

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO
PORTO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E
COMPUTAÇÃO

MOBILE COMPUTING

Mobile Computing Project Report

Autores:

Diogo REIS - up201505472

Tiago MAGALHÃES - up201607931



Universidade do Porto

Faculdade de Engenharia

FEUP

November 2019

Contents

1	Architecture	3
1.1	Server	3
1.1.1	REST WebAPI	3
1.1.2	Database	3
1.1.3	RSA Encryption & Signing	3
1.2	Store And Client Applications	3
1.2.1	HTTP Requests	3
1.2.2	QR Codes	3
1.2.3	RSA Encryption & Signing	4
1.2.4	Data Caching	4
2	Data Scheme	5
2.1	Database Data Schema	5
2.1.1	Client	6
2.1.2	Voucher	6
2.1.3	Purchase	6
2.1.4	Product	7
2.2	Checkout Information	7
2.3	Data Verification Signatures	8
3	Features & Tests	9
3.1	Features	9
3.1.1	Available Product List In Store App	9
3.1.2	User Registering	9
3.1.3	User Login	9
3.1.4	User Login Caching	9
3.1.5	Adding Products To Cart Through QR Codes	9
3.1.6	Viewing and Editing Cart	9
3.1.7	Viewing Products Bought In Past Transaction	9
3.1.8	Viewing Vouchers Available To User And Amount Of Money Usable For Discounting	9
3.1.9	Checkout	10
3.2	Testing	10
4	Usage Manual	11
4.1	Server	11
4.2	Store Application	11
4.2.1	Viewing Available Product List	11
4.2.2	Checkout	12
4.3	Client Application	12
4.3.1	User Registering	12
4.3.2	User Login	13
4.3.3	User Login Caching	14

4.3.4	Adding Products To Cart Through QR Codes	15
4.3.5	Viewing and Editing Cart	16
4.3.6	Viewing Past Transaction	17
4.3.7	Viewing Products Bough In Past Transaction	18
4.3.8	Viewing Vouchers Available To User And Amount Of Money Usable For Discounting	19
4.3.9	Checkout	20
5	References	22

1 Architecture

1.1 Server

1.1.1 REST WebAPI

The server uses a REST WebAPI provided by the ASP.NET framework to handle all the communication with it. All routes are present in a single controller under the router of server.

1.1.2 Database

Communication with the PostgreSQL database is done using the Npgsql package, it is then abstracted into a singleton class that provides the common insert and select operation. Prepared statements are done via an entry system where each entry has a name, a value and a boolean indicating if the entry is a UUID or not, the entry names are then matched to the query markers for parameters.

All responses to select queries are done via list of dictionaries that map strings to object, each element in the list is a row in the database, and the elements of the dictionaries contain the values of the various columns in the database, the entries have the same name as they do in the database schema.

1.1.3 RSA Encryption & Signing

Encryption is done using the .NET Crypto Service Providers, specifically RSA Crypto Service Provider, as well as the BouncyCastle package that is used for parsing and exporting public and private keys to and from the PEM string format. This is also done with a singleton class that exposes Encryption, Decryption, Signing and Verifying data methods.

Any encrypted data or signatures are in Base64 format, while all data to be verified or encrypted are expected to be in a Unicode UTF-16 encoding.

1.2 Store And Client Applications

1.2.1 HTTP Requests

All HTTP requests are done via static methods that make use of standard Java facilities such as URL and HttpURLConnection. All requests are run in an Android AsyncTask so as not to block the UI main thread of Android. Each request method also receive an instant of a class called HTTPResultHandler, this class handles the result of the HTTP request once it is ready, and this is done via its Handler method.

1.2.2 QR Codes

QR Code handling is done via the Google Zebra Crossing (zxing) library and some of its android integrations, methods are provided for generating a QR code and for reading a QR code.

QR code generation is rather straightforward with a method call that returns a Bitmap.

QR code reading is a bit more involved as it makes use of the external bar code scanner application and requires the activity to override its onActivityResult method, therefore to simplify

this system and force QR code reader users to properly overload this method we have created an Android Activity that is abstract and forces its subclasses to implement a handler that it will internally call when the onActivityResult method is triggered.

1.2.3 RSA Encryption & Signing

RSA encryption and signing is done via the standard java facilities for security purposes and Androids KeyStore facilities. When a new user is registered a new private key public key pair is introduced into the Android Secure Key Store Enclave.

All further usages of this class are done via static methods that when needed receive the username of the user, so that they can fetch his key for various security purposes.

