

ACME Emarket

A shop and pay Android app for a supermarket

Practical Assignment 1



Faculdade de Engenharia da Universidade do Porto

Mobile Computing - M.EIC

Carolina Albuquerque - 201906847

Henrique Nunes - 201906852

Rui Alves - 201905853

Table of contents

Introduction.....	1
Architecture.....	2
Data Schemas.....	3
System Features.....	4
Emarket Server.....	4
Emarket Customer.....	5
Activities.....	5
<i>Splash Screen</i>	5
<i>Registration</i>	6
<i>Login</i>	7
<i>Basket</i>	7
<i>Checkout</i>	9
<i>Payment QR Code</i>	10
<i>Payment NFC</i>	11
<i>Profile</i>	12
<i>Profile Tab</i>	12
<i>Transactions Tab</i>	13
<i>Transaction details</i>	15
<i>Settings</i>	16
Notifications.....	17
Emarket Terminal.....	18
Activities.....	18
<i>Main</i>	18
<i>Result</i>	19
Representations.....	20
Checkout Information Representation.....	20
Testing.....	21
Get User Information.....	21
Update User Information.....	21
QR Code Reading.....	21
Payment QR Code or NFC.....	21
How to run.....	22
Emarket Server.....	22
Emarket Customer.....	22
Emarket Terminal.....	22
Conclusion.....	23
Appendix.....	24
Landscape layouts.....	24
Emarket Customer.....	24
<i>Splash Screen</i>	24
<i>Registration</i>	25
<i>Login</i>	25

<i>Basket</i>	26
<i>Checkout</i>	26
<i>Payment - QRCode</i>	27
<i>Payment - NFC</i>	27
<i>Profile</i>	28
<i>Profile Tab</i>	28
<i>Transactions Tab</i>	29
<i>Transaction details</i>	29
<i>Settings</i>	30
Emarket Terminal.....	30
<i>Main</i>	30
<i>Result</i>	31

Introduction

In recent years, mobile devices, such as smartphones and tablets, have become an essential part of an individual's daily life. As a result, mobile development has become increasingly important for businesses looking to reach and engage with their customers. Mobile apps have proven to be an effective way to provide a personalized user experience, increase customer engagement and loyalty, and drive business growth.

Acme Supermarket is taking a step towards giving its customers a more efficient and engaging shopping and payment experience by developing an autonomous shopping system. This system provides customers with an Android app, which allows them to shop at ACME without any need for external intervention.

The Customer App allows customers to register themselves in the system, consult past transactions and retrieve vouchers, add items to the shopping basket by scanning a tag or QR-code in each product, and perform the checkout, possibly adding an already retrieved voucher and discounting the accumulated amount from previous voucher uses. Additionally, the checkout Terminals at the supermarket also run an Android application that receives the items list and any vouchers used on the transaction from the Customer App. The Terminal communicates this information to the Server, which executes the payment and transmits the operation result and total value charged back to the Terminal. The Terminal then opens the gate doors, allowing customers to exit the supermarket.

Architecture

The elaborated system contains three main applications, the *Emarket Server*, the *Emarket Customer* and the *Emarket Terminal*. The first one is a REST API developed in **Flask (Python)** and it is essentially the brain of the system as it provides an interface for the actions that can be performed. This application uses a **MongoDB** database. The *Emarket Terminal* is a **Kotlin** application that makes the bridge between the Customer application and the Server, by simply reading a QR Code or an NFC tag and redirecting the data to the server by *HTTP*. It then receives an answer to open the gates or to keep them closed. The *Emarket Customer* is also a **Kotlin** application and it uses an **SQLite** database to store information. It can communicate directly with the server by *HTTP* or, for the specific case of the checkout, through the Terminal application. A component diagram of the system can be found in Fig. 1.

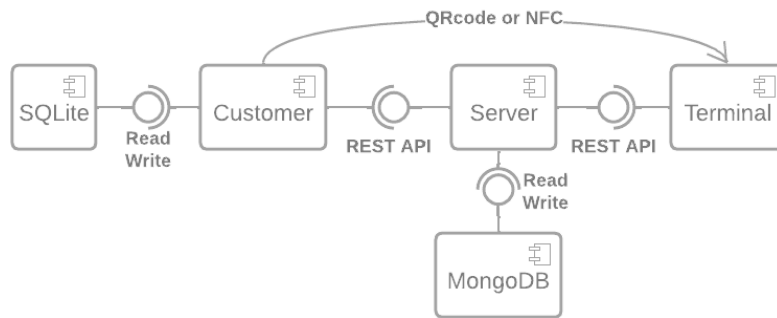


Figure 1. Components diagram of the ACME Emarket system

Data Schemas

The ACME Emarket system can store information on several users. Each user has some specific attributes and past transactions, which are associated with certain products and their respective quantities. The user can also have vouchers, which, if already used, should be associated with a single transaction. The data schema of the system for a specific user is represented in Figure 2. In the Server database, each user is represented by a **document** in a **collection** named “*users*”. The Server also has a collection named “*products*” which stores all the products registered in the supermarket. On the Customer side, the self user information is stored on the *Android Shared Preferences*, since a Customer application can only have one user at the time. The *Transactions*, the *Vouchers*, and the *Products* are stored on the corresponding **table** on the SQLite database.

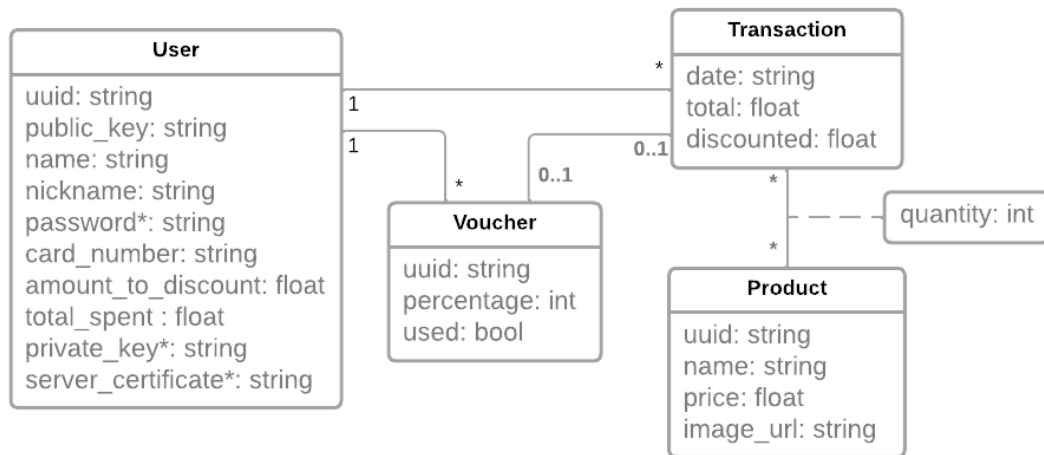


Figure 2. Data schema of a user. The attributes followed by an “*” are only saved on Emarket Customer.

System Features

In this section the main features of the three developed applications will be discussed.

Emarket Server

The Server's REST API contains five routes available. These routes are well documented within a swagger route, available on route **GET /swagger**.

The route **POST /register** allows a user to register in the system by sending the server their public key, name, nickname and credit card number. The user is stored on the database and the server responds to them with the server's public key and the UUID generated for the user, encrypted with the user's public key to ensure that only the user receives that UUID.

The route **POST /checkout** is used by the Terminal to process the user's payment. The content of the request should be a byte array where the first 64 bytes are the signature of the rest of the content of the array. A more detailed explanation on the format of the byte array expected can be found in section [Checkout Information Representation](#). The Server verifies the signature and processes the payment order, creating the user's transaction, updating the old and new user's vouchers and updating the amount to discount. If all goes as expected, the response to the Terminal's request should contain the total amount paid by the user, otherwise the payment is aborted and an error message is sent.

The route **GET /user** allows the user to retrieve and update their own information like the amount to discount, total amount spent, unused vouchers and past transactions. This route demands two query parameters, one with the user UUID and another with the signature for the user UUID which is verified before the request is processed. This route has another particularity, it allows the user to send one more query parameter named "date" which should contain the date of the last transaction the user has in their database so that the Server can send only the subsequent transactions.

The route **POST /user** allows the user to update their name and/or card number by sending them, alongside with their UUID, in the request body, signed with their own private key.

The route **GET /product/<uuid>** simply retrieves the complete product (UUID, name, price and image url) with the UUID indicated on the request.

Emarket Customer

This application is provided to the ACME customers and enables them to make purchases in the supermarket, making them a part of the system already described.

Activities

All the activities presented in this work have been developed to support screen rotation in a user friendly and visually pleasant way. Additionally, the app was developed to support two customized color themes, that are defined based on the user's device settings (light and dark mode). These layouts are presented below.

Splash Screen

Upon launching the application, users are greeted by a welcoming screen that is shown for two seconds. The screen, as depicted in Figure 3, features the application's icon along with a loading spinner, which represents the initialization process of the application. The welcoming screen is designed to create a pleasant and inviting user experience.

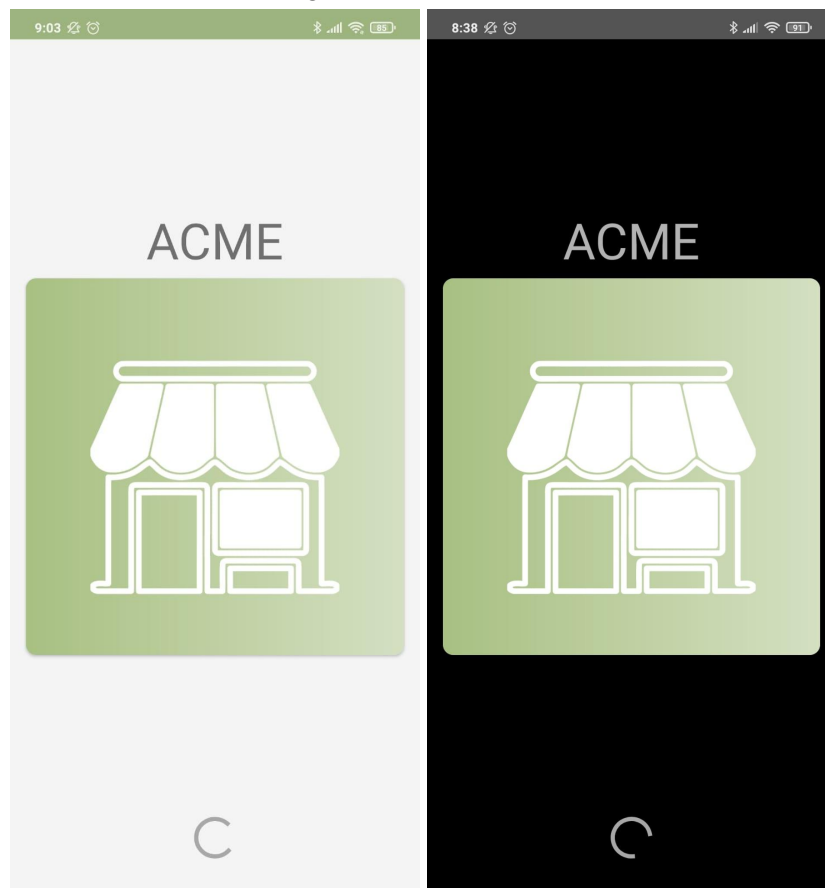


Figure 3. Emarket Customer Splash Screen

Registration

When a customer installs the ACME app it is required for them to **register**, inputting personal and payment information as it can be seen on Figure 4. When the user clicks the “Register” button, a **cryptographic key pair is generated** on the AndroidKeyStore and the generated Public Key and user’s name, nickname and card number are sent to the server. The **server responds with** a user **UUID** (which it generated) and with its Certificate which contains the **Public Key**.

While the user is waiting for the server’s response, the “Register” button is replaced by a **loading spinner**.

