

ACME Coffee Shop

Report

Mobile Computing

Integrated Masters in Informatics and Computer Engineering
2020/2021

Grupo

Diogo Alexandre Silva Teixeira - up201606124
Fábio Alexandre Matos Azevedo - up201606540
Guilherme dos Santos Amaro - up201508537

Faculdade de Engenharia da Universidade do Porto Rua Roberto Frias, sn,
4200-465 Porto, Portugal

18 de Novembro de 2020

Index

Index	1
Introduction	2
Architecture	2
2.1 Description	3
2.2 Client	4
2.3 Terminal	5
2.4 Server	5
2.5 Database	6
Functionalities	7
3.1 Register	7
3.2 Login	8
3.3 Logout	9
3.4 Profile	10
3.5 Previous Transactions	11
3.6 Previous Transaction Details	12
3.7 Available Vouchers	13
3.8 Menu	14
3.9 Checkout	15
3.10 Checkout transaction (terminal side)	16
3.11 Saved Data	17
Performed Tests	18
Setup	19
Conclusions	20

1.Introduction

The project was developed as part of the class of Mobile Computing, which is a part Masters in Informatics and Computer Engineering course at the Faculty of Engineering of the University of Porto.

The project simulates a coffee shop app, with which users can make their orders, by adding items to the cart, generate a QR Code and proceed to the Terminal, where their order will be processed and the transaction complete.

The development consisted in the cooperation between two Android mobile applications and a server. The three are able to communicate between each other, in order to keep information updated.

2. Architecture

2.1 Description

The first application, the Client app, is the main application to be used by the customers, where they can register into the system, add products and checkout their basket.

The second application, the Terminal app, is the secondary application which reads the generated QR Code and sends the information back to the server.

The server is responsible for communicating with the other apps and updating the coffee shop database, by registering users and processing checkouts. Both apps were developed in Android Studio, using the Volley library to make requests to the server, which was implemented in Express.js, a Node.js web application framework, with an SQLite database.

The communication with the server is made with the HTTP protocol, using JSON formatted requests.



Image 1 - System Architecture

2.2 Client

The Client app was developed in Java, using Android Studio.

The `ui.client` Package includes all the used Activities in the application and a `Util` class that contains reusable functions between the activities.

The `data.model` Package contains the three Entities used throughout the application, in which the `User` contains reference to the others.

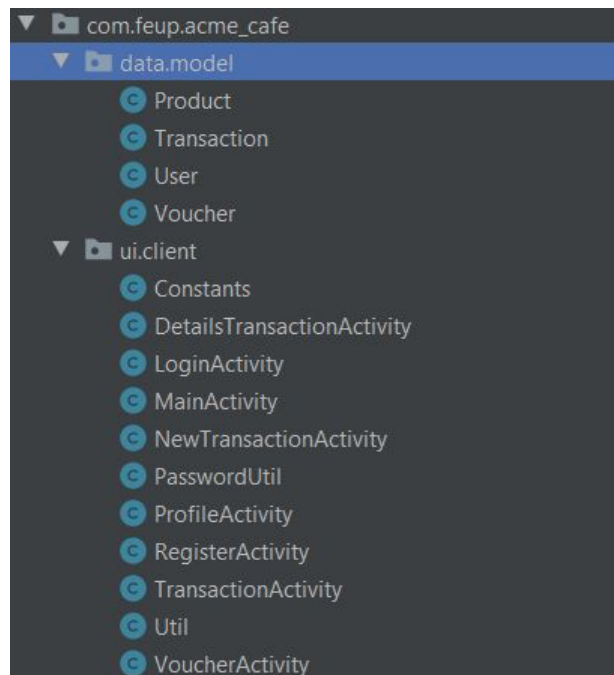


Image 2 - Client app structure

2.3 Terminal

Likewise, the terminal app was developed in Java, using Android Studio. The main package contains 2 activities and 2 fragments.

The `QRAct` Activity is used for taking a QR code using the camera, while the `Main` Activity is the normal terminal activity, where both fragments are used.