This class offers static methods for encryption, decryption, signing and verifying data.

1.2.4 Data Caching

All data caching is done via the Android SharedPreferences Facilities.

2 Data Scheme

2.1 Database Data Schema

```
CREATE EXTENSION IF NOT EXISTS "uuid-oss";

DROP TABLE IF EXISTS Client CASCADE;
CREATE TABLE Client(
    id uuid primary key default uuid_generate_v4(),
    name text not null,
    username text not null,
    password text not null,
    credit_card int not null,
    public_key text not null,
    current_total_spent_euro INTEGER not null default 0,
    current_total_spent_cent INTEGER not null default 0,
    current_accumulated_euro INTEGER not null default 0,
    current_accumulated_cent INTEGER not null default 0
);

DROP TABLE IF EXISTS Voucher CASCADE;
CREATE TABLE Voucher(
    id uuid primary key default uuid_generate_v4(),
    client uuid not null REFERENCES Client(id),
    was_used BOOLEAN not null DEFAULT FALSE
);

DROP TABLE IF EXISTS Purchase CASCADE;
CREATE TABLE Purchase(
    id uuid primary key default uuid_generate_v4(),
    client uuid not null REFERENCES Client(id),
    voucher uuid REFERENCES Voucher(id) DEFAULT NULL,
    should_discount BOOLEAN not null DEFAULT false
);

DROP TABLE IF EXISTS Product CASCADE;
CREATE TABLE Product(
    id uuid primary key default uuid_generate_v4(),
    price_euro INTEGER not null,
    price_cent INTEGER not null,
    name text not null,
    image_url text DEFAULT NULL,
    purchase uuid REFERENCES Purchase(id) DEFAULT null
);
```

2.1.1 Client

This table contains information about users of the platform and has the following fields:

- id - UUID representing the identification of the user.
- name - String representing the name of the user.
- username - String representing the username or nickname of the user.
- password - String representing the password of the user.
- credit_card - Integer representing the credit card number of the user.
- public_key - String containing the users RSA public key.
- current_total_spent_euro - The amount of money spent by the user, this is the euro component.
- current_total_spent_cent - The amount of money spent by the user, this is the cent component.
- current_accumulated_euro - The amount of money the user has accumulated from voucher, this is the euro component.
- current_accumulated_cent - The amount of money the user has accumulated from voucher, this is the cent component.

2.1.2 Voucher

This table contains the voucher registered in the system and has the following details:

- id - UUID representing the identification of the voucher.
- client - UUID representing the identification of the user the voucher belongs to.
- was_used - Boolean representing whether or not the voucher has been used, by default this value is false.

2.1.3 Purchase

This table contains information about purchases and has the following details:

- id - UUID representing the identification of the purchase.
- client - UUID representing the identification of the user the purchase is associated with.
- voucher - UUID representing the identification of the voucher used in this purchase, this value is optional and is null by default.
- should_discount - Boolean representing whether or not the cost of the purchase was amortized with money the user had accumulated via vouchers.

2.1.4 Product

This table contains the products registered in the system and has the following information:

- id - UUID representing the identification of the product.
- price_euro - This is the price of the product, this is the euros component.
- price_cent - This is the price of the product, this is the cents component.
- name - String representing the name of the product.
- image_url - String representing the link to an image of the product.
- purchase - UUID representing the purchase this product is in, this is optional and is null by default.

2.2 Checkout Information

Checkout information is represented in the following format:

```
{  
  products: ['product_id1', 'product_id2', ...],  
  user_id: 'user_id',  
  use_discount: true|false,  
  sign: 'sign',  
  voucher_id: 'voucher_id'  
}
```

These fields have the following meanings:

- products - Array with UUID ids of products to be purchased.
- user_id - UUID id of the user executing the purchase.
- use_discount - Boolean indicating weather or not the user wants to use the money he has saved with vouchers to reduce the cost of the purchase.
- sign - sign of the configurations of the purchase, to verify the authenticity of the user executing the purchase.
- voucher_id - UUID id of a voucher to be used this parameter is optional as the user is not forced to use a voucher.

2.3 Data Verification Signatures

The products list needs to be verified so each product that comes from the server has its json data pre the addition of the sign parameter signed with the servers private key, the applications then can verify that the data contained in the QR code and received from the server are indeed valid as they have access to the server public key to verify the signature of the product data.

