Mobile Computing Assignment #1
Mobile Ticket and Payment System

Cláudia Marinho
up201404493

Tiago Silva
up201402841

November 19th, 2018

1 Introduction

A big music and event theater (for instance http://www.casadamusica.com) wants to provide to its customers an integrated system for an easy acquisition and validation of tickets and cafeteria vouchers.

This system can be decomposed into four different applications:

- RESTful Services, running on a server.
- Customer App, running on Android.
- Ticket Validation Terminal, running on Android.
- Cafeteria Order Terminal, running on Android.

In the following sections we'll go more in-depth about each of these, their architecture and what operations they are able to accomplish.

2 Architecture

2.1 RESTful Services

These services are running on a NodeJS server connected to an SQLite database. All the other apps connect to the server at some point in order to consume its services, be it for buying tickets, completing cafeteria orders or validating vouchers, (among others). Besides the Express Framework, the server also makes use of the following NodeJS packages:

- body-parser - for ease of access to the request body.
- sqlite3 - for accessing the SQLite database.

- bcrypt - for providing password hashing and salting for safely storing user passwords.
- uuid/v4 - for creating unique identifiers when needed (such as user and voucher ids).
- node-rsa - for validating the request signature.

The list of available services and their respective endpoints are as follows. Please not that the request **should be signed** when buying tickets or cafeteria orders.

- Users
  - GET /login - Check if user credentials match.
  - GET /users/:id - Get use by id.
  - POST /users - Create a new user.
  - PUT /users/:id - Update user by id.
- Credit Cards
  - GET /users/:id/creditcard - Get user's credit card.
  - POST /users/:id/creditcard - Create a new user's credit card.
  - PUT /users/:id/creditcard - Update user's credit card.
- Shows and Tickets
  - GET /shows?page=<PAGE>&limit=<LIMIT>- Get the next airing shows.
  - POST /shows/:id/tickets - Buy tickets for a show.
  - POST /shows/:id/tickets/validation - Validate tickets for a show.
  - GET /users/:id/tickets - Get all tickets belonging to a user.
- Products
  - GET /products - Get all products.
  - GET /products/:id - Get a product by id.
- Vouchers
  - GET /users/:id/vouchers - Get all vouchers belonging to as user.
  - GET /users/:userId/vouchers/:voucherId - Get a voucher belonging to an user.
- Orders
  - GET /users/:id/orders - Get all orders belonging to an user.
  - POST /users/:id/orders - Create a new order.

The database is structured as such:

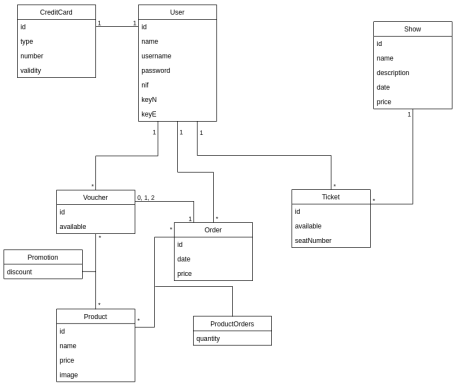

Figure 1: UML model of the databse.

2.2 Customer App

The Customer App has a local SQLite database in which vouchers and tickets are stored locally. This local database is used anytime these vouchers and tickets need to be accessed on the app. It is to be noted that the local database's data is updated each time the user sees the transactions' history in the app. In this operation, all the tickets marked as used in the server are deleted from the local database. Furthermore, the vouchers that were selected by mistake and that were not used in any order are sent back to the app and stored locally.
The app's registered users are also stored locally in the internal storage. The username of the currently logged-in user is also saved so that the user can be automatically logged-in whenever he opens the app (also, whenever the user isn't logged-in, he is redirected to the login page).
The user's current cafeteria order is also stored locally on the Shared Preferences so that it isn't lost even when the user navigates through the pages.
It is also important to mention that to connect to the server it is needed to change the server address variable to the machine's address in which the server is running on the class ServerConnection. The google library 'com.google.zxing:core:3.3.3' is also used in this app to generate the QR codes.