`OrderFrag` is used to show the order taken from the QR code, and to confirm its contents, while the `ResultFrag` fragment shows the Order number on a successful checkout, or an error message on failure.

The `utils` sub-package contains a single class holding useful constants, such as the server address.

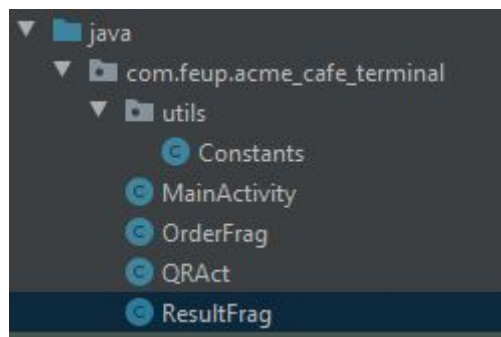


Image 3 - Terminal app structure

2.4 Server

The server was developed in Javascript, using the Express framework. With a REST API, it is able to accept requests and receive and reply JSON requests. Aided with Sequelize ORM, the database queries are easily processed.

The *database* folder contains the database file used to store the data. The routes folder includes all the used routes to process the requests.

The *sequelize* folder contains the models folder, which contains the used models from the database, and the sequelize.js file which sets up the database connection. The app.js file starts up the server.