For the checkout we need to sign the data in a way that the server can then verify that the issues of the data is indeed who he says he is, to do this we take the optional parameters of the request(voucher_id and weather or not to use the discount accumulated) and the user id and concatenate and then produce a sign from this data. The reason we do not sign the product array is that depending on the size of the RSA key used there will be an upper limit to the size of the data we can encrypt or sign, so this would put a cap on the amount of products we can have in a single transaction, so for that reason we have opted to not include the array that can vary in size and use only the static elements of the request that we know will not cause an issue with out 512 key.

It is also notable to state the increasing the key size would allows us to encrypt or sign more data, but would create a large issue with the QR code system, as the bitmaps would become far too complex for the average phone camera to be able to decode with any semblance of precision and accuracy. As it stand 512 bits for the key seems to be the upper limit of what is reasonable due to the result signature or encrypted data size and how those interact with QR code complexity.

This leads us to conclude that QR codes do not scale well and are an unfeasible way to implement this type of system and a NFC based solution would be more much feasible, however many phones still don't have NFC support, including our own, making us unable to implement such a solution.

3 Features & Tests

3.1 Features

3.1.1 Available Product List In Store App

In the store app there is a list of all products currently available in the system along with their prices, if a user click the product he will be displayed the corresponding QR code. All products are signed by the server to validate they have been issued by a trusted source.

3.1.2 User Registering

Users have the ability to register via the client application, they need to provide their name, username, password and credit card, the user will then be assigned an UUID identifier and a public and private key will be generated for him and security stored in Androids Secure Key Store.

3.1.3 User Login

The user is able to login into his account by providing his username and password, this login is merely local and servers to retrieve the server keys and the user data. If a user registers with the same username as a previously existing user, he will overwrite that users local data.

3.1.4 User Login Caching

User login data is cached so that the user only needs to login in case he explicitly logs out of the system.

3.1.5 Adding Products To Cart Through QR Codes

By clicking a button and downloading the bar code scanner application the user can scan a QR and add a product to his cart.

3.1.6 Viewing and Editing Cart

The user is able to view the products in the cart and remove them from the cart. When viewing transactions the user is also able to view the items he bough in that.

3.1.7 Viewing Products Bough In Past Transaction

If a user has internet connectivity he is able to fetch his past purchases in the platform from the server. When viewing transactions the user is also able to view the items he bough in that.

3.1.8 Viewing Vouchers Available To User And Amount Of Money Usable For Discounting

At any time the user can fetch the vouchers he has associated to himself in the server that have not been used.

3.1.9 Checkout

The user can generate a checkout QR code that he can validate in the store application to execute his purchase. When using a voucher the system will automatically select the first voucher that it can use as the choice of voucher does not matter. So this is done merely through a toggle.

3.2 Testing

Testing of the several apps has been done in acceptance test format, listing out what each feature needs to do and proceeding to manually test them, attempting to cover all possible edge case scenarios.

4 Usage Manual

4.1 Server

To run the server you will need the dotnet core 2.1 kits and a PostgreSQL database running, the database must contain a database called cmovdb and must have the schema presented in this documented on it, you must also have a user with username cmov and password cmov. That being said these are the default and this can be configured in the server files.

To start the server all you need to do is run the command `dotnet run` in the server folder.

In the server file `startup.cs` you can configure the IP address and Port that the server will run on. On the `program.cs` file you can configure the IP address, port, username, password and database server database that the store server will connect to and with.

4.2 Store Application

4.2.1 Viewing Available Product List

To view the product list the user must click the "Product List" button in the store main menu, he will then be presented with the full list of available products in the store, by clicking any of the products in the list the user will be able to view its QR code that it can add it to his cart.