2.3 Ticket Validation Terminal

The google library 'com.google.zxing:core:3.3.3' is used in this app to scan QR codes.

2.4 Cafeteria Order Terminal

The google library 'com.google.zxing:core:3.3.3' is used in this app to scan QR codes.

3 Interface

3.1 RESTful Services

Since these are running on a simple NodeJS server, there is no fancy interface for them. However, the server has a simple command line interface that prints information every time an error occurs.

3.2 Customer App

The first screen the user faces when he opens the app for the first time is the login screen (**figure 2**) in which the user has to insert his username and password, if he is already registered. If he is not, he should sign up by clicking on the text urging the user to create an account.

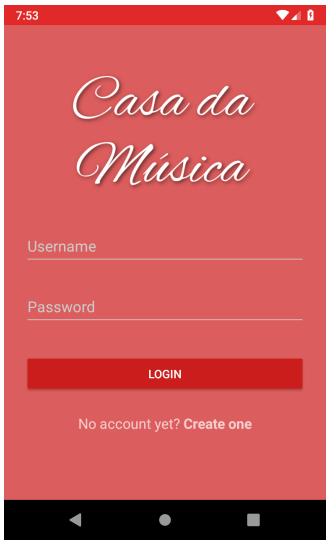

Figure 2: Login Screen

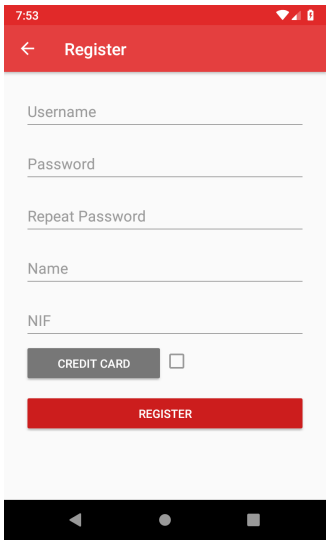

Figure 3: Register Screen

In the register screen (**figure 3**), the user has to insert his username, password, name and nif. He should also insert his credit card information by clicking on "Credit Card" that will trigger a fragment in which the user has to choose the card's type, insert the the card's number and indicate the card's expiration date (figure 4).
Note that all the fields are validated, in such a way that the user can't sign up with empty data in any of the fields mentioned before, has to insert the same password in the fields "password" and "repeat password", the credit card's expiration date can't be invalid and the username, nif and card number must be unique to the user. The password also has to have at least 4 characters.
After registering with valid credentials, the user can now log into the app. Note that if the user is logged-in when he leaves the app, he will be able to start the app automatically logged-in.

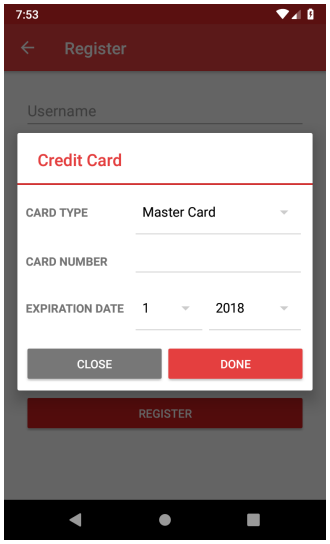

Figure 4: Credit Card

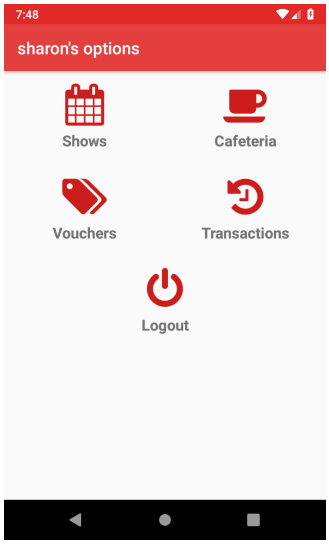

Figure 5: Main Screen

On the main screen (figure 5), the logged-in user can access all the main features of the application. He can also log out whenever he wants so he can create log into another account.