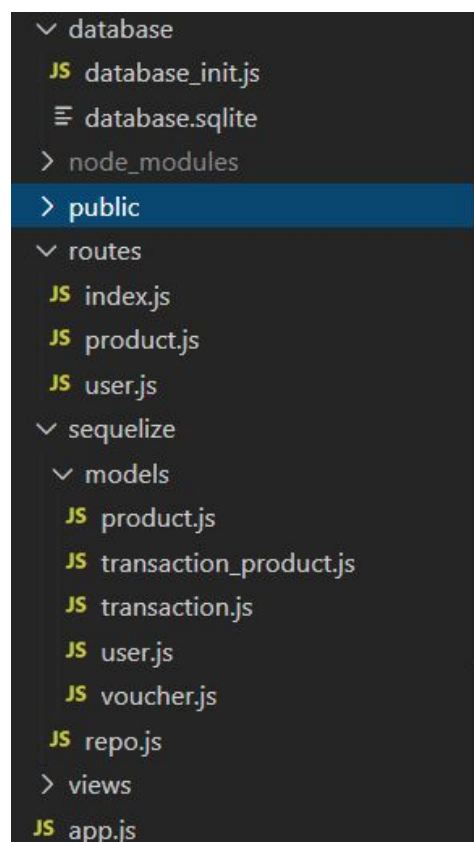


Image 4 - Server Structure

2.5 Database

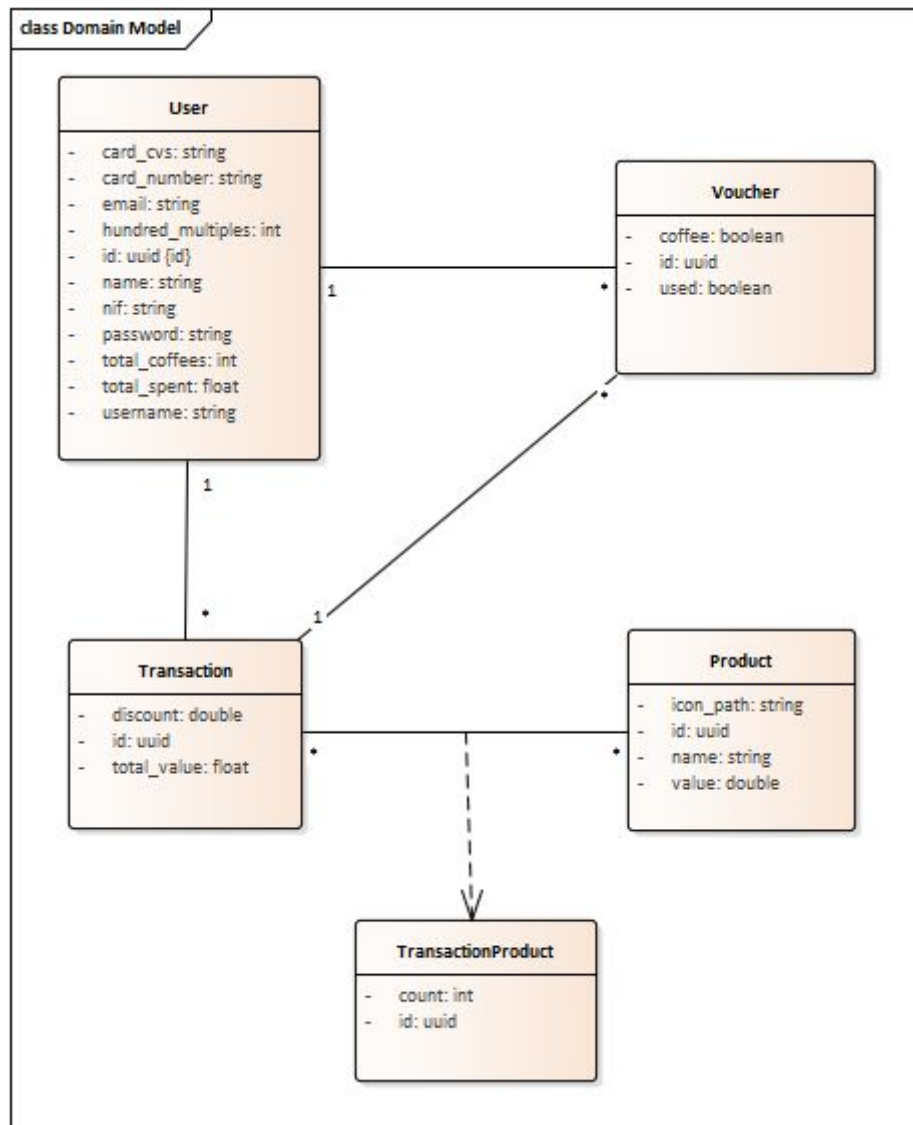
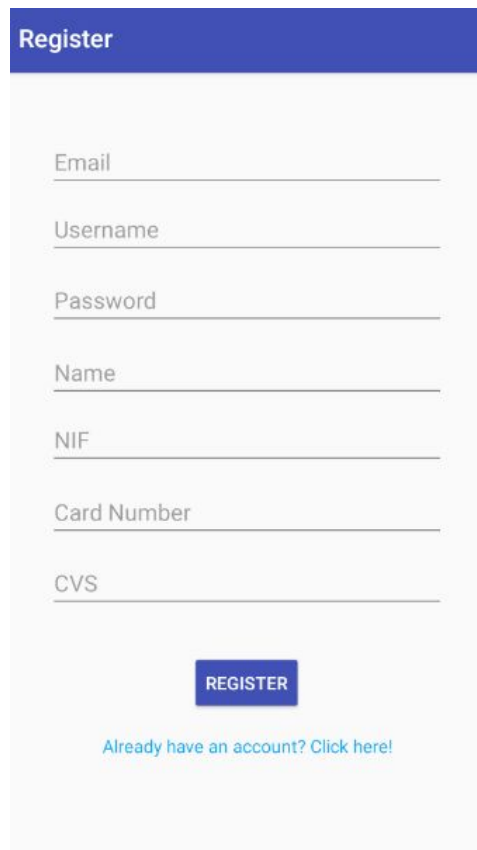


Image 5 - Database Relational Model (ignoring the certificate and qr list tables)

3.Functionalities

3.1 Register

The user must be registered to access all the features of the service. To register, customers fill out a form containing information regarding the user's name, username, password and credit card information. The server receives the info, creates a new user and then the app redirects to the login page. If any of the fields is empty, the user is notified.