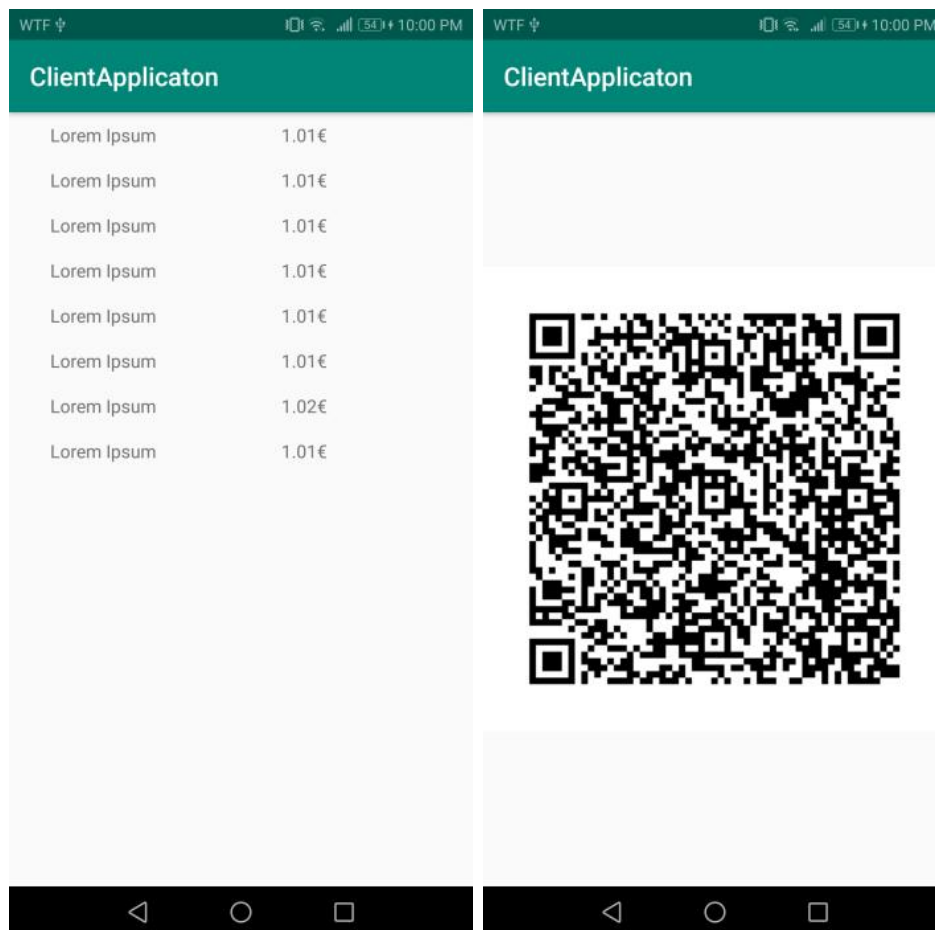
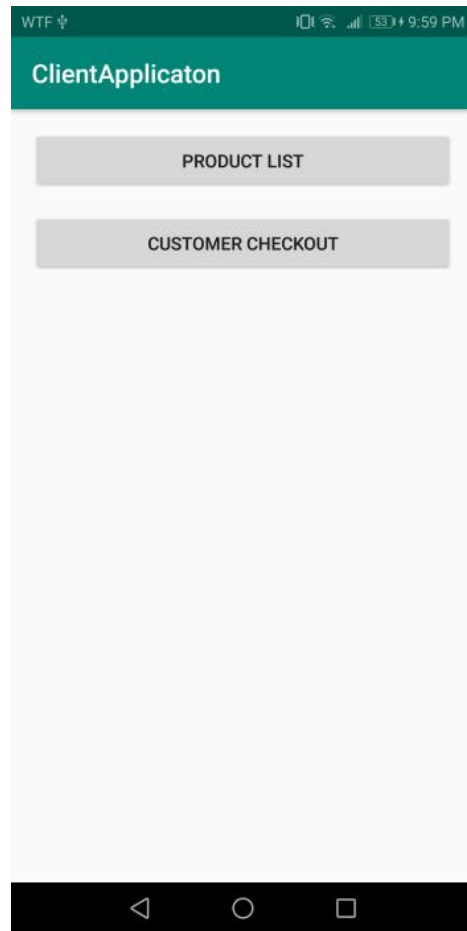


Figure 1: Product List and Viewing QR Code From Product In List

4.2.2 Checkout

To perform the client checkout we need to click the "Customer Checkout" button in the store main menu, the camera will then open up and allow us to scan the QR code, once that is done the purchase will be sent to the server and processed.

**Figure 2:** Store Main Menu

4.3 Client Application

4.3.1 User Registering

The user can register in the system by clicking the Register Button in the client app authentication menu, he will then be taken to the register view, where he will be able to input his data and once registered in the server will be redirected to the client app main menu.

