FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Mestrado em Engenharia Informática e de Computação

Computação Móvel

Practical Assignment #1
Ticket/Cafeteria Ordering System

Group 4

Diogo André Pereira Babo (up202004950)
Ana Sofia Oliveira Teixeira (up201806629)

24/04/2024

Architecture

- **High-Level Overview**

The architecture of the application follows a client-server model, with the client-side developed using Kotlin for Android devices. The server-side logic and communication are implemented using Node.js, while SQL databases handle data management tasks. This architecture allows for efficient handling of user interactions, data storage, and communication between the client and server components.

- **Technology Stack**

The application was built using Kotlin for the Android client-side, SQL for data management, and Node.js for the server-side logic and communication

- **Client-Side Architecture**

The client-side architecture, developed with Kotlin, integrates a Model-View pattern and makes extensive use of XML resources for defining layouts, styles, and other visual elements. This approach streamlines development by promoting code organization, consistency, and efficient management of the Android application's user interface.

- **Controller**

The Kotlin controller module acts as the intermediary between the app and server, handling tasks like user registration, ticket purchases, and data retrieval. It utilizes coroutines for efficiency, HTTP requests for server communication, and cryptographic operations for data security. By employing SharedPreferences for local data storage and LiveData for UI updates, it ensures smooth functionality and enhances user experience.

- **Server-Side Architecture**

The server-side architecture utilizes RESTful APIs and controllers to handle business logic, data processing, and client communication. Implemented in Node.js with Express.js and SQLite3, it efficiently manages functionalities like customer registration, ticket purchases, cafeteria orders, and transaction consultations. These components ensure smooth system operation while interacting with the database and processing client requests.

- **Data Storage**

The system utilizes SQLite, a relational database, to store user information, ticket details, and cafeteria orders, among others.

- **Security Considerations**

The architecture emphasizes security with user authentication, data encryption, and HTTPS for secure communication. RSA key pairs are employed for user identification and transaction validation, ensuring data integrity and authenticity.

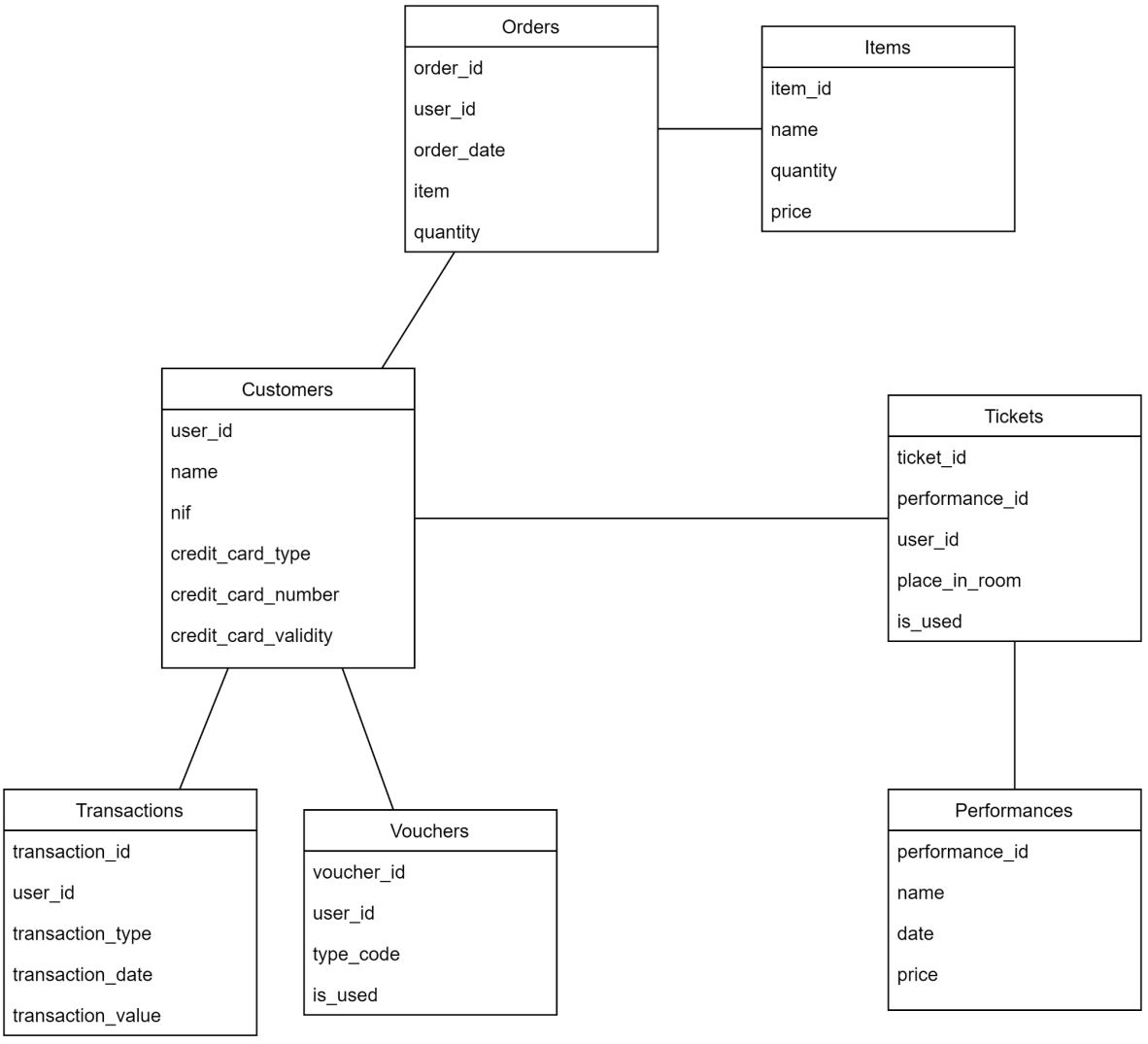
- **Communication Protocols**

The system predominantly relies on RESTful APIs for communication between client and server components. However, QR code transmission serves as an alternative communication channel for specific functionalities, facilitating interactions such as ticket validation and data transmission between devices that lack direct network connectivity.