A screenshot of a mobile application's registration form. The form has a blue header with the word "Register" in white. Below the header, there are seven text input fields, each with a label above it: "Email", "Username", "Password", "Name", "NIF", "Card Number", and "CVS". At the bottom of the form, there is a blue button with the word "REGISTER" in white. Below the button, there is a link in blue text that says "Already have an account? Click here!".

Register

Email

Username

Password

Name

NIF

Card Number

CVS

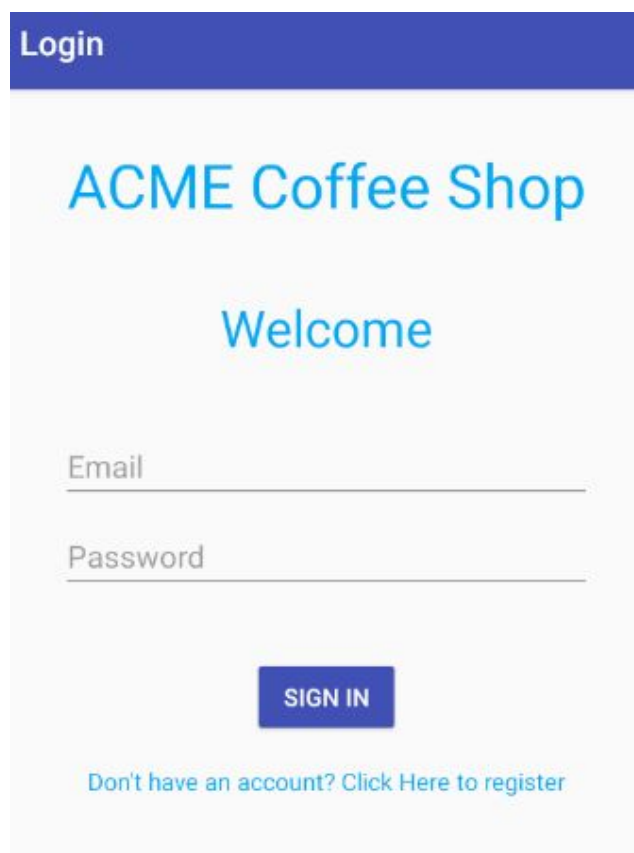
REGISTER

[Already have an account? Click here!](#)

Image 6 - Register Form

3.2 Login

Customers must log in to their account in order to use the app. To login, users fill out a form containing the username and password. The information is sent to the server, which replies with the user's data and allows him/her to enter the app. If the username and password combination isn't correct, the user is notified. If any of the fields is empty, the user is notified. This is the first screen shown to the user, from here he could go to the register page if haven't an account yet.



The image shows a login form for 'ACME Coffee Shop'. It features a blue header with the word 'Login'. Below the header, the shop's name 'ACME Coffee Shop' is displayed in a large blue font, followed by the word 'Welcome' in a smaller blue font. There are two input fields: 'Email' and 'Password', each with a horizontal line for text entry. Below these fields is a blue button with the text 'SIGN IN' in white. At the bottom, there is a link that says 'Don't have an account? Click Here to register' in blue text.

Image 7 - Login Form

3.3 Logout

If the user wishes, they can safely log out of the app and save their data locally. All they need to do is click on the options and select logout.



Image 8 - Logout Option

3.4 Profile

The user can access their profile, in order to check their information, including the amount of stored discount and the number of available vouchers, as well as the total amount they spent in the store. From here the user can also choose to see his previous transactions and available vouchers.



Image 9 - Profile page

3.5 Previous Transactions

From the profile page the user can access a list of its previous transactions. Is shown to the user the date, and total price of each transaction.

Transactions	
18 November 2020	
13:40:45	152.00€
18 November 2020	
13:40:45	30.00€

Image 10 - Previous Transactions List

3.6 Previous Transaction Details

If the user selects a transaction from the list of previous transactions a page containing the info about the transaction will be displayed. In this page the user can see the products that are associated with the transaction and the total price. The user can delete the transaction if he wants.