Figure 6: Shows' List

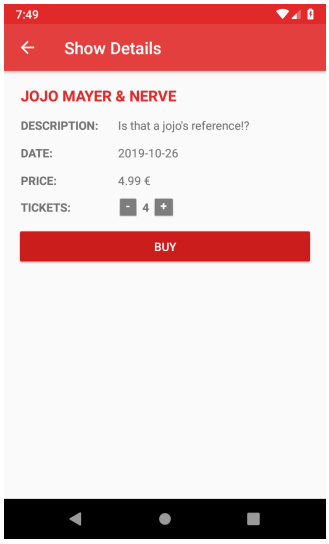

Figure 7: Show Details

If the user chooses the option "Shows" on the main screen, the user will see a list of the upcoming shows to buy tickets for (Figure 6). Note that pagination was implemented for the shows, in such a way that they show up as the user scrolls down. If the user clicks on a show, he will be redirected to a screen in which he can see the show's details and buy any number of tickets for a show (Figure 7). After clicking on "Buy", a pop-up will show up confirming the purchase of tickets with the total price. If the user confirms, the user will have to perform a local user authentication in which he only has to insert his password. After this, a message informing the user of the number of free vouchers received is shown and both the bought tickets and vouchers are stored in the local database.

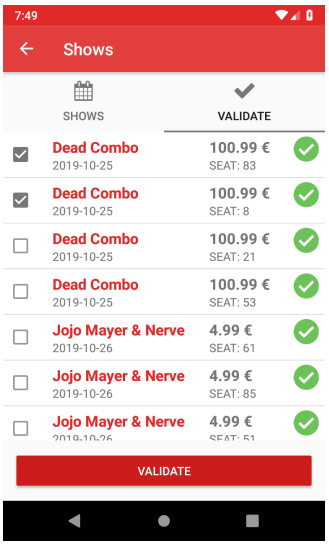

Figure 8: Bought Tickets

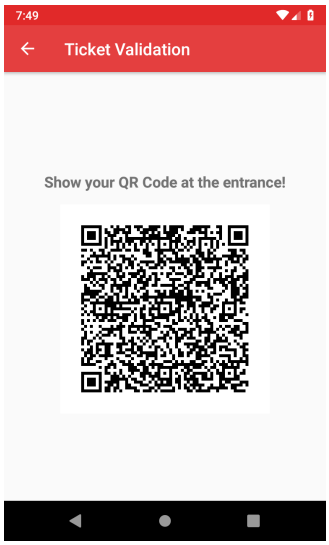

Figure 9: Tickets' Validation

On the same screen, the user can select the tab "Validate" which displays all the already bought tickets (figure 8). Here the user can select up to four tickets of the same show to validate (by clicking on the "check" of each ticket). If the user tries to validate tickets for more than four shows or for different shows, error messages are shown. After clicking on "Validate", a pop-up is shown to confirm the validation of the tickets. At this point, if the user confirms the validation the tickets are marked as "used" in the local database and they are shown in the list of bought tickets as greyed-out and with a red X mark (the user also can't select them anymore). The user is also redirected to a screen with a QR Code (figure 9) that should be shown to the Ticket Validation Terminal at the entrance of the event to which the tickets are for.

