Instituto Superior Técnico

Mestrado Bolonha Engenharia Eletrotécnica e de Computadores

Distribuited Applications on the Internet

Project - Part 2

Group 2

José Duarte Ferrão de Oliveira Lopes, nº 100336

Francisco Miguel Bernardo Mendes, nº 96529

Professor João Nuno de Oliveira e Silva

October 2023

Index

1 - Functionalities

Our application performs the following functionalities, later explained where these are performed.

- Authentication

  (F 1) login on FENIX

- Restaurants and canteens

  (F 2) see the current menu
  (F 3) check-in in a restaurant
  (F 4) check-out from a restaurant
  (F 5) evaluate the meal at the restaurant that he checked-in

- Class rooms

  (F 6) see a room schedule
  (F 7) verify if the next class in the room is from one enrolled course
  (F 8) check-in a class that is taking place in the room
  (F 9) check-out a class

- Study room

  (F 10) Check-in and assign an enrolled class to a study period
  (F 11) Check-out a study period

- Other users

  (F 12) Send messages to users that are on the same room

2 - Main App

The main app is written in HTML and Javascript, and supported by a Flask backend.

2.1 Login Menu

In this menu, users can perform the following actions:

2.1.1 Login with Fénix

By clicking the login button, users will be redirected to the Fénix login page **(F 1)**. If authentication is successful, they will arrive at the Main Menu (Section 2.2).

2.1.2 Scan a Room/Restaurant QRCode

Users can scan a Room QR code and get the room's schedule **(F 6)**, or scan a Restaurant QR code and get its Menu **(F 2)**. The string in the QR Code should be 'menu/<resId>' for a restaurant and 'spaces/<roomId>' for a room.