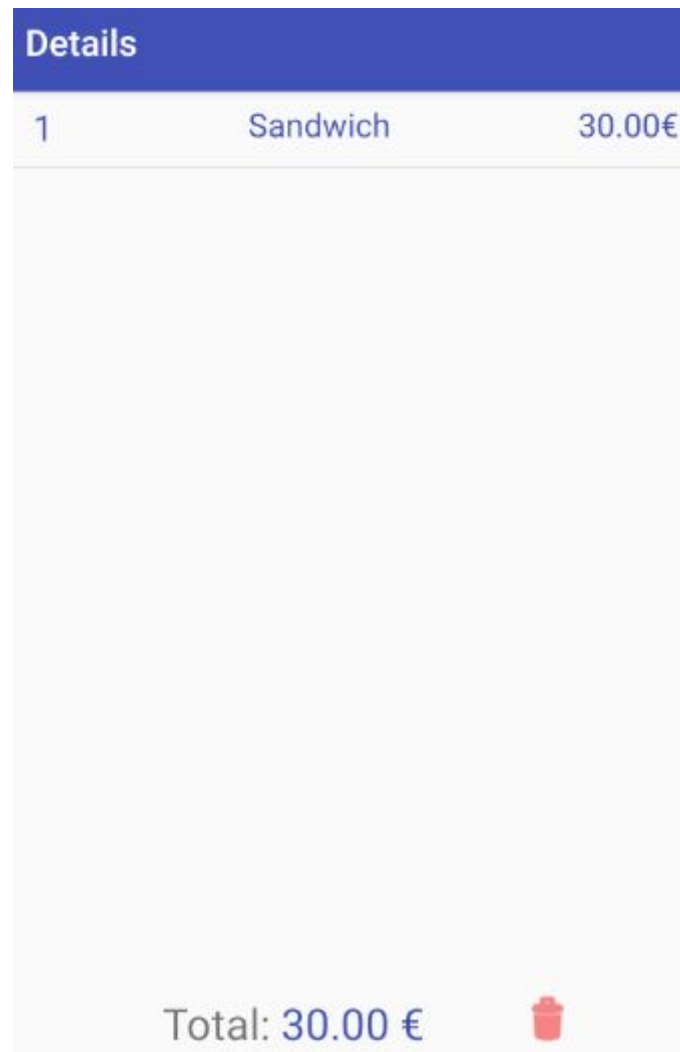


Image 11 - Previous transaction details page

3.7 Available Vouchers

On this page the user can see the information about his available vouchers. Name and discount are displayed.

Vouchers	
Coffee Voucher 1	20.0€
Coffee Voucher 2	20.0€
Normal Voucher 1	5%
Normal Voucher 2	5%

Image 12 - Available Vouchers page

3.8 Menu

In the menu page the user can see the available products on the shop as well as their prices. The user can also select the amount of each product he wants to buy and then proceed to the checkout.



Image 13 - Menu page

3.9 Checkout

On this page the user can see the products we choose to buy, the total price of the transaction and select a voucher to have a discount. Coffee vouchers can only be selected if the user chooses to buy at least one coffee. The total amount with and without discount is displayed and the user can ask for the qr code that will be displayed in the center of the screen.

New Transaction

1	Sandwich	30.00€
2	Orange Juice	100.00€

SELECT A VOUCHER

Normal Voucher 1

Total: **130.00 €** w/ discount: **123.50**

BACK

QR CODE

Image 14 - Checkout page

3.10 Checkout transaction (terminal side)

Upon reaching the terminal, the user is prompted to scan a new QR code to proceed, and can do so by pressing the “+” button.

When doing so, it enables the camera, which must be pointed at a valid order QR code. If so, it'll display the order information, and enable a button to confirm the order. Upon pressing it, it returns with the Order number which the user will have to wait for, or an error message, with a button to return.