The information inputted by the user and the information sent by the server is saved on the *Shared Preferences*. For security reasons, the **password is hashed** before being saved. The user’s card number should be in the format “xxxx-xxxx-xxxx-xxxx”, where the “x” represent a number between 0 and 9. The user is required to input the 16 digits, and the ‘-’ **separator is filled automatically** at each 4 digits. All the **fields are validated** according to their expected input size (e.g. the card number should have exactly 19 characters and no input field should be empty).

At the end, if all went as expected, the user is redirected to the Login page.

The figure displays two side-by-side screenshots of the ACME app's Registration screen. Both screens feature a green header with the text 'ACME'. The left screenshot shows the registration form with pre-filled data: 'Name: Carolina', 'Nickname: Carol', 'Password: [masked]', and 'Card no.: 1373-5164-3451-6434'. The right screenshot shows the same form with empty input fields. Both screens have a green 'Register' button at the bottom.

Figure 4. Emarket Customer Registration Activity

Login

Whenever a user starts the app, if already registered, they are prompted to input the nickname and password set on the registration, as it is presented on Figure 5. This information is used to perform a **local login** that authenticates the user. If the user is successfully authenticated, the application **requests the server** for the user's personal information, such as past **transactions**, unused **vouchers**, **total spent** amount, and **amount to discount** and updates its local database with it. When the connection with the server is not reached the application continues with the information previously stored in the database. Either way, the authenticated user ends up in the Basket activity.

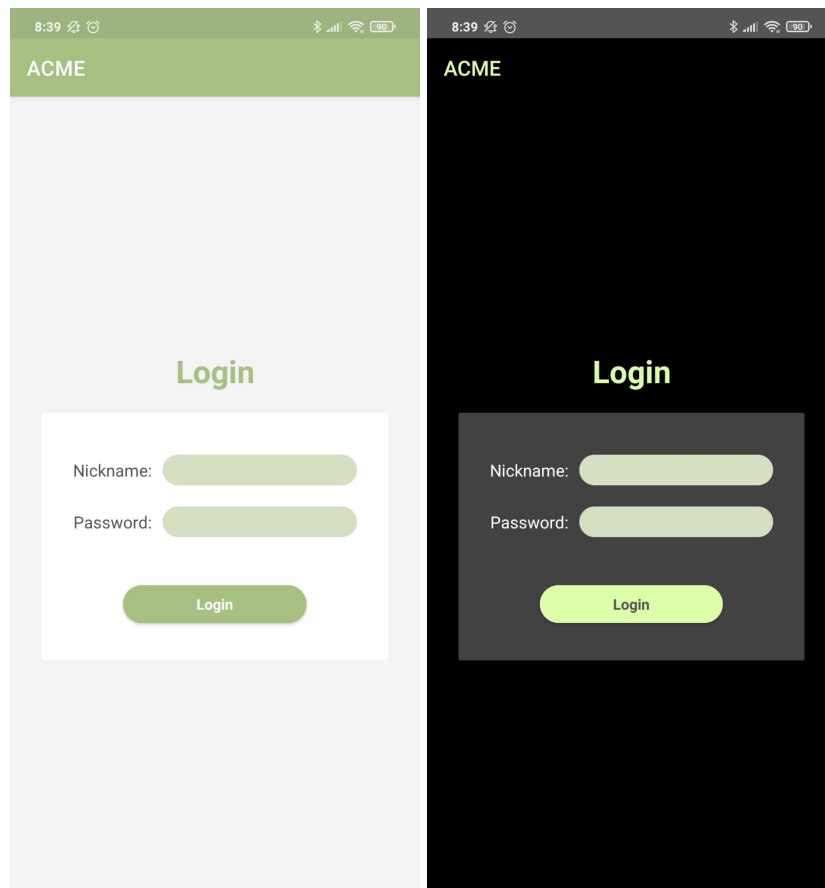


Figure 5. Emarket Customer Login Activity

Basket

After the login operation is concluded, the Basket Activity is started. On this page, the user can **add items** to their shopping basket by **scanning their QR Code**. The scanner to read the products' QR Codes can be reached by **clicking the “plus” button** or **shaking the device**. Whenever a new product is added, its existence on the basket is checked. If it is not yet present on the basket, a new card is created, otherwise, the **quantity** field on the existing card is increased. Additionally, the **total price** of the products in the basket is displayed on the bottom bar. The products can also be **removed** from the basket by clicking in the trash can on the right top of each card.

Moreover, the “**plus**” button is disabled when the basket reaches its **maximum quantity** of items (10 items), whereas the button to proceed with the **checkout is disabled** when the **basket is empty**.

The scanner, an outsource activity, reads the **QR Code**, which is **encrypted by the server**, and returns its content to the application. The app has the responsibility to decrypt it using the server’s public key present on its certificate and parse its content. The QR Code **contains the product’s name, price, UUID** and a special **tagId** to ensure the credibility of the QR Code. The product’s UUID allows the customer application to **request the server** for the URL where a **representative picture of the product** must be hosted, whereas the **tagId** ensures that the QRCode was generated by the server. The product is then saved on the database and its picture is fetched and saved in cache to avoid multiple requests.

This page, as others presented below, has an **Options Menu** on its App Bar which allows the user to navigate to the Profile activity. Clicking on the “Checkout” button, the user can navigate to the Checkout page, presented in the following section.

The layout of the activity is presented on Figure 6.

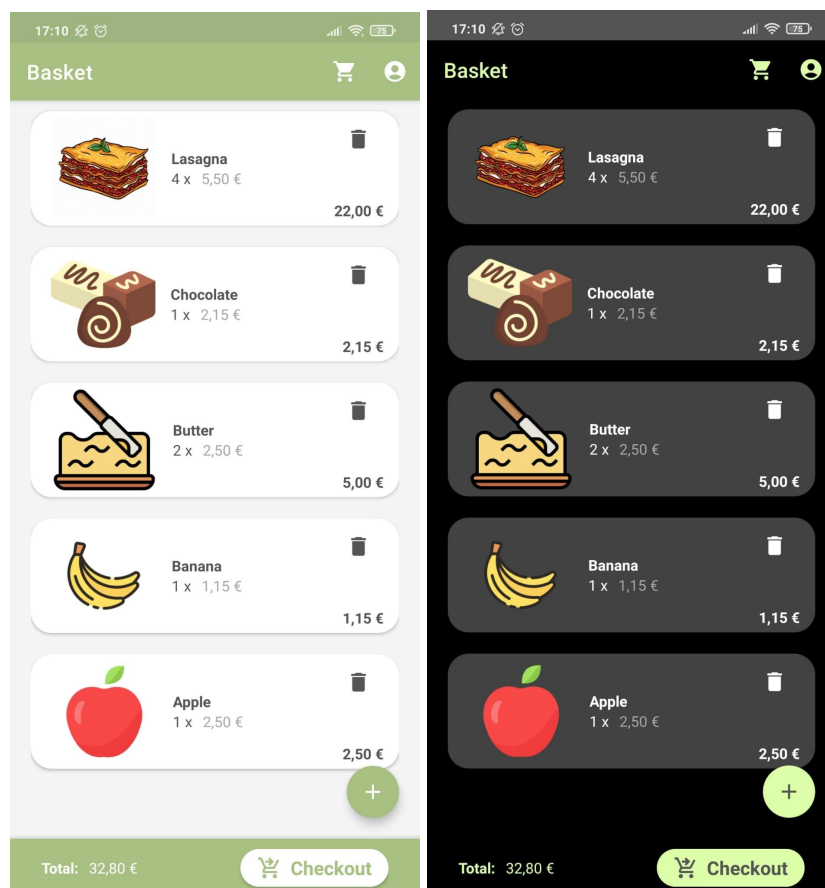


Figure 6. Emarket Customer Basket Activity

Checkout

The Checkout page (Figure 7) allows the user to **select**, or not, **one voucher** to use on that purchase and to decide if the **accumulated amount** from previous voucher uses is **to be discounted**.

Additionally, it displays the total amount of the purchase and, if the discount checkbox is checked, the amount to pay after the accumulated amount is discounted.

Clicking on the “Confirm” button the user is sent to the Payment activity corresponding to the default method of payment (which can be defined on Settings).

The user can also choose the payment method to be used on that purchase by long clicking the “Confirm” button. This will open a **Context Menu** that enables them to select between completing the transaction via **NFC** or **QR Code**.

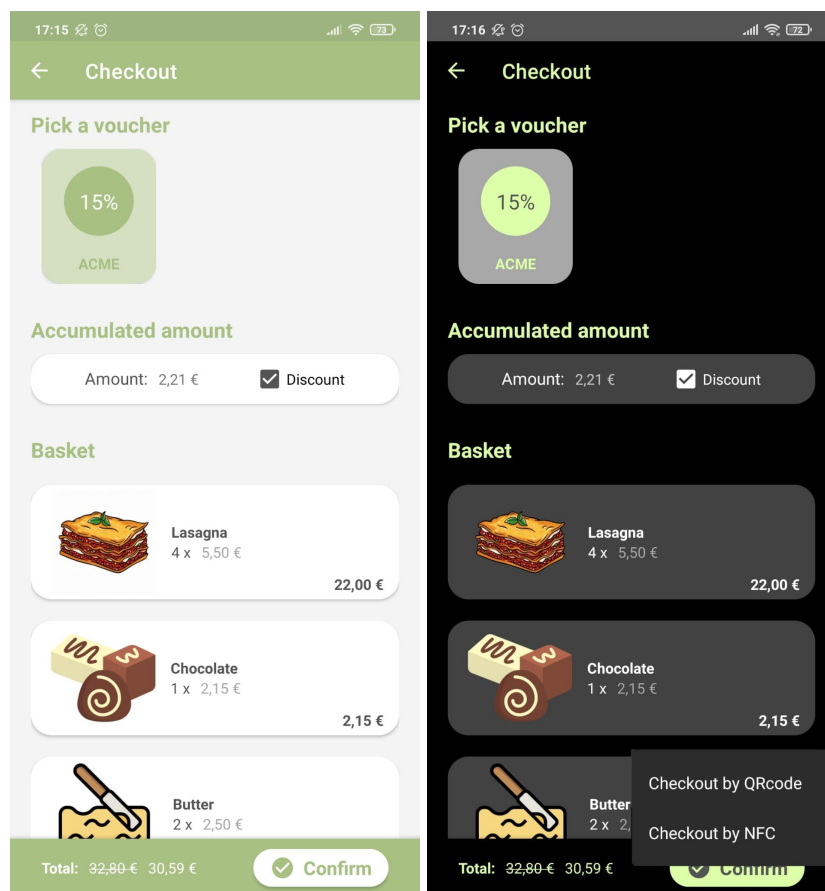


Figure 7. Emarket Customer Checkout Activity

Payment QR Code

In the Payment QRCode activity (Figure 8), a **QR Code is generated** that encodes a byte array with the **payment information** and the respective **signature**. More information on this representation can be found in [Checkout Information Representation](#). This QR Code can be scanned by the Emarket Terminal application, which transmits the data to the server and displays the outcome of the operation upon receiving a response.