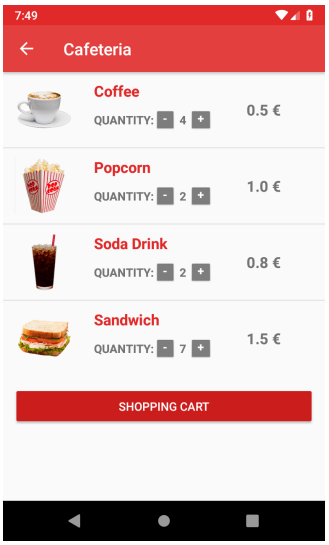

Figure 10: Cafeteria

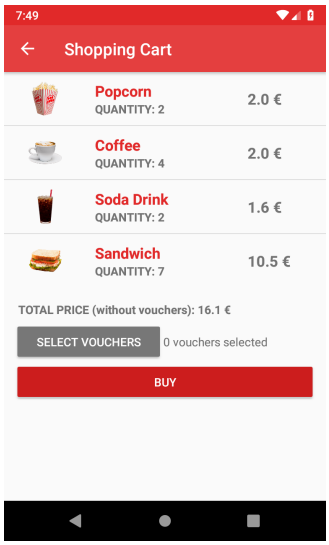

Figure 11: Shopping Cart

If the user chooses the option "Cafeteria" on the main screen, he will be redirected to a screen displaying all products available (figure 10). In this screen the user can choose the amount he wants for each product. After clicking on "Shopping Cart", the user will be redirected to a screen in which the total price for each product is shown and an estimate of the price the user will have to pay without vouchers applied (figure 11). Then the user can select the vouchers he wants to add to his order by clicking on "Select Vouchers".

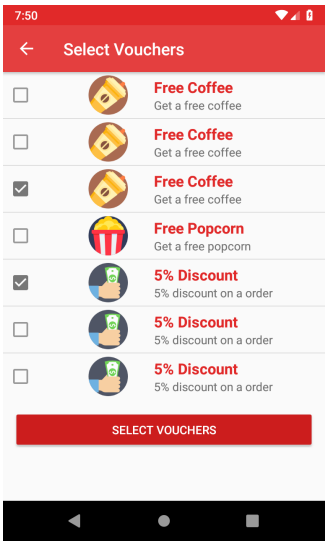

Figure 12: Select Vouchers Screen

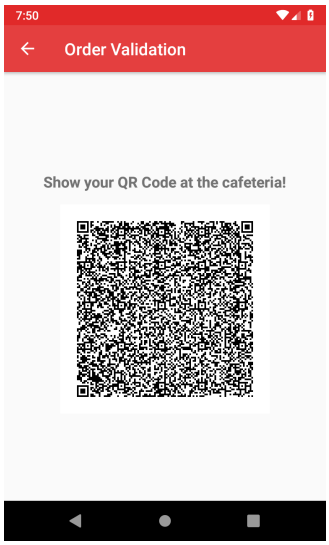

Figure 13: Order Validation

On the "Select Vouchers" screen, the user can select up to two vouchers (figure 12). The user can't also select two vouchers of the type "5% discount". If he tries to select invalid vouchers, error messages (toasts) will be shown. If the user clicks on "Buy" in the Shopping Cart Screen, then the user will be redirected to a page with a QR code matching the user's order that should be shown at the Cafeteria Terminal (figure 13).

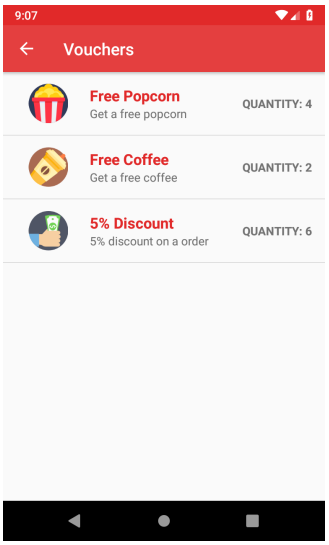

Figure 14: Available Vouchers

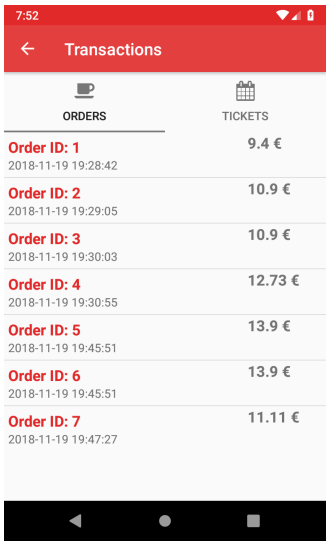

Figure 15: Orders' Transactions

If the user chooses the "Vouchers" option at the main screen, he will be redirected to a screen showing all the available vouchers for each type of voucher (figure 14). If the user chooses the "Transactions" option, his tickets and vouchers will be updated and the orders' history will be shown by default (figure 15).