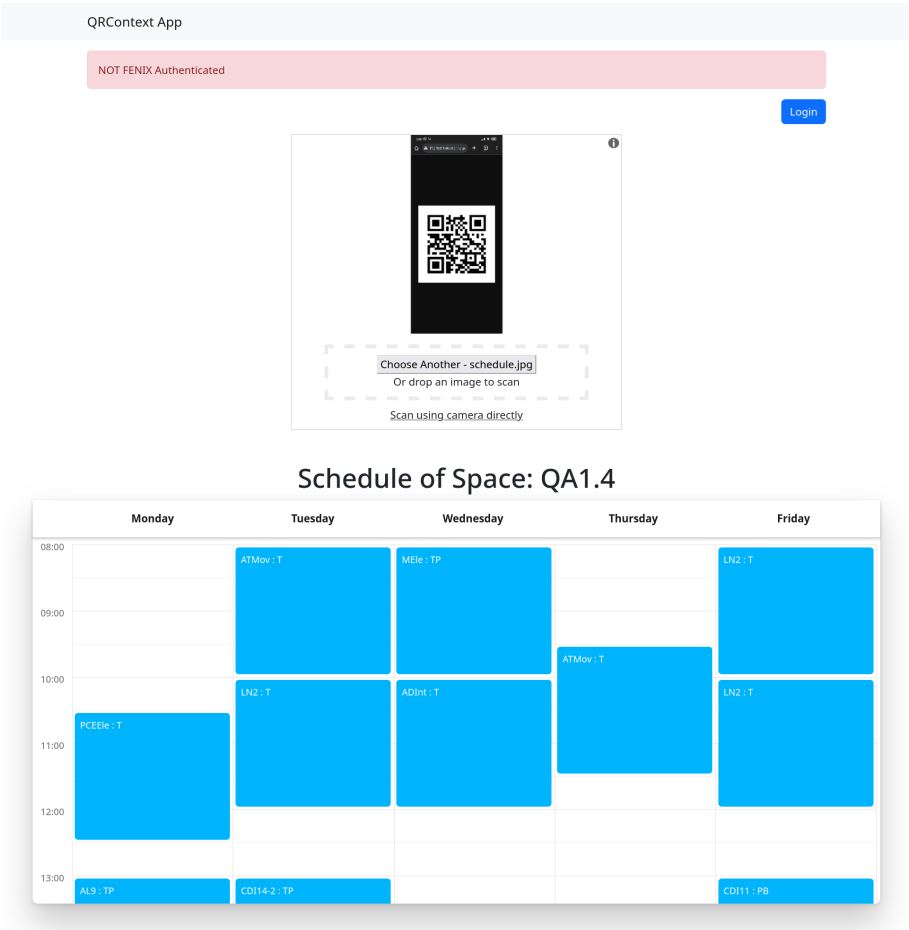

Figure 1: Login Menu: here users can Login and Scan a QR code of a Room/Restaurant and get its information.

2.2 Main Menu

This is the menu available after the user logs in with their Fénix credentials. It allows the following actions:

2.2.1 Logout of Fénix

By clicking the logout button, the user will be logged out and redirected to the Login Menu (Section 2.1).

2.2.2 Scan a Room/Restaurant QRCode

Users can scan a Room QR code and get the room's schedule **(F 6)**, or scan a Restaurant QR code and get its Menu **(F 2)**. Additionally, this now allows the User to check-in on the room or restaurant (Section 2.2.3).

2.2.3 Check-in and Check-out

If the user scans a **Restaurant QR Code**, they can check-in on the said restaurant by clicking the 'CheckIn' button **(F 3)**. This now allows the user to Evaluate the Restaurant with a written comment and a 5-star rating **(F 5)**.