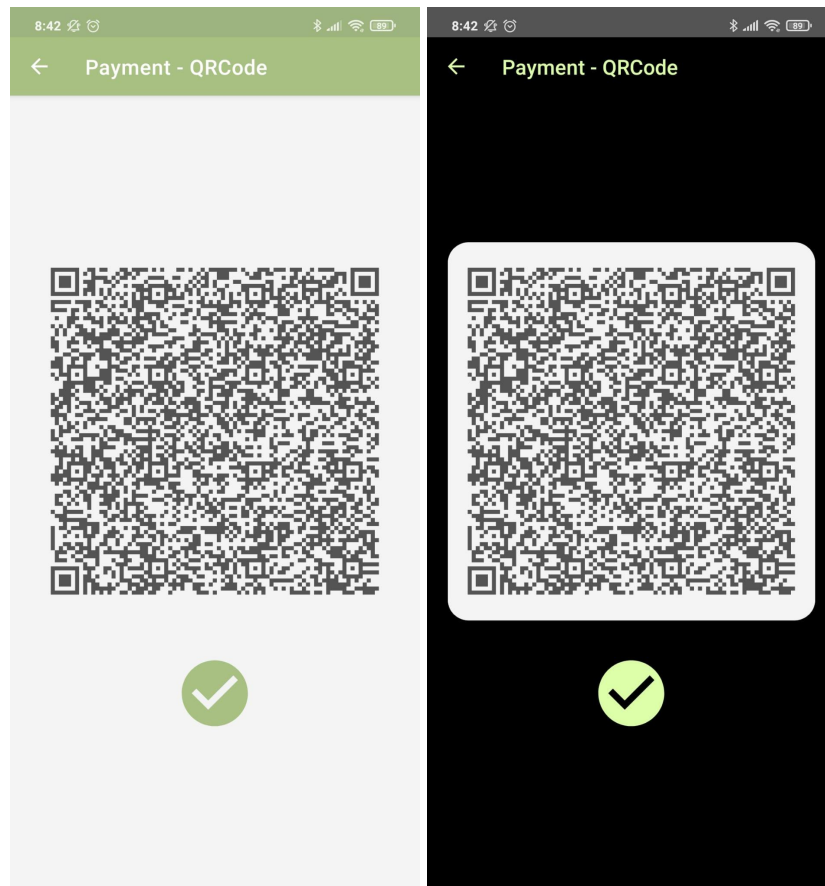


Figure 8. Emarket Customer Payment QRCode Activity

Payment NFC

The Payment NFC activity (Figure 9) handles **NFC communication** between the Emarket Customer and the Emarket Terminal applications. It emulates an NFC tag with the byte array described in [Checkout Information Representation](#). This tag is transmitted to the Terminal upon approximation of the two devices. This transmission triggers a **broadcast receiver** which displays the check button. This button redirects the user to the Basket activity, cleaning the current basket.

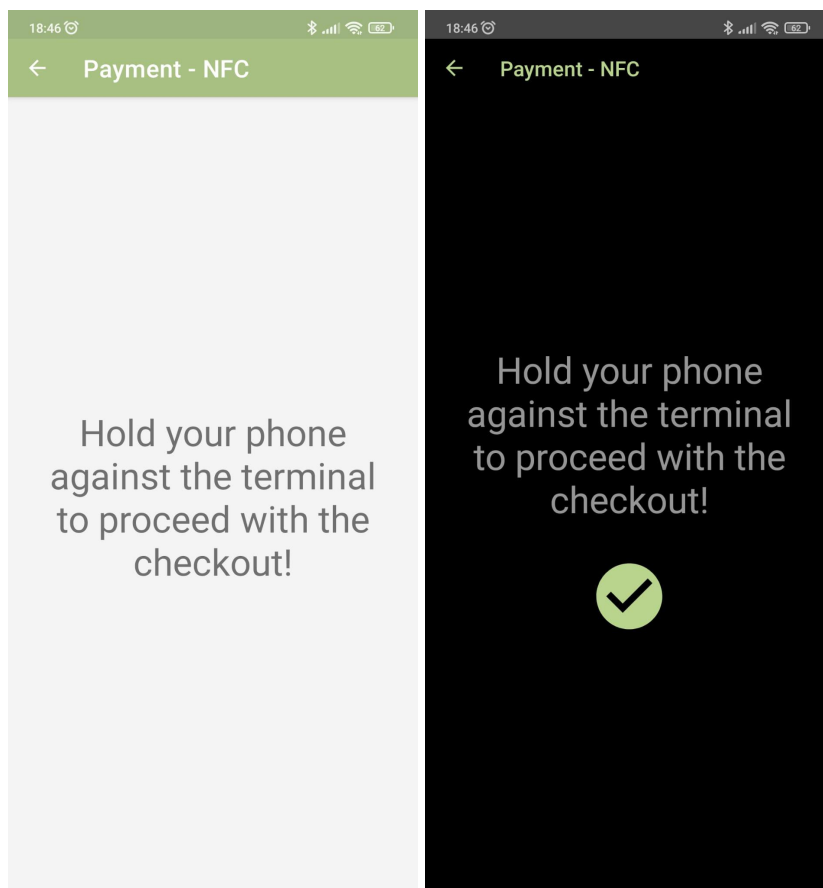


Figure 9. Emarket Customer Payment NFC Activity. left - light mode before tag transmission, right - dark mode after tag transmission

Profile

The profile activity contains valuable information on the user, and is divided in two **tabs**. The Transactions tab, where the user can access their **past transactions**, and the Profile tab, where other information on the user can be found.

Profile Tab

This fragment (Figure 10) presents to the user their name, nickname, card number, their total amount spent on the ACME supermarket and the amount they have to discount on future purchases.

Additionally, it presents to the user all their **unused vouchers**, and enables them to **edit** their name and card number, prompting a Dialog when the user clicks on the pencil icon (Figure 11).

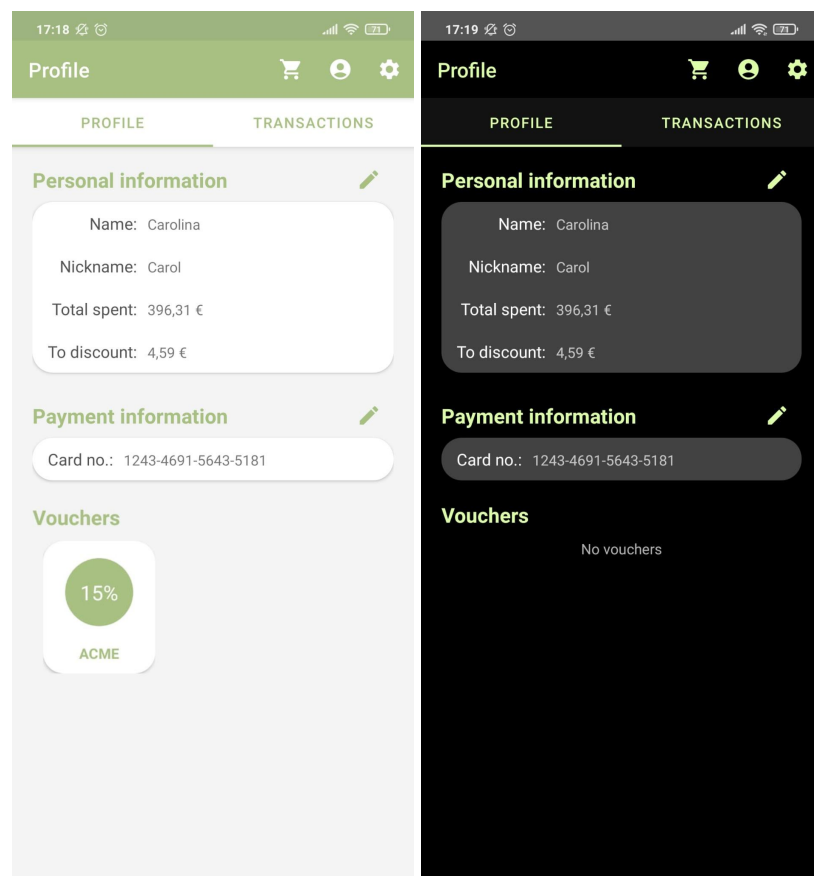


Figure 10. Emarket Customer Profile Activity - Profile Fragment

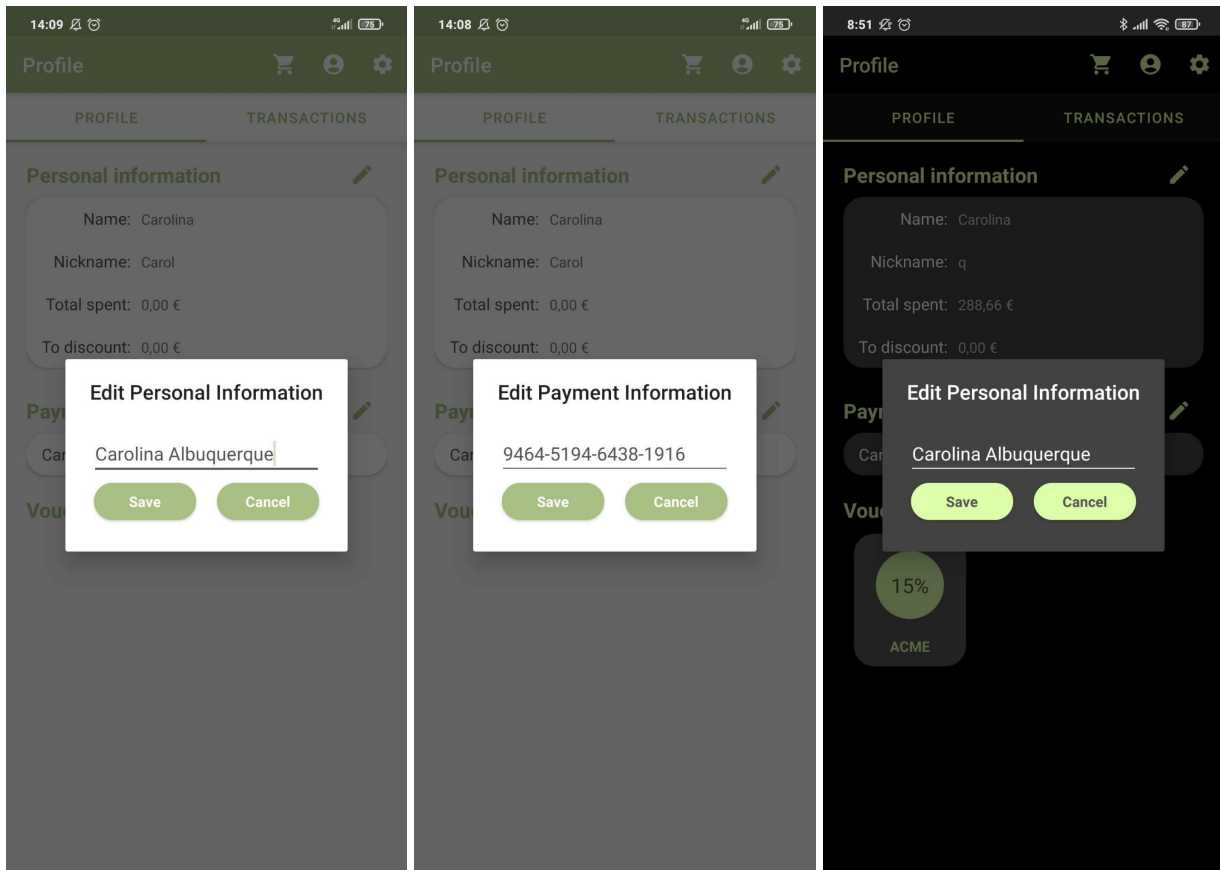


Figure 11. Edit user information dialogs

Transactions Tab

This fragment (Figure 12) displays a list of all the transactions that the user has made. Each transaction card contains the **purchase time** and the **total amount paid**. If the user has opted to use any accumulated amount towards a particular transaction, the total amount before and after the discount is displayed.

Clicking on a transaction card takes the user to a detailed view of the selected transaction. Additionally, this section allows users to **filter** their transactions by date. By clicking on the 'Filter' floating button, a bottom bar with controls to input the beginning and ending dates of the search is displayed. Users can enter just one date to apply a unilateral filter, or both dates to view transactions made within a specified range. To input a date, users can use a **date picker dialog** by clicking on the corresponding input field, as shown in Figure 13. When a filter is selected, the user can **remove the filter** by clicking on the same floating button which now corresponds to the no filter button. Whether there is a setted filter or not, the user can **hide the bottom action bar** by clicking on the down arrow button. This allows the user interface to be cleaner and more suitable for smaller screens.