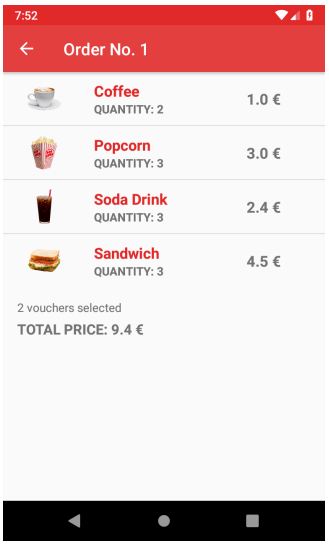

Figure 16: Available Vouchers

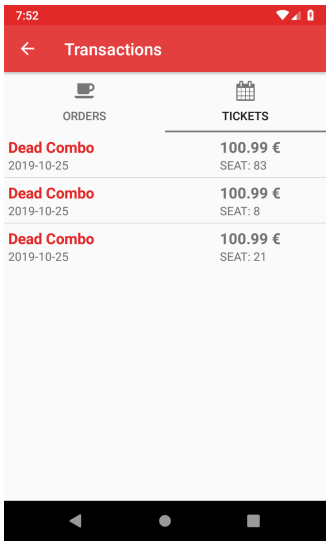

Figure 17: Orders' Transactions

By clicking in each order, the user can see more information about each order, like the products bought, their quantity, their price, number of vouchers used and the orders' total price (figure 16). If the user changes the Transactions' tab to "Tickets" then he will be able to see all bought and validated tickets up until now (figure 17).

3.3 Ticket Validation Terminal

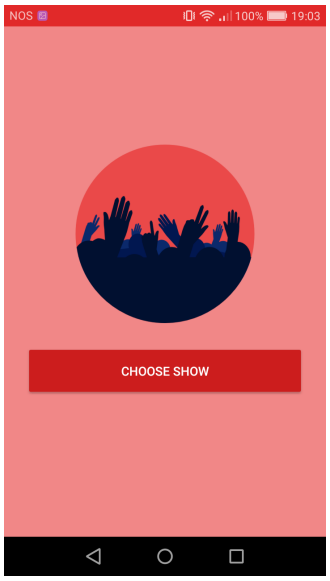

Figure 18: Events' Main Screen

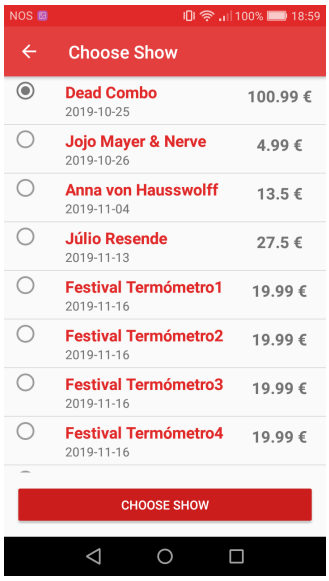

Figure 19: Upcoming Shows

As for the Ticket Validation Terminal, the user is first shown a fancy main screen (figure 18). Then the user should click on "Choose Show" in order to choose the event which the app will validate. After selecting one of the shows by clicking on them, the user will either be asked whether he wants to download a QR code's scanner if he doesn't have an app that takes cares of that already or will be redirected to the scanner itself which will show the device's camera. Here the user should point the camera to the QR code of the tickets to validate. If the detected tickets are not for the show selected, a error message will be shown.