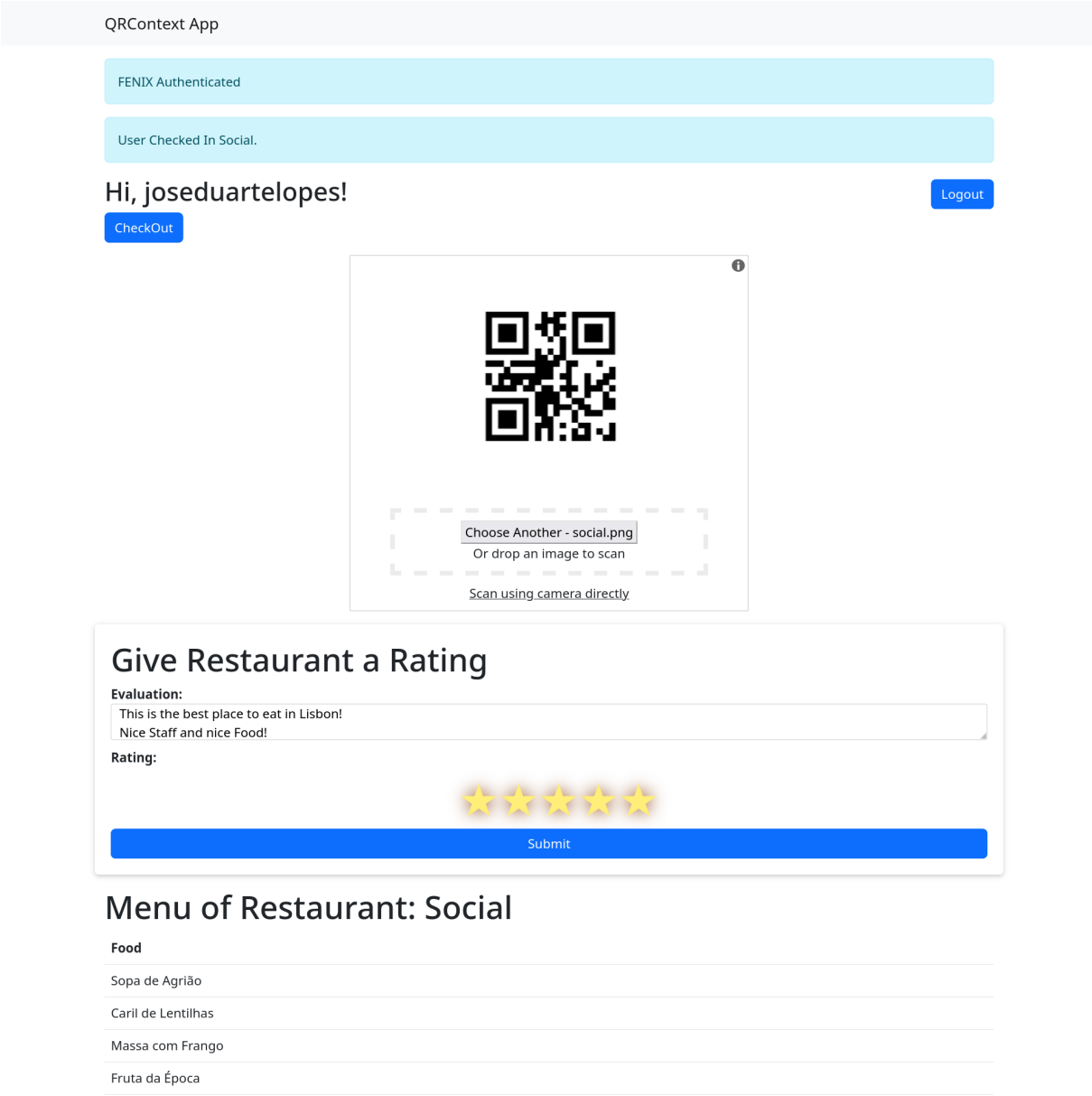

Figure 2: Main Menu: if the user Checks-in on a restaurant, they can leave an evaluation.

If the user scans a **Room QR Code**, the following can be done:

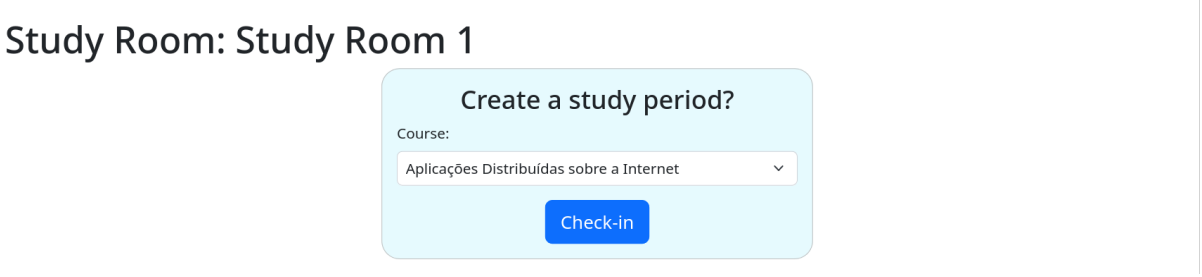

Figure 3: The user can assign a course they're enrolled to the study period.

- If it is a regular room and the user is enrolled in the current class (or the next one if it's in less than 15 minutes) **(F 7)**, they can check-in the class by clicking the button **(F 8)**.
- If it is a study room, the user can assign a course they're attending to the study period and check-in **(F 10)**.

Checking in a Room allows the user access to the Messages Menu (Section 2.3)

A user can perform the check-out by clicking the 'CheckOut' button [**(F 4)**, **(F 8)** and **(F 11)**]. A user can not check-in another room without checking-out.

2.3 Messages Menu

Upon checking in a room, the user is able to send messages to other users in the same room, in the Messages Menu **(F 12)**.

Here, it is presented a drop-down list of users in the same room, of which the user can select one to chat. There is also a 'chat-like' window with the messages sent to and received from the chosen user. And at the bottom, an input area to write a new message to send, by clicking the 'Send' button.