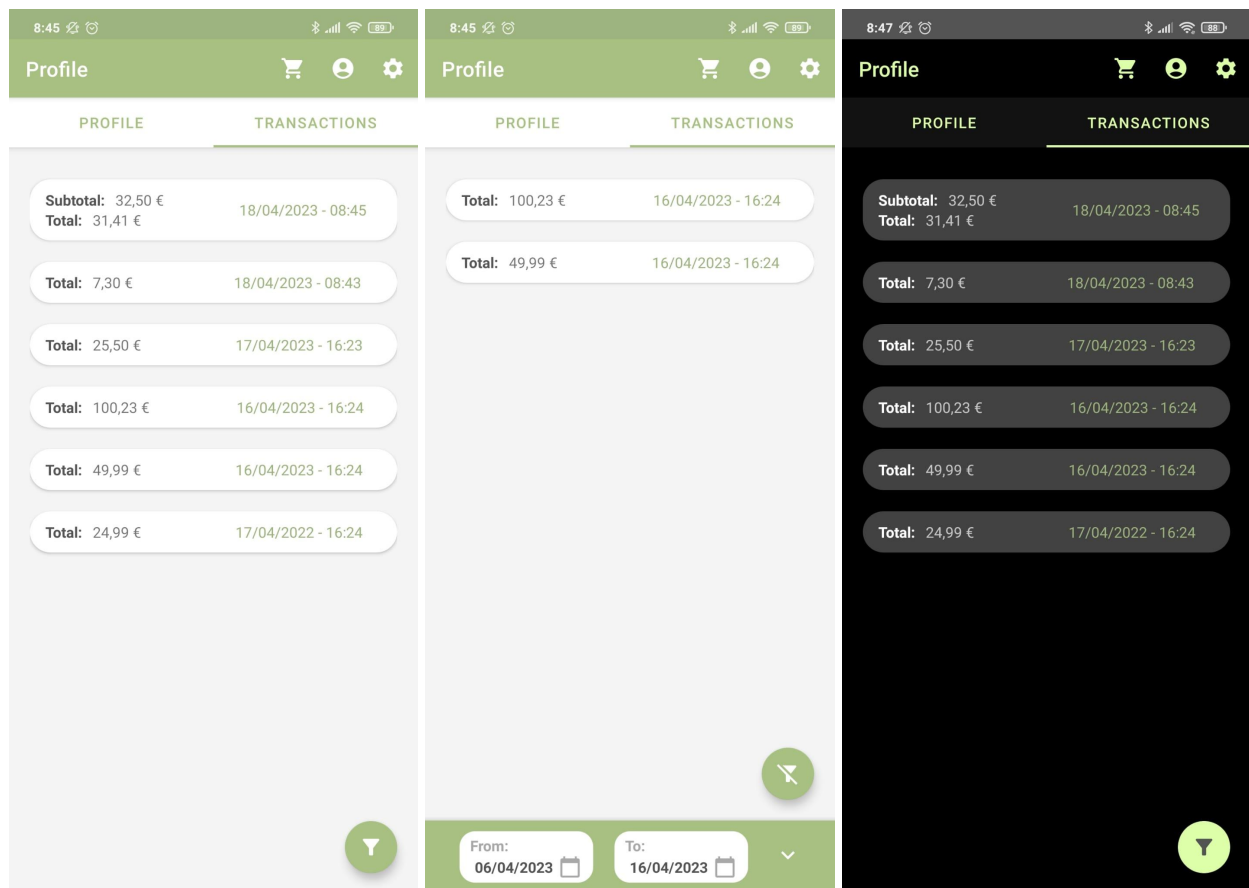


Figure 12. Emarket Customer Profile Activity - Transactions Fragment

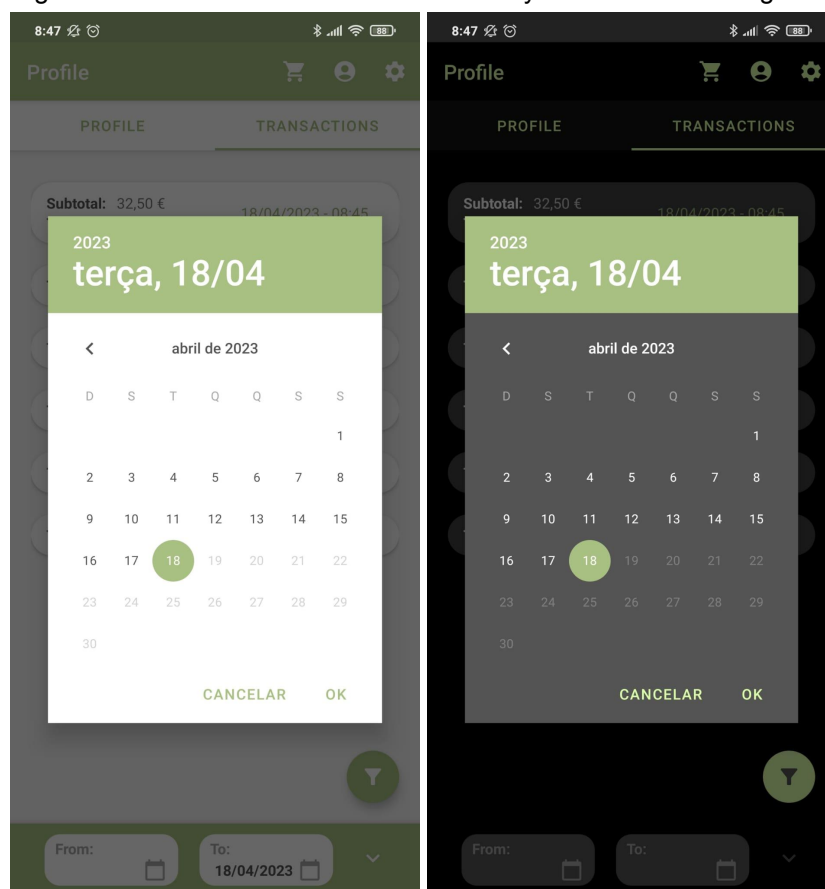


Figure 13. Emarket Customer Profile Activity - Day Picker Dialog

Transaction details

The Transaction Details activity (Figure 14) enables users to view **specific information** about one of their past **transactions**. This page displays the **products** associated with the purchase (using a similar layout to the "Basket" page), the **amount paid** by the customer, and, if the user has discounted any value on the purchase, the **total cost** of the purchased products. In addition, if a **voucher** was used for the purchase, this page displays that information as well.

All of the information on this page is retrieved from the local database, which is regularly updated based on the data available on the server.

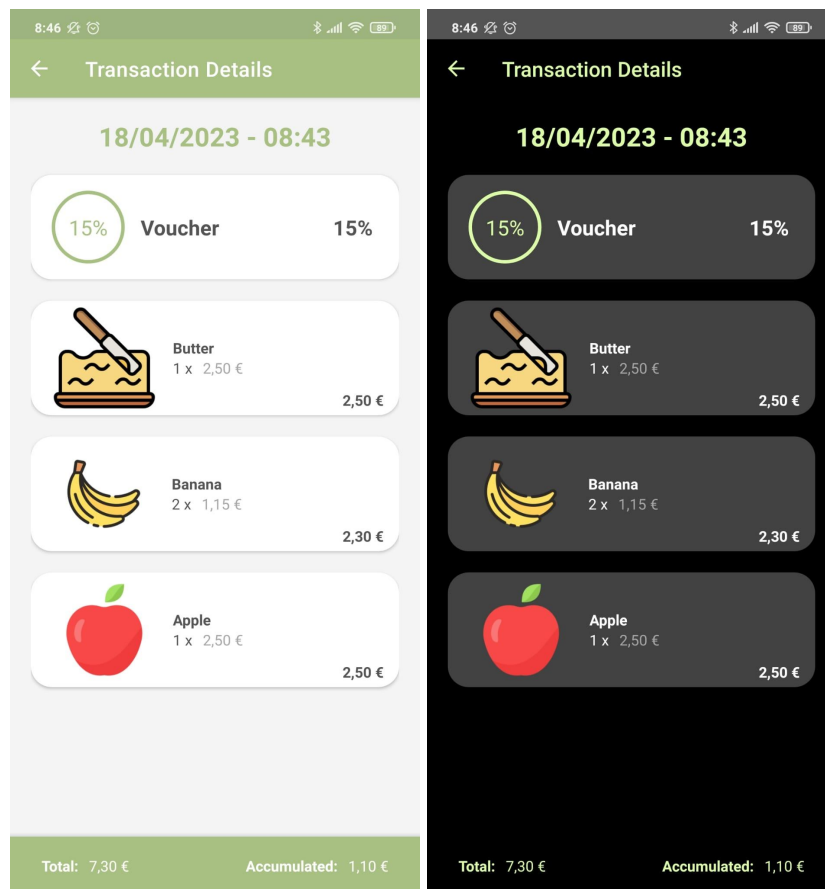


Figure 14. Emarket Customer Transaction Details Activity

Settings

The settings page (Figure 15) enables the user to turn the app **notifications on or off** and to select the **default checkout method**. This information is saved on the *Shared Preferences* as two booleans. The NFC radio button is disabled when the device doesn't support NFC communication or has it disabled.

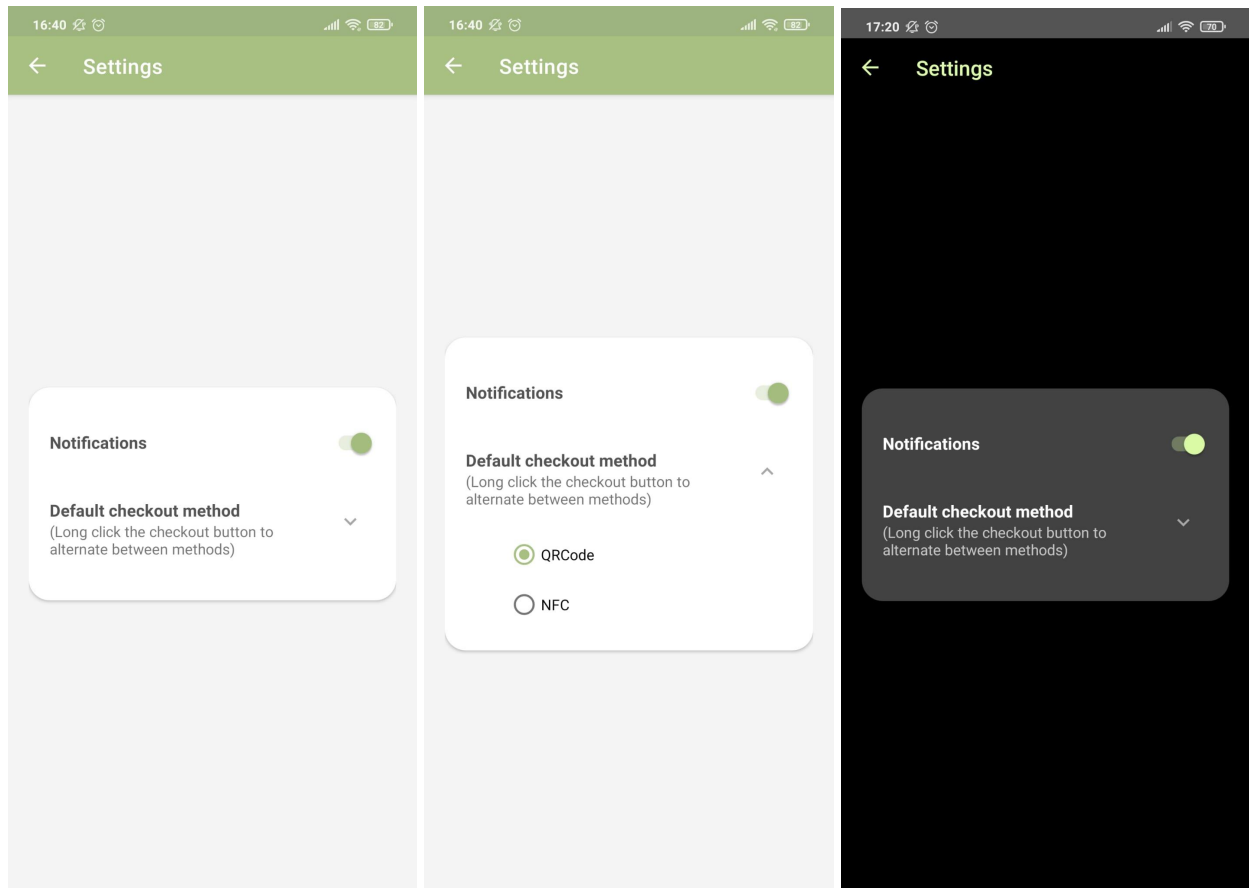


Figure 15. Emarket Customer Settings Activity

Notifications

Whenever a **new transaction** is fetched from the server, the user **receives a notification** with the amount paid on that purchase, as it can be seen in Figure 16. By clicking on the notification the user is redirected to the corresponding Transaction Details activity.

