Technical Specification

The application is a web app made with Angular8 for the front-end and Java Rest API + Jersey + Jax-Rs for the back-end.

The application is (similarly to Socrative)for generating a quiz for the classroom as a teacher, and taking the evaluation on said quiz as a student.

Features :

The scope covered by this application is the following :

- Register users who could be either a teacher or a student (The users created are stored in a database)

- Logging in the users and based on the roles of the user give an appropriate menu.

- AS TEACHER :

- you can create questions with as many choices as you set.

- you can create a quiz, set its title and select the questions you want to create the quiz with.

- AS A STUDENT :

- You choose the quiz you want to take.

- Run the evaluation and provide the automatic mark in the end of this execution.

Authentication :

The app use JWT to generate a unique token per session for users. This token is also used to give permissions to performing CRUD actions based on roles.

JAVA :

- The user logs in

- A token is generated using a HS512 signing key based on username and claiming a role (USER, ADMIN)

- The login function authenticated the user's username and password against the database and if the data matches gives the token in the 'authorization' of the response header.

ANGULAR :

- When the user logs in, the response header is read and the token is extracted

- The token is saved in a localStorage

- An Http interceptor is defined, the token extracted earlier is added to the 'authorization' of the http header and is distributed thanks to the interceptor to all http requests that follow in order not to manually replace each time.

- On logout the localStrage is emptied.

Permissions :

Aside from authentication, the token is also used for managing permission.

For main CRUD functions for creating quizzes a guard is added before the function @JWTTokenNeeded(Permissions = Role)

Each function with this guard validates the token and extracts the role of the user from it. If the user role matches the roles defines in permissions he will be able to make the http request otherwise an UNAUTHORIZED status is thrown.

Future Improvements :

- Error handling, for instance error handling is only done in the back-end side, nothing on the front-end yet.

- Some pages don't refresh automatically which doesn't over a great UX.