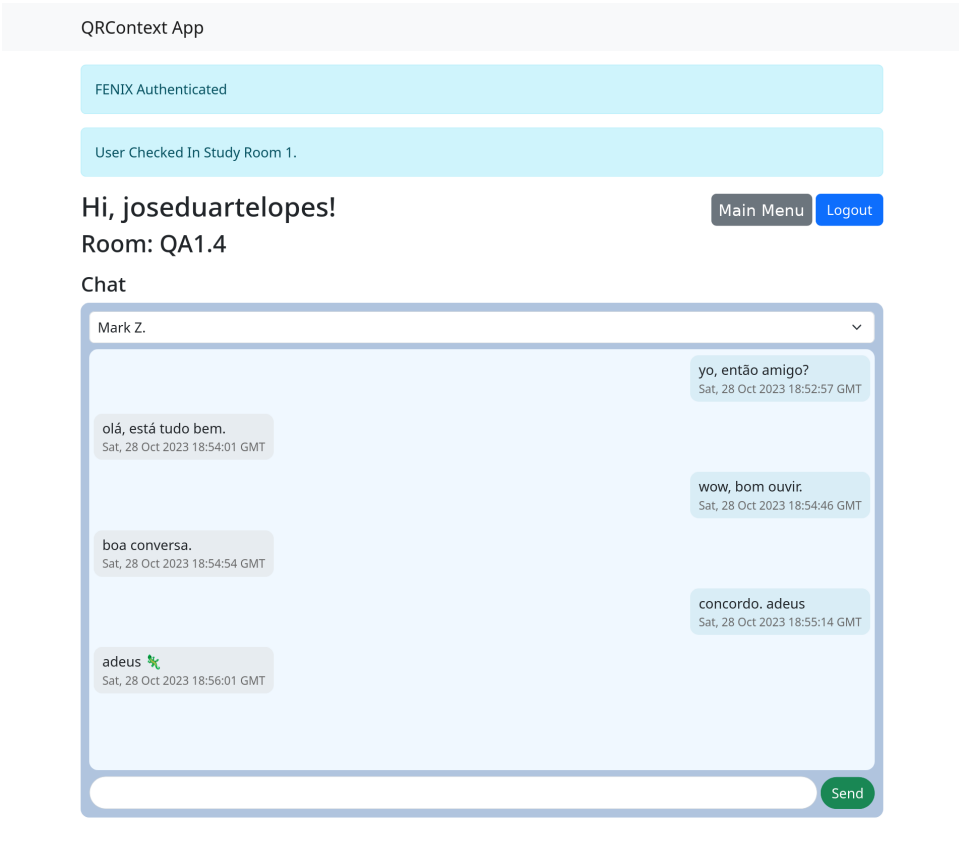

Figure 4: Message Menu: in this page the user can send text messages via a chat to other users also checked in the same room.

3 - QRContext: Flask Backend

3.1 Endpoints

3.1.1 Index

URL /

Method GET

Description The home page of the application.

Hash it was implemented hash differentiation.

- `#mainMenu` see Section 2.2.
- `#messages` see Section 2.3.

3.1.2 Logout

URL /logout

Method GET

Description Logs out the user and redirects to the home page.

3.1.3 OAuth2 Authorize

URL /authorize/<provider>

Method GET

Description Initiates OAuth2 authorization with a specific provider.

3.1.4 OAuth2 Callback

URL /callback/<provider>

Method GET

Description Handles the OAuth2 callback from the provider and logs the user in.

3.1.5 HTML5 QR Code Reader Script

URL /files/html5-qrcode.min.js

Method GET

Description Serves the HTML5 QR Code reader JavaScript library.

3.1.6 QR Code Reader

URL /API/QRCodeReader/<path:data>

Method GET

Description Reads QR code data and sends requests to other servers based on the decoded data.

3.1.7 Check-in

URL /API/checkin/<userId>

Method POST, GET

Description If POST: handles users Check-in. If GET: retrieves user Check-in status.

POST Parameters (if Method is POST)

- **roomId** (string): Id of the room to checkin

3.1.8 Checkout

URL /API/checkout

Method POST

Description Handles user check-out.