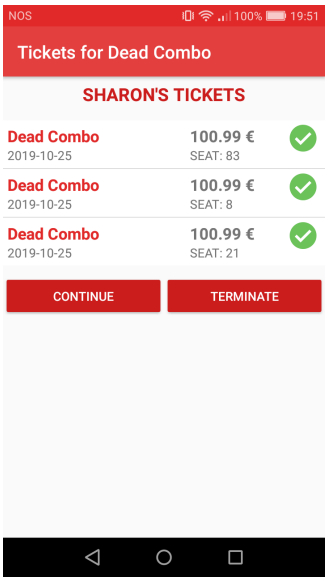

Figure 20: Valid Tickets

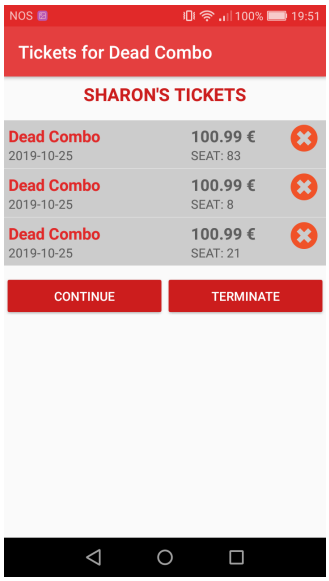

Figure 21: Invalid Tickets

If the tickets detected are for the selected show, then the user will be redirected to a screen which tell the user whether the tickets were validated or not. The first example (figure 20) is of valid tickets and second example is of invalid tickets (figure 21). Then the user can choose whether to keep scanning tickets for the same show (by clicking on "Continue" which will redirect to the scanner immediately) or to stop scanning tickets for that show. If the user chooses to stop scanning tickets for the same show, he will be redirected to the shows' list screen.

3.4 Cafeteria Order Terminal

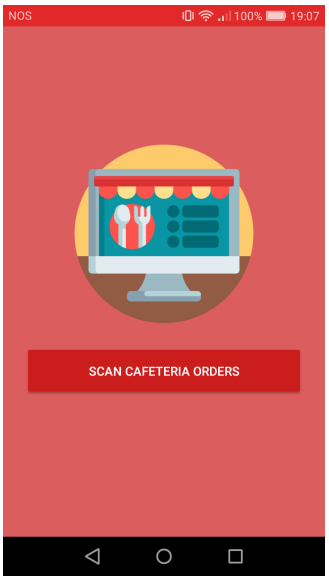

Figure 22: Cafeteria's Main Screen

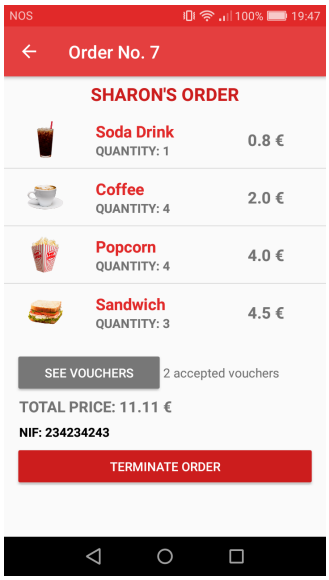

Figure 23: Order Details

As for the Cafeteria Order Terminal, the user is first shown a fancy main screen again (figure 22). After clicking on Scan Cafeteria Orders, the app will try to scan QR codes as described before and if the scanner detects a QR code, then the user will be redirected to a screen showing the order's details (figure 23). In this screen, not only the products and its quantities are shown, but also the order's id, accepted vouchers, total price and buyer's NIF. By clicking on "Terminate Order", the user can scan more orders.

4 Features

All of the requested features and operations were implemented successfully. In addition, the project is flexible enough that using different kinds of vouchers (including one voucher discounting multiple products) would be a simple change to implement. Also as an extra, the customer application supports multiple user accounts. However, because each user account requires a private key, an user account can only be accessed on the same phone it was first created in (e.g. an account created on phone A can't be accessed on phone B).

As far a communications go, since the group didn't have at their disposal two phones capable of using NFC, the communication between the Android applications was done through the use of QR Codes.