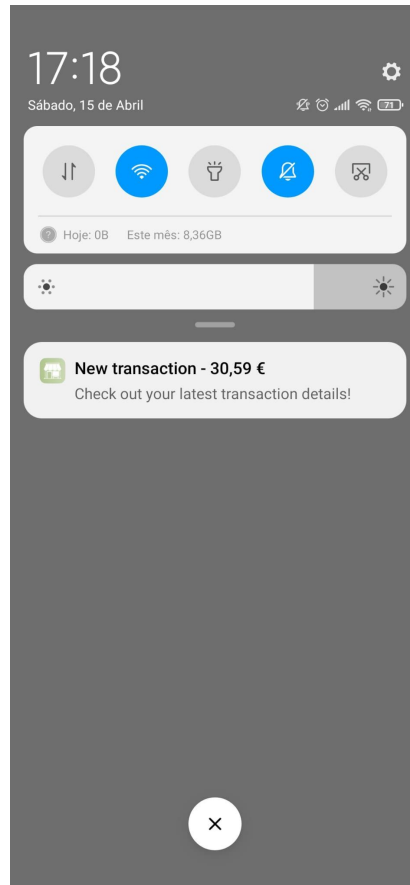


Figure 16. Emarket Customer Notification

Emarket Terminal

This application runs on the Terminal Machines of the supermarket, and enables the user to autonomously checkout their own products, opening the supermarket doors on successful completion of the purchase.

Activities

Main

This activity (Figure 17) enables customers to complete their purchase using either **NFC Communication** or **QR Code scanning**, making the supermarket accessible to those who do not own NFC-supported devices.

If the user selects the NFC checkout option on their ACME app, the checkout procedure is initiated by simply approximating their device to the supermarket terminal running the Emarket Terminal app. This transmits the NFC tag generated by the customer's app to the Terminal, where it is processed and sent via HTTP to the server.

If the user selects the QR Code checkout option, they have to click on the “Read QR Code” button, in order to open a QR Code reader activity. This QR Code is processed and the checkout information is sent to the server via HTTP.

On either way, upon transmission of the checkout information to the server and consequent response, the terminal is redirected to the Result activity.

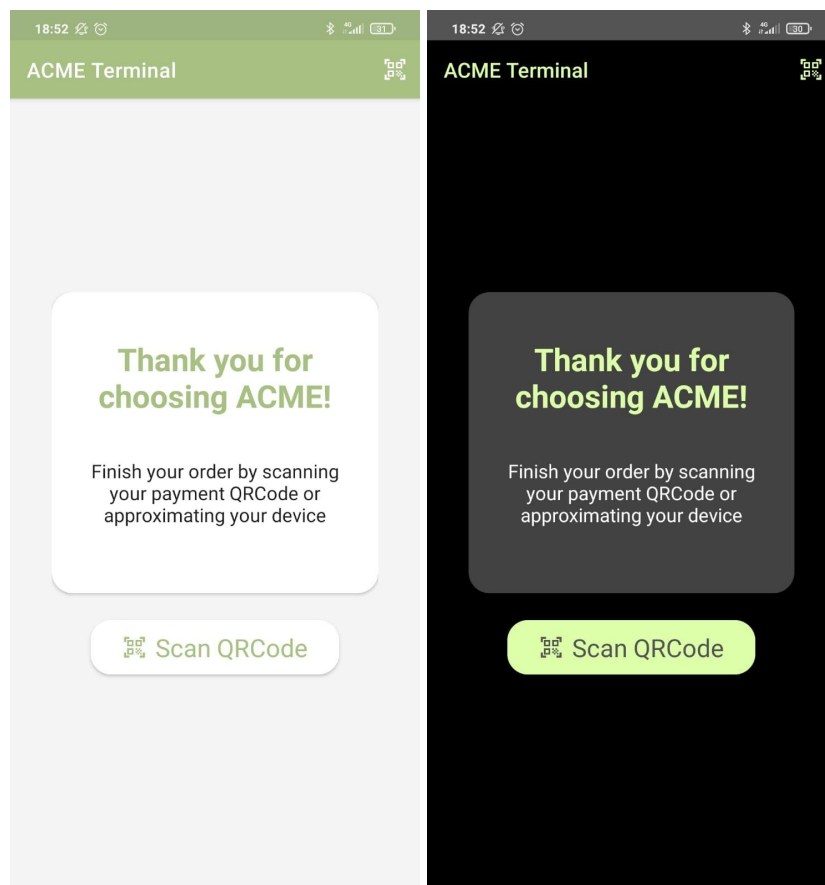


Figure 17. Emarket Terminal Main Activity

Result

This activity (Figure 18) informs the user on the **success of the checkout** operation. If the checkout was completed successfully, the terminal displays a **success message** and the total **amount paid** by the user in a very visible way. Additionally, a **“Finish” button** is available, which redirects the application to the Main page.

Otherwise, an **error message** is displayed and a **“Try again” button** is made available, which also redirects the user to the Main page.

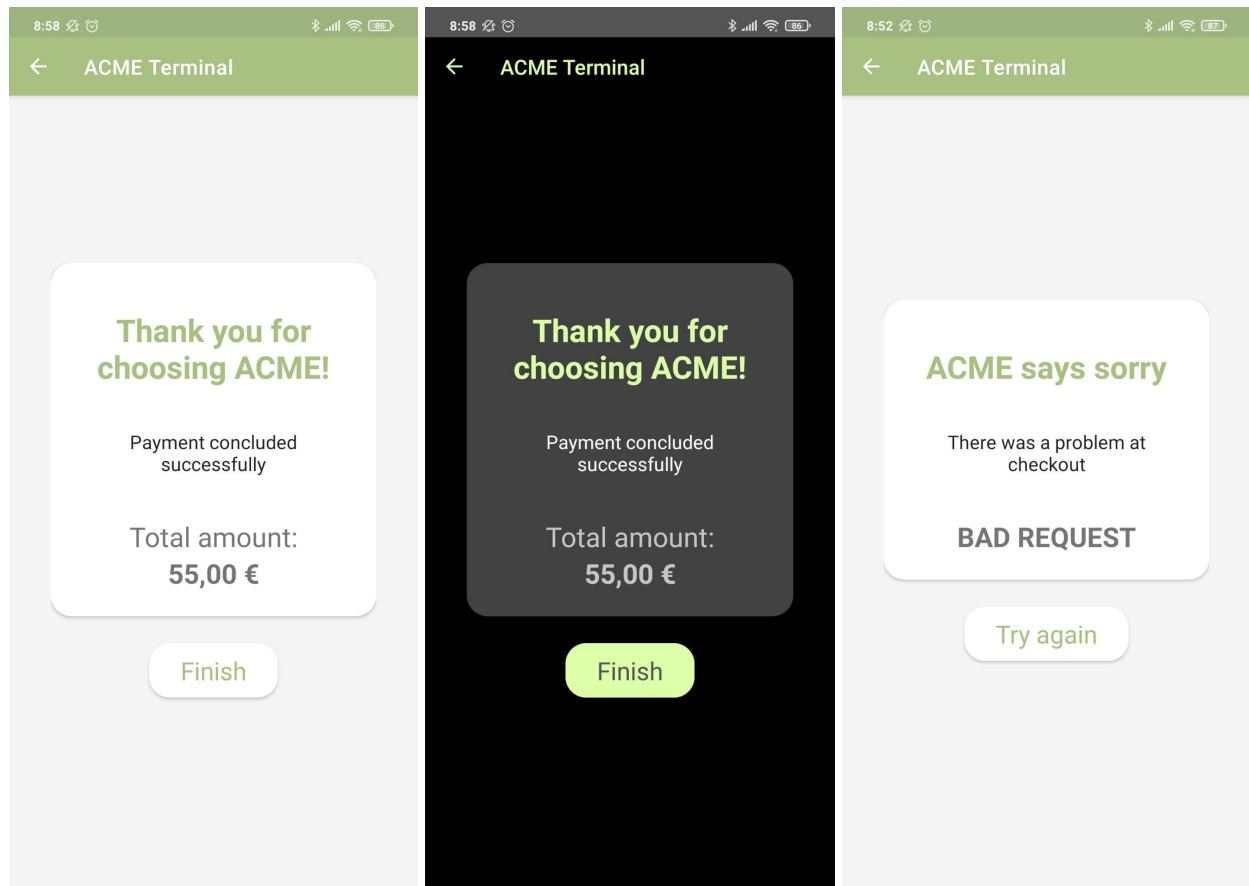


Figure 18. Emarket Terminal Result Activity upon: left - successful checkout, center - successful checkout (dark mode), right - unsuccessful checkout

Representations

Checkout Information Representation

The communication between the Emarket Customer and Emarket Terminal must be as concise as possible. Therefore, it is represented as a **Byte Array** on both checkout methods (QRCode and NFC). This Byte Array is composed of a **signature** (the first 64 bytes) which signs the payment information, and the **payment information** itself (the remaining bytes of the array).

The payment information contains the **user UUID**, a byte indicating whether the **discount** was applied or not, a byte indicating if the user has used a **voucher** and, if so, the **voucher's UUID**. Additionally, the payment information includes a byte representing the **number of different products** to be purchased and 20 bytes for each product. Of these 20 bytes, 16 are for the **product UUID**, 2 for the integer part of the **price**, 1 for the decimal part of the price and one for the **quantity**.

Overall, the maximum size of the checkout information is 299 bytes, which is suitable for generating reasonably sized QRCodes. Figure 19 has an illustrative representation of the described byte array.

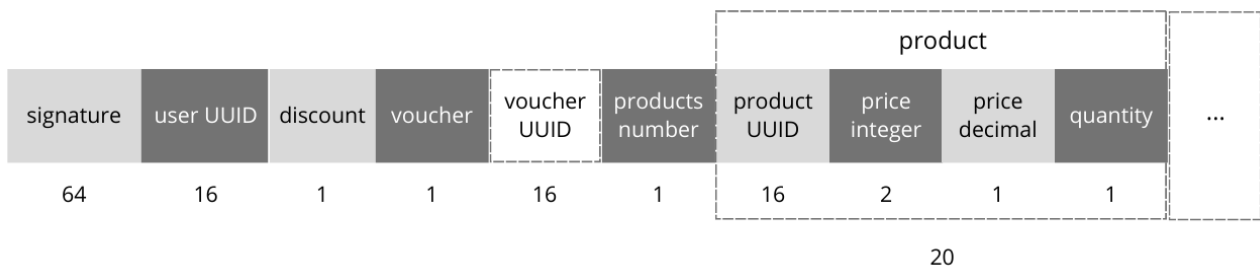


Figure 19. Checkout information byte array representation. The numbers below each fragment correspond to its size in bytes.