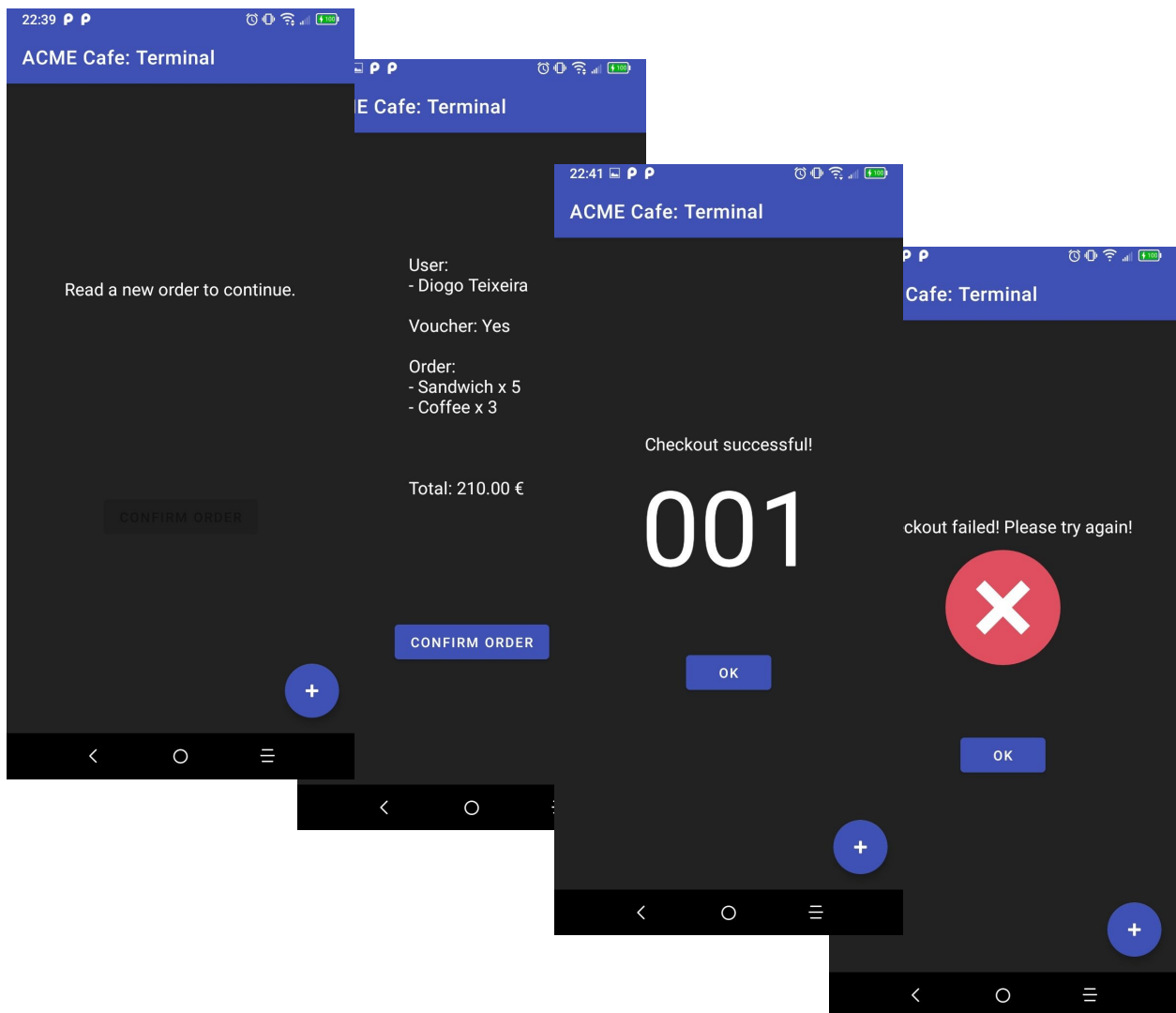


Image 15 - Terminal flow

3.11 Saved Data

The user's information is saved locally in the device when they login.

When the user enters the app, if their information is stored, it will automatically log in and enter the app.