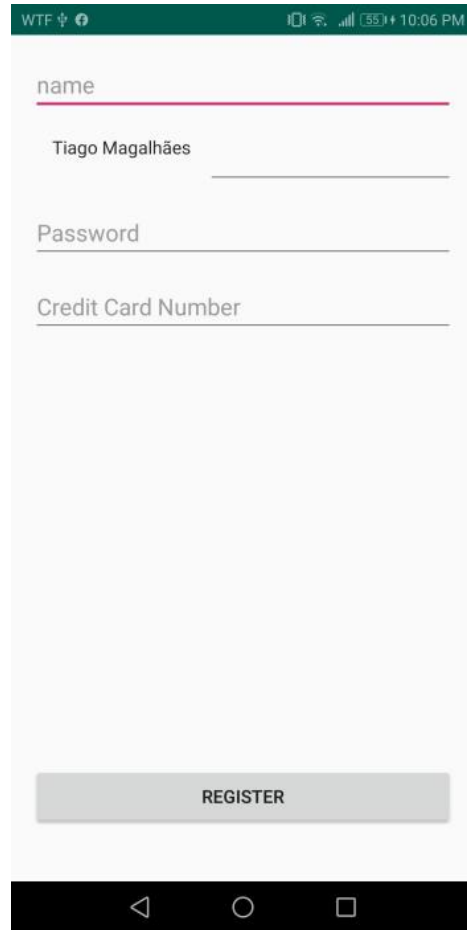
A screenshot of a mobile application's 'Register' page. The page has a white background with a dark green status bar at the top. The status bar shows 'WTF' as the carrier, signal strength bars, a battery icon, and the time '10:06 PM'. Below the status bar, there are four input fields: 'name' (with a pink underline), 'Tiago Magalhães' (with a grey underline), 'Password' (with a grey underline), and 'Credit Card Number' (with a grey underline). At the bottom of the form area is a grey button labeled 'REGISTER'. The bottom of the screen shows a black navigation bar with three white icons: a back arrow, a circle, and a square.

Figure 3: Register Page

4.3.2 User Login

If a user has logged out of the system but want to log back in he can click the login button in the client app authentication menu, he will be taken to the login page where he can input his username and password, if that matches the application credential database he will be taken to the client application main menu

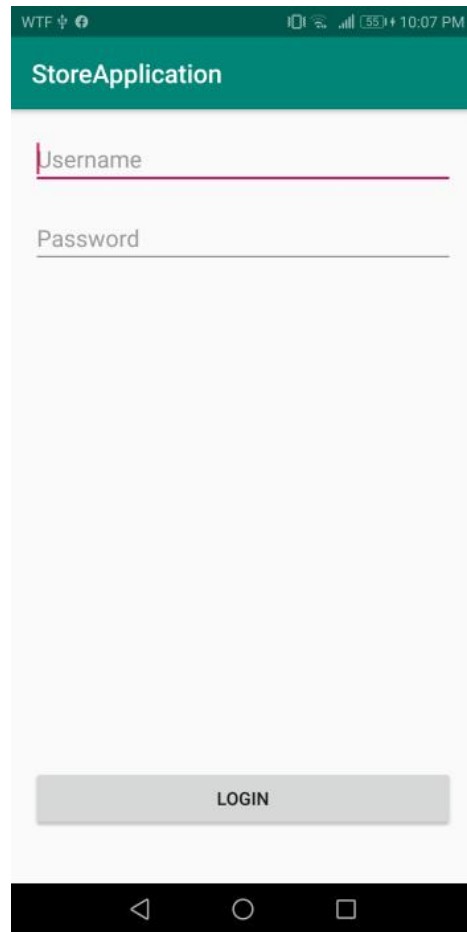


Figure 4: Login Page

4.3.3 User Login Caching

If the user is currently signed in on the client application, when the application opens it will verify this and if the user is logged in he will be instantly redirected to the client application main menu without having to go through the manual login process.