Testing

Cryptography is a critical component of modern digital security, as it enables the protection of sensitive information from unauthorized access. In order to ensure that an application is secure, it is essential to thoroughly test its cryptographic implementation.

To test the security of the application, we tested every external request and tried to surpass them, both to crash the server or the app, to replicate any action, or simply to try to impersonate another user.

Get User Information

When requesting user information, the app must send to the server the *user UUID* and a *signature* of it, using the user's private key. This signature is verified in the server, as he doesn't accept signatures using another private key instead of the user that is making the request. On the other hand, it also displays the respective error in case the user didn't provide the necessary information (the user UUID and the respective signature).

Update User Information

When updating the user information, the app must send to the server the new user information and the respective signature, using the user's private key. In the new user information, it should be the user UUID and the new information of the user, which is verified by the server. Since all of this must be signed with the user's private key, the server also verifies the signature and responds according to the result.

QR Code Reading

As explained in the [Basket activity section](#), the QR Code is encrypted by the server, and it contains a special tag ID to verify that the server created it. Our application was tested using QR codes that were encrypted using private keys from other servers and wrong tag IDs, displaying to the user that the QR Code is not authentic.

Payment QR Code or NFC

In the payment procedure, the Emarket Customer app sends the payment information and the respective signature to the Emarket Terminal app. To test this, we simulated a checkout procedure with wrong signatures (not signing the correct content, or not using the user's private key) or checkout information with invalid fields. In every case, the respective error is displayed to the user. From the payment information, it is verified if the voucher is from the respective user and if it wasn't used yet, and if the product's UUID and price are according to the ones in the Emarket Server database.

How to run

Information on how to run each of the 3 applications presented on this report can be found on the respective README files, and on the following section.

Emarket Server

To run the Emarket Server, there has to be a MongoDB instance running on the device. Further instructions on how to install and run MongoDB can be found [here](#).

The required dependencies should also be installed. To do so, use the following command:

```
pip install -r requirements.txt
```

The MongoDB URI can be configured by setting the MONGO_URI environment variable in the .env file.

```
MONGO_URI=mongodb://localhost:27017/ # default value
```

Before running the application, the database should be populated with the products corresponding to QR codes already available in the system. To do so, open a MongoDB client like MongoDB Compass and import the products.json file in the Emarket Server folder.

Run the following command to start the flask application exposing the REST API to the private network on port 5000 (by default):

```
python3 app.py
```

Register the ip address where the server is running so it can be used to configure the other apps of the system.

Emarket Customer

After the server is running, the variable SERVER_URL in the Constants.kt file must be replaced for the correct url.

This url is composed by the ip address and the port of the server. For example, if the server is running on localhost with port 8080, the url is `http://localhost:8080`.

After this configuration, the application can be run on an emulator or on a real device.

Emarket Terminal

Similarly to the Emarket Customer, after the server is running, the variable SERVER_URL in the Constants.kt file must be replaced for the correct url.

This url is composed by the ip address and the port of the server. For example, if the server is running on localhost with port 8080, the url is `http://localhost:8080`.

After this configuration, the application can be run on an emulator or on a real device.

Conclusion

The developed system meets all the specified requirements and includes some additional features such as the ability to filter previous transactions, different checkout methods, and initiate the QRCode reader by shaking the device, among others.

Even though the cryptographic requirements were completely satisfied, the cryptographic techniques employed were primarily for academic purposes to enhance our knowledge in the field. Some of the operations performed may not be completely secure as per the required standards.

Overall, the successful development of this system provided us with valuable insights into useful tools and techniques in the domain of mobile app development.

Appendix

Landscape layouts

Some of the landscape layouts presented are extended print screens since the contents of the page are scrollable and the images on this report are not.

Emarket Customer

Splash Screen

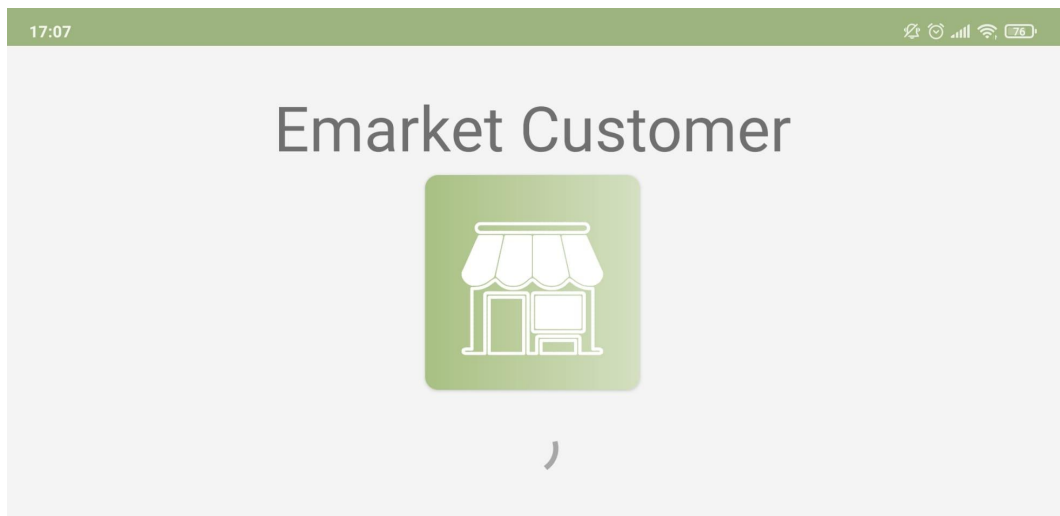
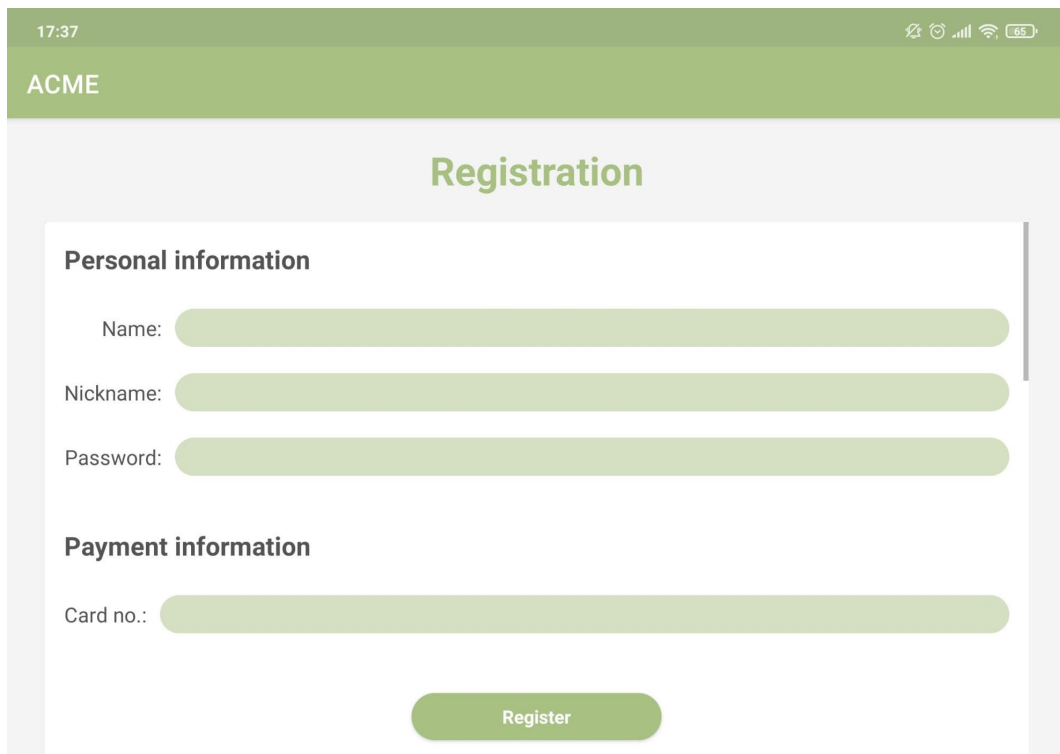


Figure A1. Emarket Customer Splash Screen - landscape

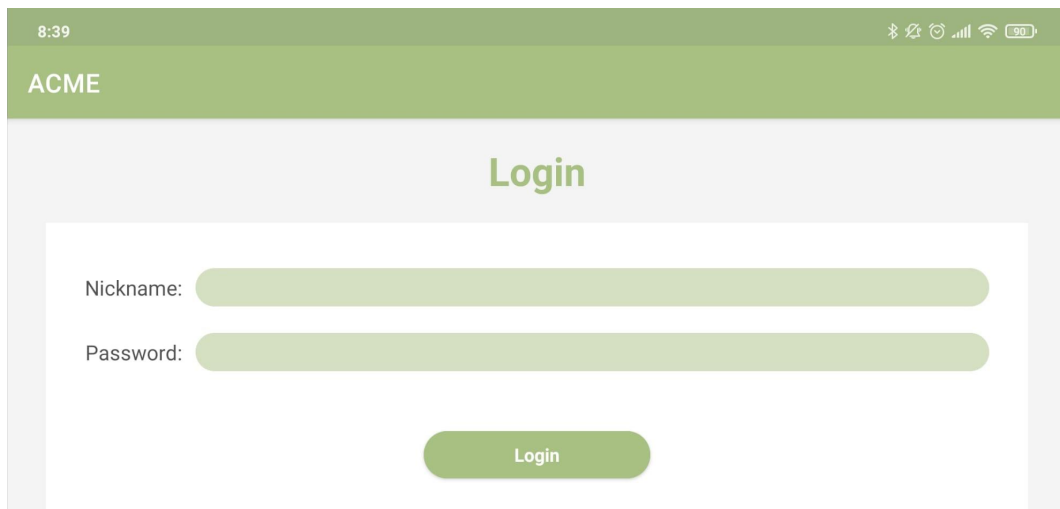
Registration



The image shows a mobile application interface for the 'ACME' brand. At the top, a green header bar contains the time '17:37' on the left and various status icons (signal, Wi-Fi, battery at 65%) on the right. Below the header, the word 'ACME' is displayed in white. The main content area has a light gray background with the title 'Registration' in a large, bold, green font. Below this title is a white rectangular form. The form is divided into two sections: 'Personal information' and 'Payment information'. Under 'Personal information', there are three input fields labeled 'Name:', 'Nickname:', and 'Password:', each followed by a green rounded rectangle representing the input area. Under 'Payment information', there is one input field labeled 'Card no.:' followed by a green rounded rectangle. At the bottom of the form, centered, is a green rounded button with the text 'Register' in white.

Figure A2. Emarket Customer Registration Activity - landscape

Login



The image shows a mobile application interface for the 'ACME' brand. At the top, a green header bar contains the time '8:39' on the left and various status icons (Bluetooth, signal, Wi-Fi, battery at 90%) on the right. Below the header, the word 'ACME' is displayed in white. The main content area has a light gray background with the title 'Login' in a large, bold, green font. Below this title is a white rectangular form. The form contains two input fields: 'Nickname:' followed by a green rounded rectangle, and 'Password:' followed by a green rounded rectangle. At the bottom of the form, centered, is a green rounded button with the text 'Login' in white.