4. Performed Tests

In order to test the project, a database with some data was created and used to test all functionalities.

Throughout the app, validations are made to ensure stability, such as username/password combination, registering repeated users and checking out an empty basket.

```
.then(() => {
  User.bulkCreate([
    {id: "b0e76929-9762-45b7-be1f-2f37d2edf33c", email: "fabio@gmail.com", username: "Fabio", name: "Fábio Azevedo", password: "4sPYOZ0HJo3u0DnnDZhuYh0hLWJ1iL57Xmpj5TRE", transactionId: null},
    {id: "32bf576f-1d83-4141-9009-8d4c6435d10e", email: "diogo@gmail.com", username: "Diogo", name: "Diogo Teixeira", password: "jxRlCnQCMoGKrhyVysaZIpIlTas1ft+K8ilxM22", transactionId: null}
  ])
})
.then(() => {
  Transaction.bulkCreate([
    {id: "82d7e", voucher: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935ad", total_value: 152, discount: 8, UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e"},
    {id: "82d7f", total_value: 30, discount: 0, UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e"},
  ])
})
.then(() => {
  Product.bulkCreate([
    {id: "1f45a", name: "Coffee", value: 20.00, icon_path: "coffee.png"},
    {id: "6e89r", name: "Sandwich", value: 30.00, icon_path: "sandwich.png"},
    {id: "1h52w", name: "Milk", value: 40.00, icon_path: "milk.png"},
    {id: "1a25w", name: "Orange Juice", value: 50.00, icon_path: "orange_juice.png"}
  ])
})
.then(() => {
  Voucher.bulkCreate([
    {id: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935ac", used: false, coffee: "false", UserId: "b0e76929-9762-45b7-be1f-2f37d2edf33c", TransactionId: null},
    {id: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935ad", used: true, coffee: "false", UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e", TransactionId: "82d7e"},
    {id: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935af", used: false, coffee: "true", UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e", TransactionId: null},
    {id: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935ae", used: false, coffee: "true", UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e", TransactionId: null},
    {id: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935ag", used: false, coffee: "false", UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e", TransactionId: null},
    {id: "b1a3k3v1-cte1-4ab3-b1ce-67cf4d3935ah", used: false, coffee: "false", UserId: "32bf576f-1d83-4141-9009-8d4c6435d10e", TransactionId: null},
  ])
})
.then(() => {
  TransactionProduct.bulkCreate([
    {id: "7a9af", ProductId: "6e89r", TransactionId: "82d7f", count: 1},
    {id: "2a8ac", ProductId: "1f45a", TransactionId: "82d7e", count: 2},
    {id: "835ac", ProductId: "6e89r", TransactionId: "82d7e", count: 4},
  ])
})
```

Image 16 - Populated database

5. Setup

To setup the project for development, it is expected to have Android Studio and Node.js installed in the machine. Afterwards:

- ❖ Open the Client and Terminal folders in separate Android Studio windows.
- ❖ Open a terminal inside the Server folder.
 - Run **npm install**. This will install all dependencies.
 - Run **npm database**. This will start the database and fill with some data.
 - Run **npm start**. This will run the server.
- ❖ Execute the programs in Android Studio, connecting mobile devices to the machine.
- ❖ In order for the client application to be able to reach the server then the *ip_address* variable inside Utils.java must be changed to the ipv4 address of the user before compiling and running the project.

6. Conclusions

In conclusion, this project allowed us to explore the multiple sides of Android development, both functional and design wise, while also applying the acquired knowledge of Java and Javascript, which had been taught in previous semesters.

On the other hand, we came to the conclusion that some information available on the slides provided by this class was outdated, but that was easily overcome by reading the Android documentation and guides.

On a final note, although none of the members of the group had experience in Android development, the process as a whole was smooth and the team worked well, as any setback encountered was rapidly solved.

Bibliography

<https://developer.android.com/>

<https://www.tutorialspoint.com/android/index.htm>