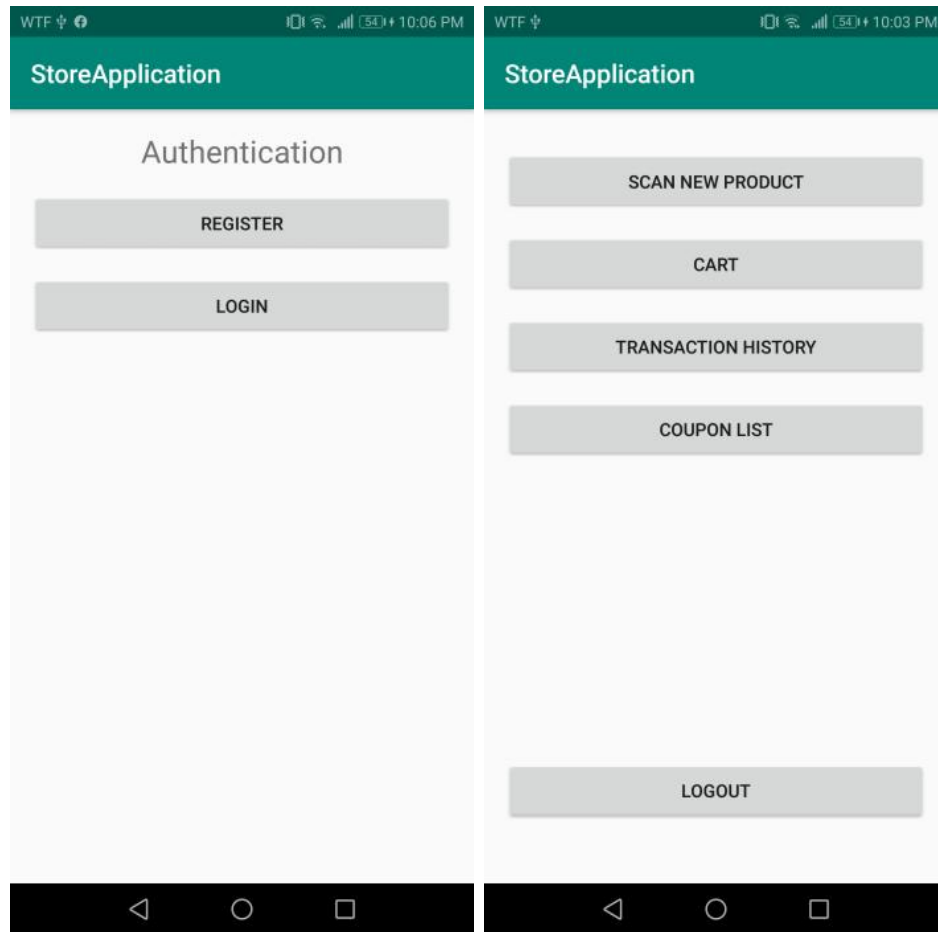


Figure 5: User Login Caching Flow

4.3.4 Adding Products To Cart Through QR Codes

To add a product to the cart the user merely needs to click the scan new product option in the client application main menu, the camera will then open up and allow the user to scan a new product into the cart.

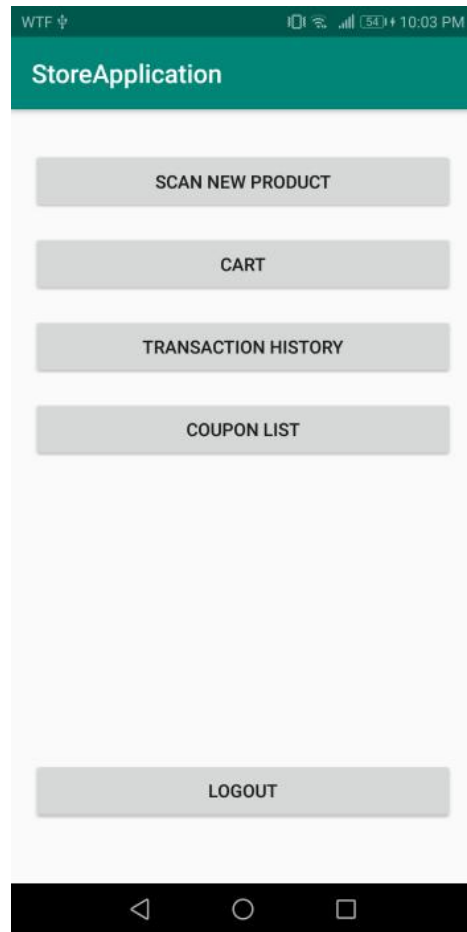


Figure 6: Client Application Main Menu

4.3.5 Viewing and Editing Cart

The user can view his cart by clicking the Cart button in the client application main menu, he will then be taken to a page with his cart, a list of the products in it and the current price of all those products put together, on any of those products, he can click the remove button to remove them from the cart.