Figure A3. Emarket Customer Login Activity - landscape

Basket

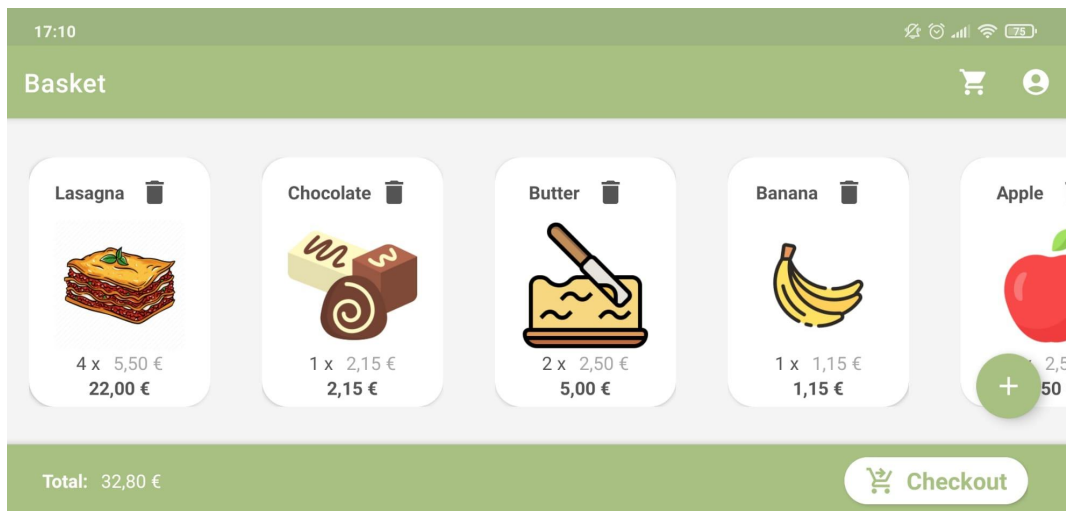


Figure A4. Emarket Customer Basket Activity - landscape

Checkout

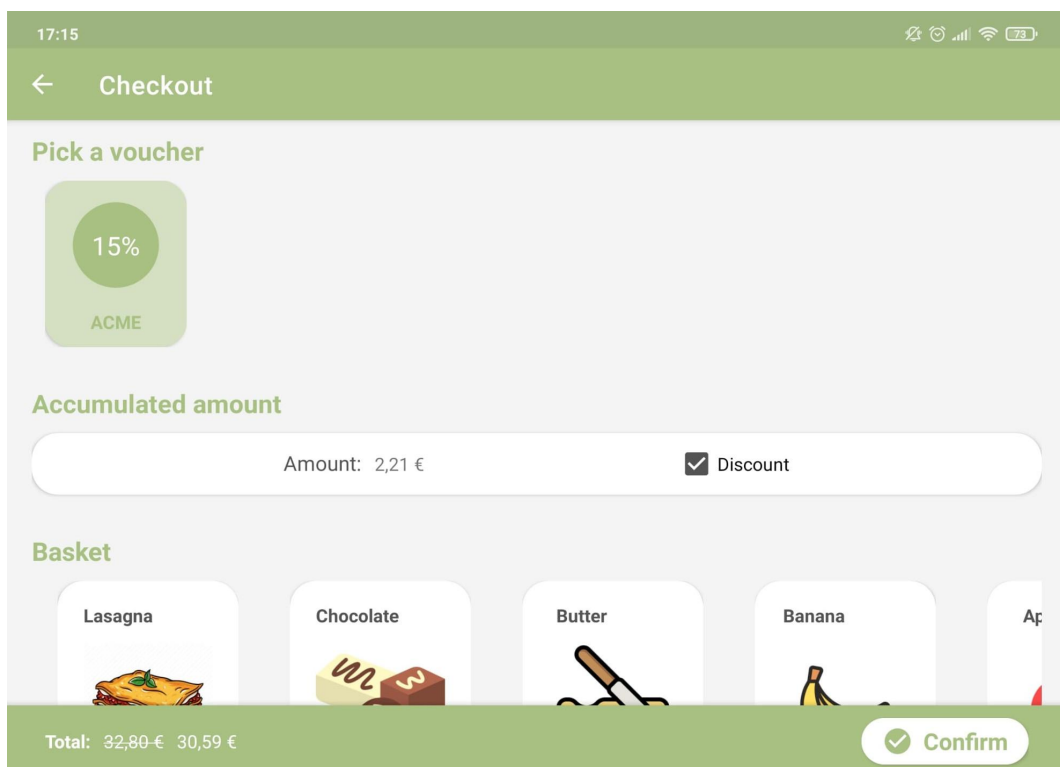


Figure A5. Emarket Customer Checkout Activity - landscape

Payment - QRCode

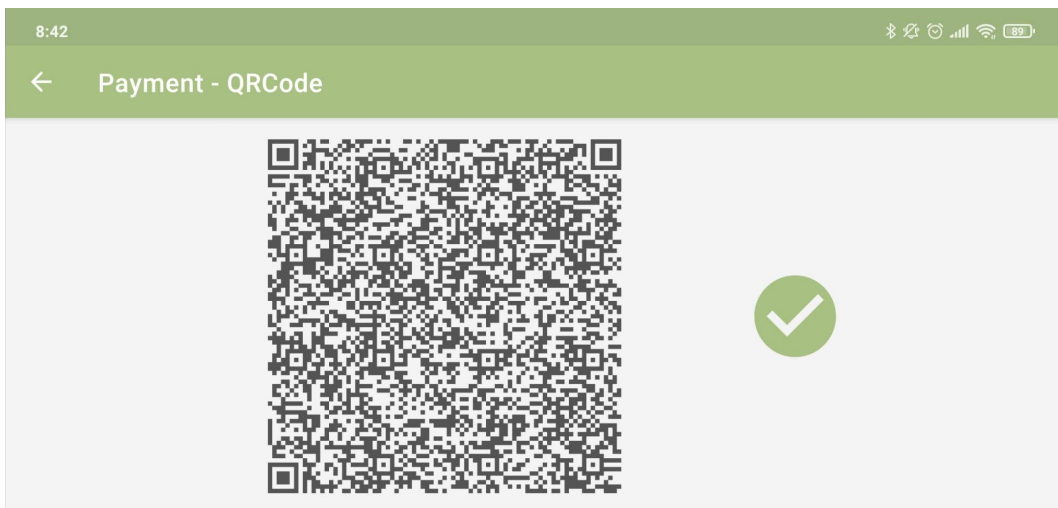


Figure A6. Emarket Customer Payment QRCode Activity - landscape

Payment - NFC

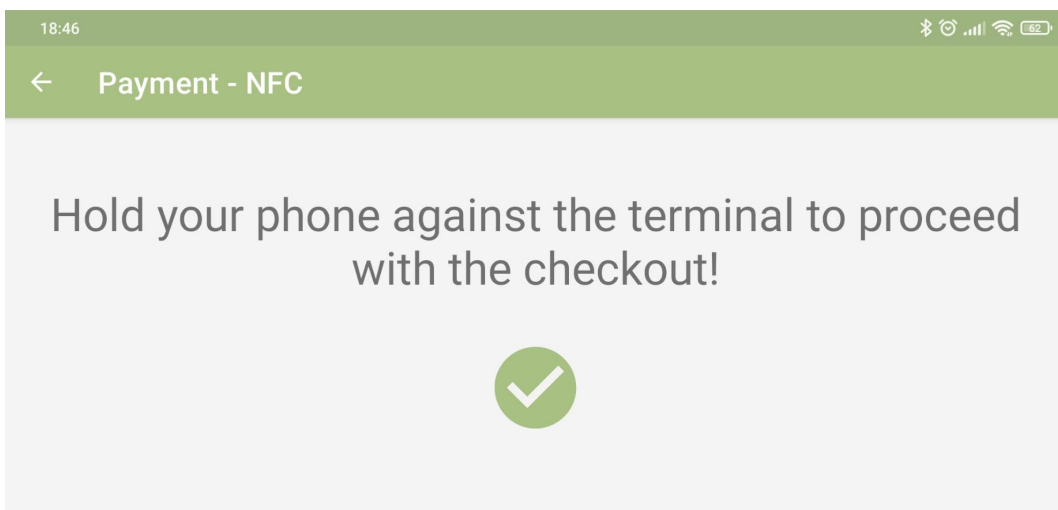


Figure A7. Emarket Customer Payment NFC Activity - landscape

Profile

Profile Tab

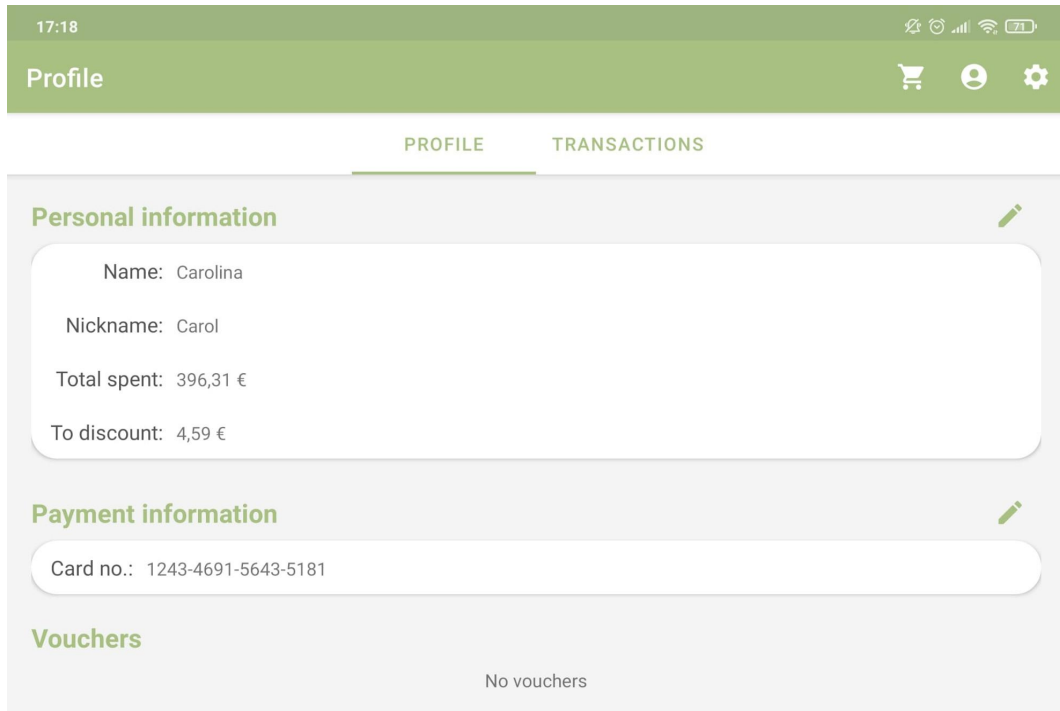


Figure A8. Emarket Customer Profile Activity - Profile Fragment landscape

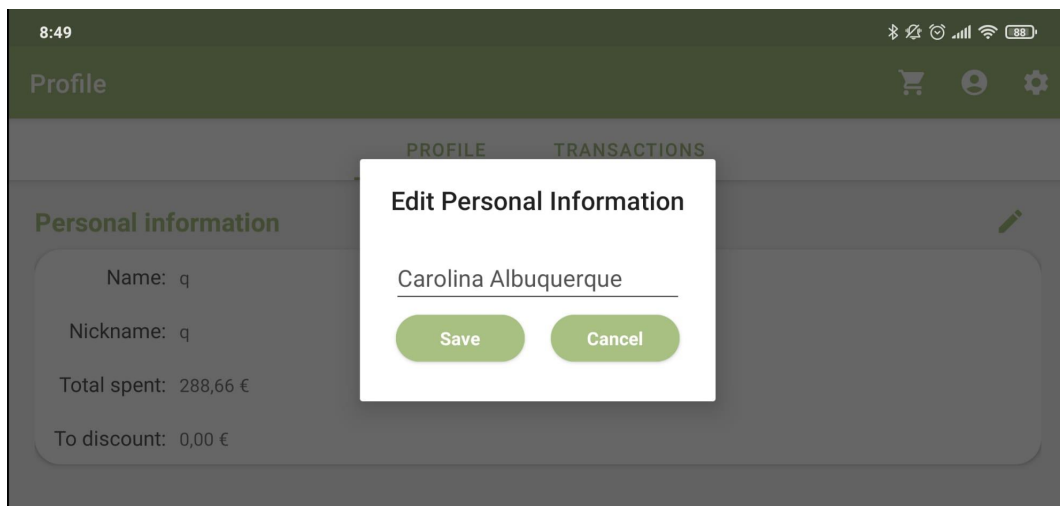


Figure A9. Edit user information dialog - landscape

