Spring Boot Test, Spring Data JPA
* H2 IN-Memory DB
* PMD maven plugin were used to make code quality assurance across the java sources.
* Swagger were used to show the api endpoints specification.

Prerequisites to run the server.

1. Maven ^3.0 installed
2. JRE 11 installed

To run the server:

1. Open a terminal and point to the server directory.
2. Set JAVA\_HOME env variable to a JDK 11 home directory.
3. Run *mvn –DskipTests install.* This command will skip tests. As output, it will generate a jar file named *dotsub-demo-0.0.1.jar* in the ‘target’ subfolder.
4. Point to the target folder
5. Run *java –jar dotsub-demo-0.0.1..jar.* This will launch the server application in the url <http://localhost:8080>.
6. If you want to open swagger UI, go to http://localhost:8080/swagger-ui.html

To run the tests:

1. Open a terminal and point to the server directory.
2. Set JAVA\_HOME env variable to a JDK 11 home directory.
3. Run *mvn –DTest=<test\_file> test.* This command will run a specific test under a test suite. If you want to run all the tests, just exec *mvn test*. Example: *mvn –Dtest=DothuWebMVCTest test*.

Tests are located under source folder ‘src/test/java’.

**Files saved**.

Files are saved in the files folder in the current program folder. The file is saved with its database id plus its extension.

**Frontend**

**Description**:

It was developed a SPA in React 16.8 + Typescript 3 using Node Package Manager 8 (NPM) as a dependency management tool. Webpack 4 was used as a module bundler to run the app locally using the webpack-dev-server and to build the app sources to deploy into production. Other tools were used:

* SASS as a CSS preprocessor
* Jest to unit test the UI components
* Enzyme to navigate over the UI components DOM tree and to compare the model values with the rendered values in the output DOM.
* Ts linter to promote code quality assurance

Prerequisites to run the frontend.

1. NPM ^8.0 installed

To run the frontend app:

1. Open a terminal and point to the frontend directory
2. Run *npm install*. This command will generate all the project dependencies in the node\_modules folder.
3. If you want to run the app in a browser, type *npm run start* and it will be launched by webpack dev server in the url http://localhost:4200.
4. If you want to build the app to get the final bundles, type *npm run build* and sources will be generated under ‘static folder.

To run the tests:

1. Open a terminal and point to the frontend directory
2. Run *npm run test*.
3. Test results will be displayed in the console output

The tests are located in source files with name matching pattern \*\*.test.tsx.

**Some tasks remaining**

1. Unit test for individual components, such as the CalendarInput and FileInput components.
2. Add some tests to test parameters border cases
3. Compress save files stored in the backend server to save disk space