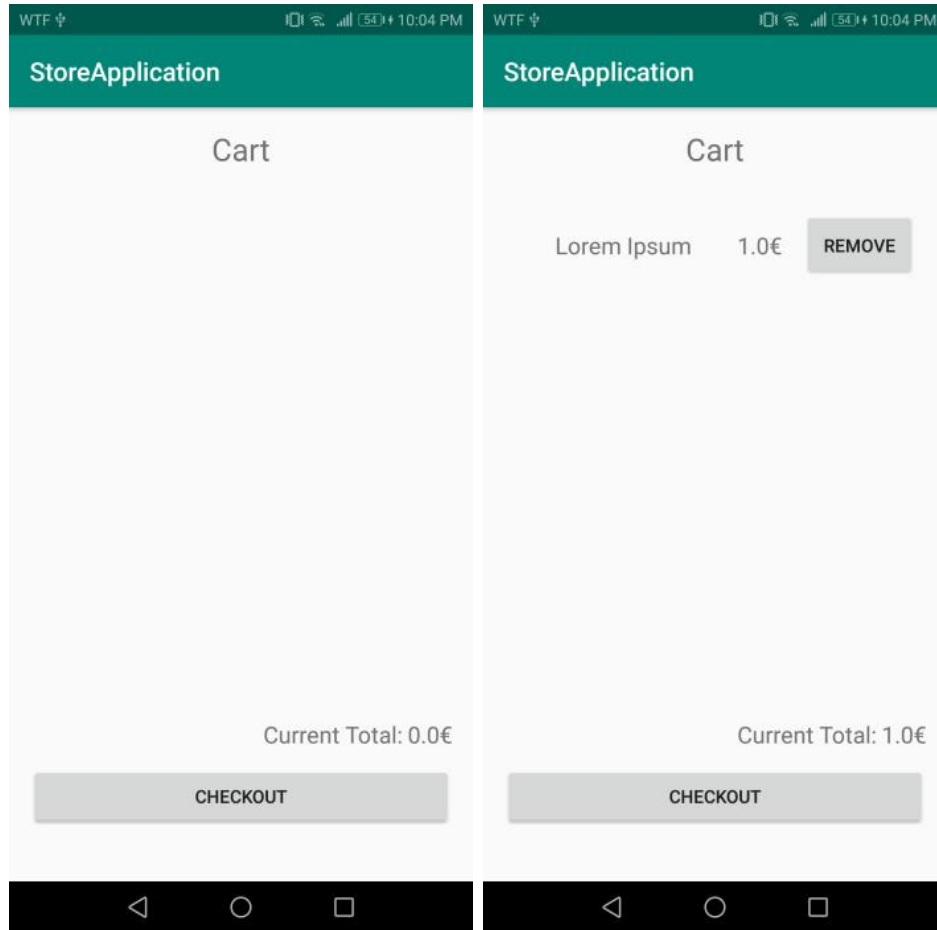


Figure 7: Client Application Main Menu

4.3.6 Viewing Past Transaction

At any time the user can view his past transactions by click the Transaction history button, then he will be taken to a page with a list of all the transactions he has made in the system.

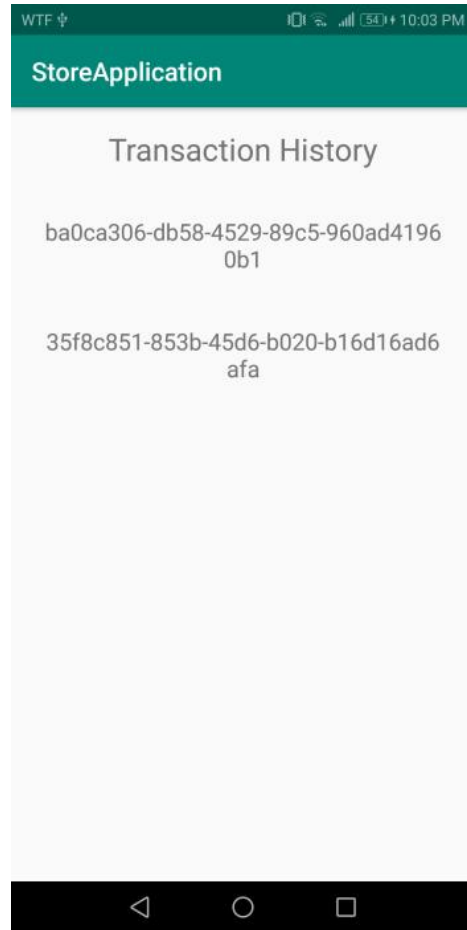


Figure 8: Client Transaction History

4.3.7 Viewing Products Bought In Past Transaction

If the user clicks an item in the transaction history list, he will get a list of all the items he purchased in that transaction.