Transactions Tab

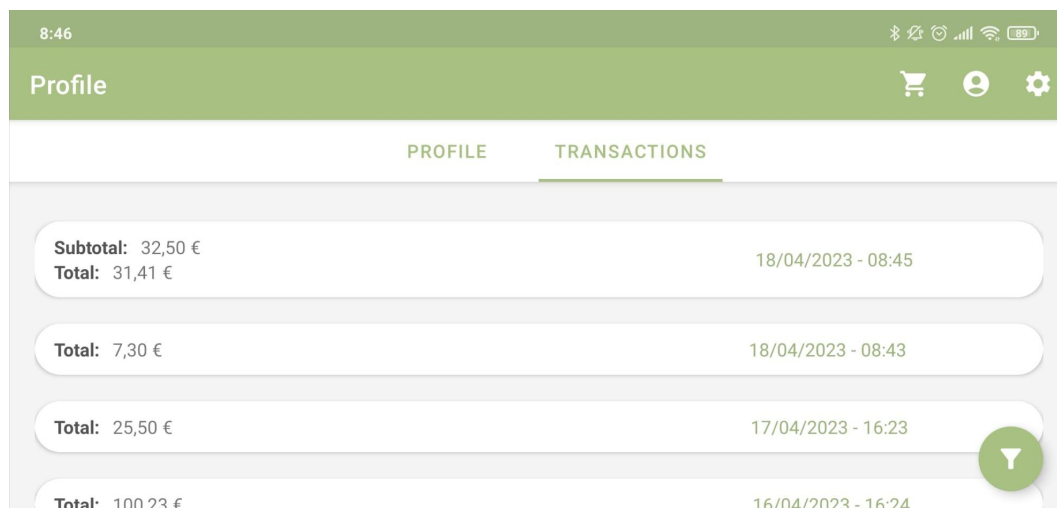


Figure A10. Emarket Customer Profile Activity - Transactions Fragment landscape

Transaction details

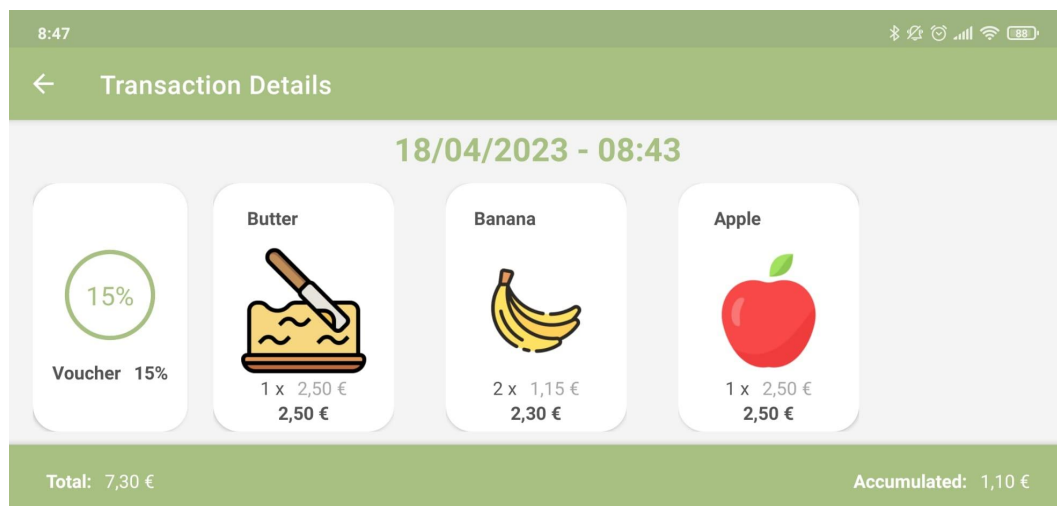


Figure A11. Emarket Customer Transaction Details Activity - landscape

Settings

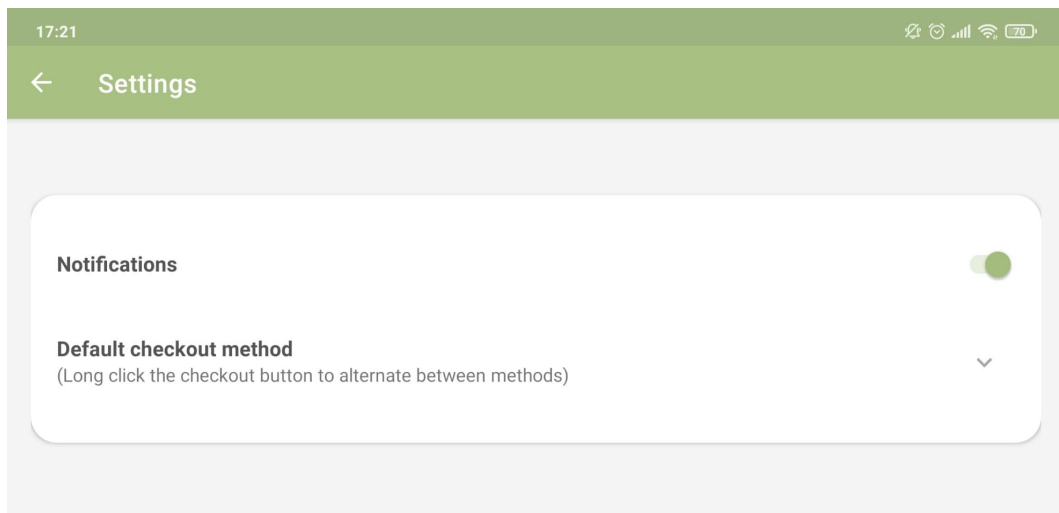


Figure A12. Emarket Customer Settings Activity - landscape

Emarket Terminal

Main

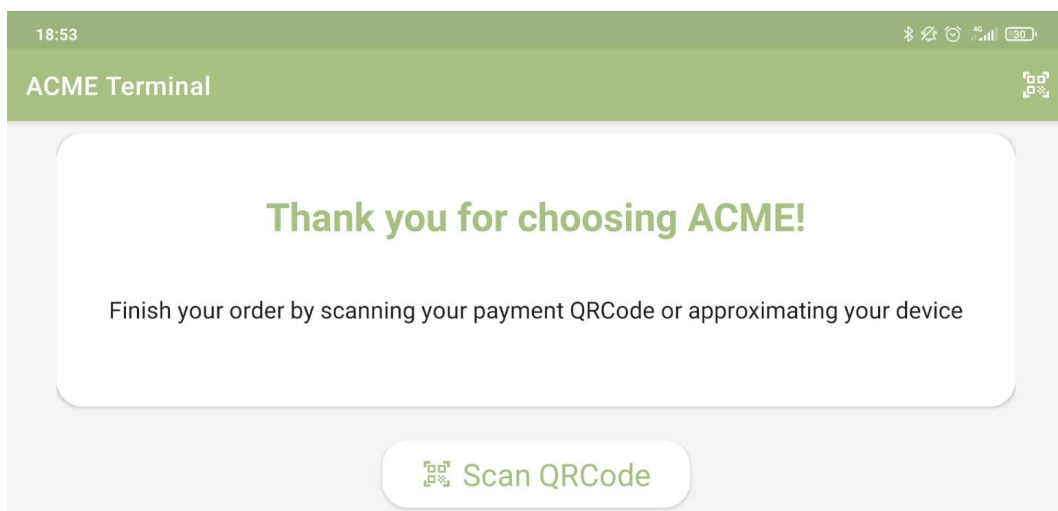


Figure A13. Emarket Terminal Main Activity - landscape

Result

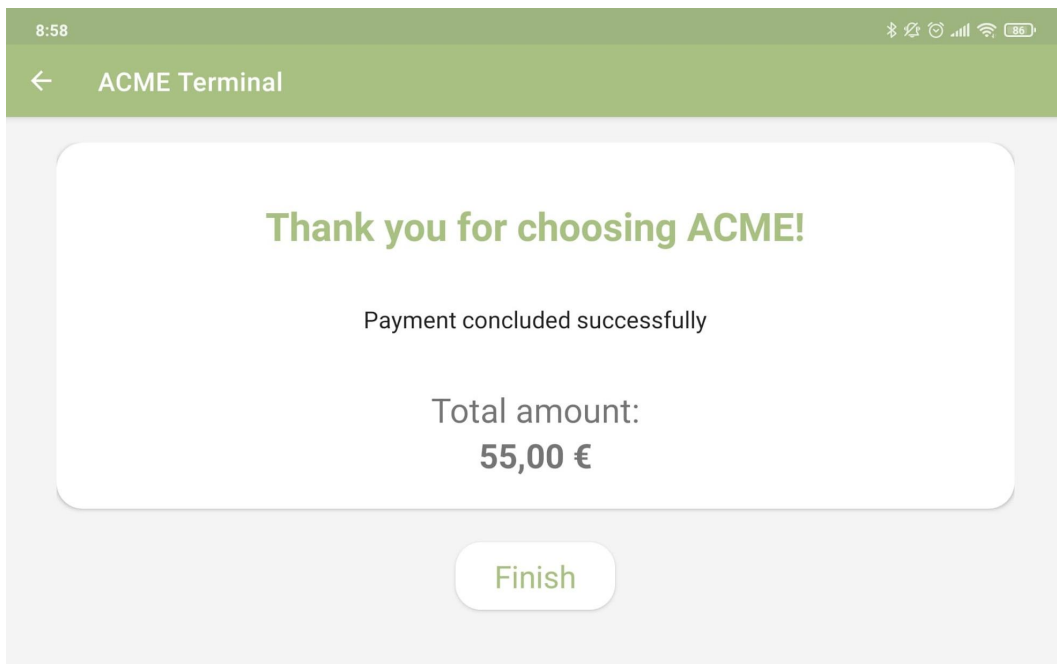


Figure A14. Emarket Terminal Result Activity upon successful checkout - landscape

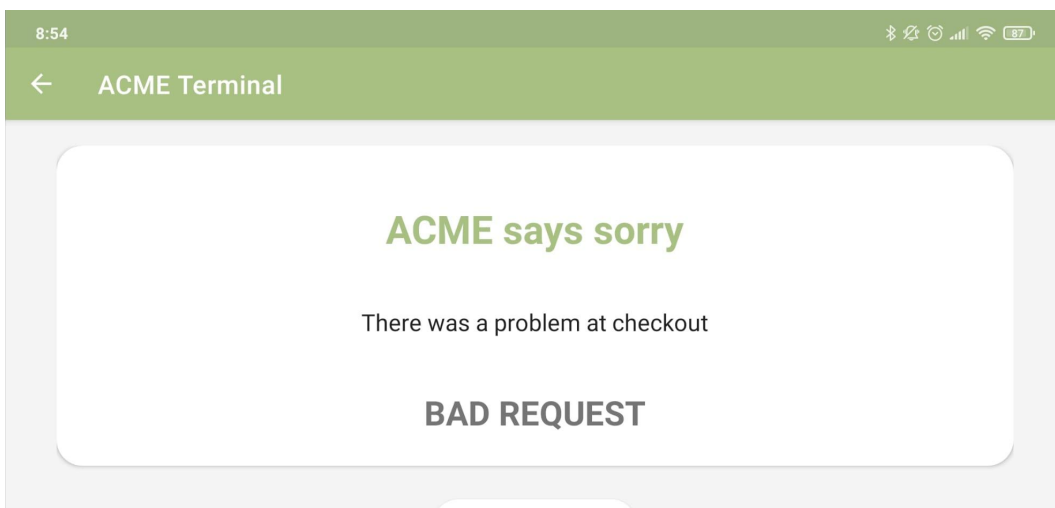


Figure A15. Emarket Terminal Result Activity upon unsuccessful checkout - landscape