3.1.9 Users Checked-in

URL /API/<path:roomId>/users

Method GET

Description Retrieves users who have checked in to a specific room or space.

3.1.10 Messages

URL /API/messages

Method GET, POST

Description Handles messages between users. The current user id is obtained within the app.

GET Parameters In query

- **other_user** (string, query parameter): The user id of the other user in the conversation.

POST Parameters In JSON body

- **to** (string, JSON parameter): The user id of the other user of the conversation.
- **message** (string, JSON parameter): The message to be stored.

3.1.11 Evaluation

URL /API/evaluation

Method POST

Description Submits evaluations for a restaurant or service.

Parameters The endpoint expects the following JSON parameters in the request body:

- **resId** (string): The ID of the restaurant.
- **userId** (string): The ID of the user submitting the evaluation.
- **eval** (string): The written evaluation.
- **rating** (string): The rating provided for the restaurant.

3.1.12 Get Courses

URL /API/person/courses

Method GET

Description Retrieves from Fénix the information about the courses associated with the logged-in user.

4 - Changes from Part 1

4.1 FoodService

The following endpoints were added to handle the FoodService Operations.

4.1.1 API Restaurant Information

URL /API/restaurant/<resId>

Method GET

Description Retrieves a restaurant's information (menu and name) based on the provided id 'resId'.

4.1.2 API Evaluation

URL /API/evaluation

Method POST

Description Submits an evaluation for a restaurant.

POST Parameters In JSON body

- **userId** (string, JSON parameter): The ID of the user submitting the evaluation.
- **resId** (string, JSON parameter): The ID of the restaurant being evaluated.
- **eval** (string, JSON parameter): The evaluation data.
- **rating** (string, JSON parameter): The rating provided for the restaurant or service.

4.2 RoomService

4.2.1 Database

Some changes were made to the database.

1. The Space Class now has a `type` attribute to differentiate between a `ROOM` and a `STUDY_ROOM`.
2. The information about start and end `datetime`s in the Event Class was changed to only store the `time`.

4.2.2 Functions

updateScheduleFromFenix(space_id): Receives the `space_id` and, through a REST GET request to Fénix, updates the schedule for that space.

4.2.3 Endpoints

Some endpoints were changed and/or added, including '`/spaces/`', '`/spaces/<space_id>`', and '`/spaces/calendar/<space_id>`'. But these endpoints are only aesthetic and not relevant for the Main App, and therefore will not be explained.

However, there was one important endpoint added.

URL /API/spaces/<space_id>

Method GET

Description Retrieves all the information regarding the space with the provided id, `space_id`.

4.2.4 RoomAdminApp

This App was modified to accommodate the changes in the Room Service: added the demand for a `type` when trying to create a new room; and it was added one more option so that the schedule information is retrieved from Fénix.

4.3 CheckIn/Out

The following endpoints were added:

4.3.1 API Check-In

URL /API/checkin/<userId>

Method POST, GET

Description Handles user check-in and provides information about user check-in status. When using POST, it allows users to check in to a specific room by providing 'roomId'. When using GET, it retrieves the current check-in status for the user with the specified 'userId'.

4.3.2 API Checkout

URL /API/checkout

Method POST

Description Handles user check-out. Users can check out by providing their 'userId'.

4.3.3 API Users Checked-In

URL /API/<path:roomId>/users

Method GET

Description Retrieves a list of usernames for users who have checked in to the specified 'roomId'.

4.4 MessageApp

4.4.1 API Messages

URL /API/messages/<user_id>

Method GET, POST

Description Handles the messages of the user with the provided id. If POST: stores the new message in the database. If GET: retrieves all messages between two users.

GET Parameters As `data`

- **other_user** (string, as `data`): The user id of the other user in the conversation.

POST Parameters In JSON body

- **to** (string, JSON parameter): The user id of the other user of the conversation.
- **message** (string, JSON parameter): The message to be stored.

5 - Conclusion

At the end of this project, it was possible to implement every service and functionality asked.