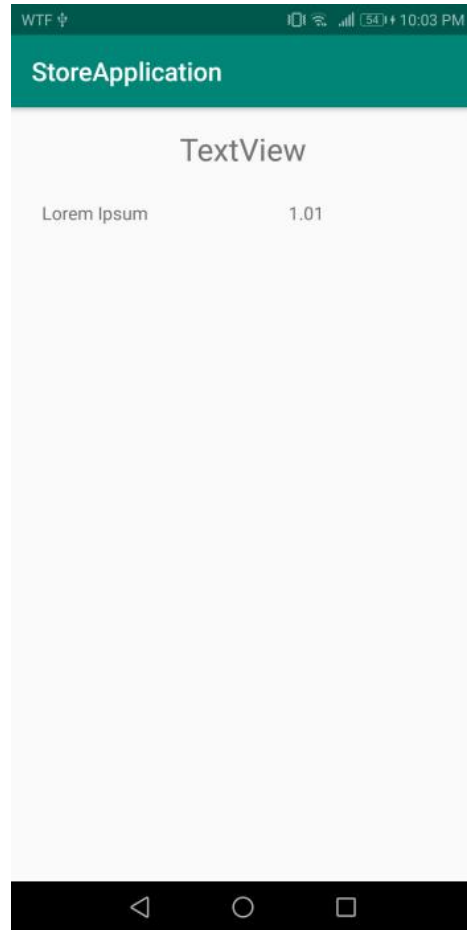


Figure 9: Products Bought In Purchase

4.3.8 Viewing Vouchers Available To User And Amount Of Money Usable For Discounting

The user can at any time view the vouchers he has available for use by clicking the Coupons button, he will then be taken to a page with a list of all the coupons he has left to use.

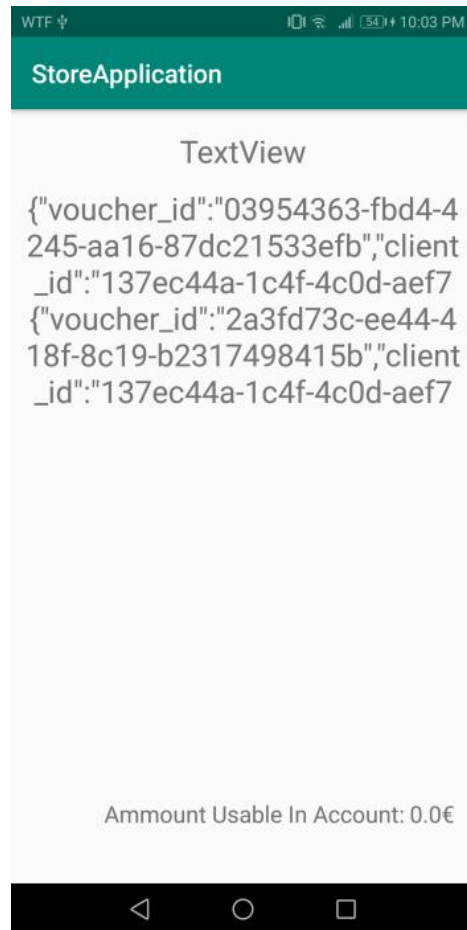


Figure 10: Client Available Voucher List

4.3.9 Checkout

To checkout the user needs to go into his cart and click the checkout button at which point he will be taken to the checkout page where he can configure the purchase and generate a QR code that he must present to the store app in order to execute the purchase.

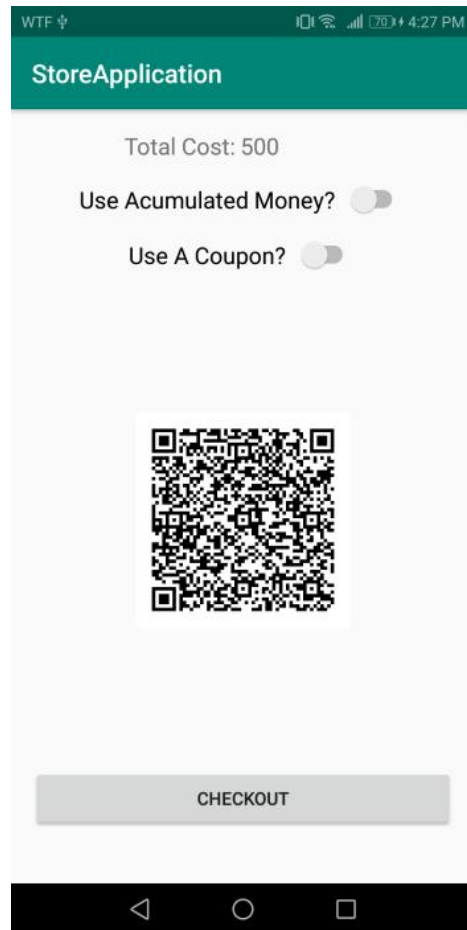


Figure 11: Checkout View

5 References

There are no meaningful nor notable references, Android, Java, C#, .Net and BouncyCastle documentation were used heavily throughout the project though.