Data Schemas

The server uses a database in sqlite with the following tables:

Customers (user_id, name, nif, credit_card_type, credit_card_number, credit_card_validity);
Performances (performance_id, name, date, price);
Tickets (ticket_id, performance_id, user_id, place_in_room, is_used);
Vouchers (voucher_id, user_id, type_code, is_used);
Transactions (transaction_id, user_id, transaction_type, transaction_date, transaction_value);
Items (item_id, name, quantity, price);
Orders (order_id, user_id, order_date, item, quantity);



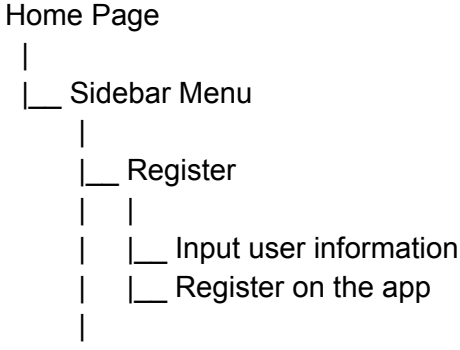
Included Features

- **Registration**
  - Allow users to register with their name, NIF, and credit card information.

    - Generate a 512-bit RSA key pair and transmit the public key to the server.
    - Generate and store a unique user ID and private key locally in a protected keystore.
- **Consult and Buy Tickets**
    - Display information about upcoming performances with dates, and prices.
    - Allow users to buy tickets for a performance, specifying the date and number of tickets.
    - Authenticate the user locally, preferably with biometric authentication.
    - Perform transactions securely using the user's private key and credit card information.
    - Emit tickets with unique IDs and cafeteria vouchers for each purchased ticket.
- **Present Tickets**
    - Allow users to transmit up to 4 tickets to a validation terminal.
    - Validate the tickets locally and send information to the server.
    - Display the validation status prominently on the validation terminal.
- **Validate Tickets**
    - Send ticket IDs, user ID, and user signature to the server for validation.
    - Verify the existence and validity of tickets on the server.
- **Make Cafeteria Order**
    - Enable users to compose a cafeteria order from a local menu.
    - Transmit order information, including user ID, ordered products, and vouchers, to the cafeteria terminal.
    - Validate the order with the server and display the order number, products, accepted vouchers, and total price on the terminal.
- **Pay Order**
    - Send the order details, including vouchers, products, and user ID, to the server for validation.
    - Calculate the final price of the order
- **Consult Transactions**
    - Allow users to view past transactions, including tickets and cafeteria orders, with receipts.
    - Update local information with transactions from the server.

The features we did not develop include pictures on the performance, biometric authentication, and vouchers.

Navigation Map



|__ Performances
| |
| |__ View next performances
| |__ Select performance
| |__ Choose quantity of tickets
| |__ Confirm purchase
|
|__ Tickets
| |
| |__ View purchased tickets
| |__ Identify validated tickets (highlighted)
|
|__ Cafeteria
| |
| |__ View cafeteria items
| |__ Select quantity of items
| |__ Purchase items
|
|__ Transactions
   |
   |__ View transaction history

Performed Scenario Tests

We conducted manual tests with our input, resulting in the refinement of certain conditions to operate within specific intervals. Additionally, it's important to note that the server operates asynchronously at all times

How to use

The app features a homepage displaying a title, with navigation facilitated through a sidebar menu. Users select from options like Register, Performances, Tickets, Cafeteria and Transactions.

When opting to register, users input personal information to create an account.

In Performances, users view upcoming events with details and select the number of tickets desired, followed by confirming the purchase.

Tickets displays purchased tickets, distinguishing validated ones in red.

Cafeteria showcases available items for purchase, allowing users to select quantities.

Transactions provides a log of all app transactions and associated prices.

Ticket validation triggers a QR code for verification.

Appendices

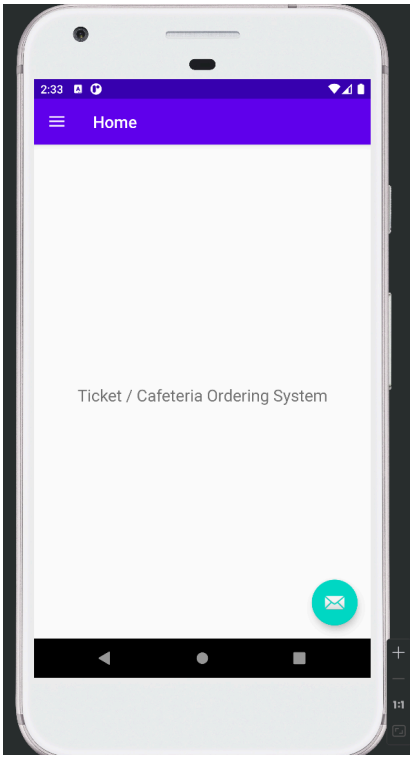


Fig.1 - Home Page

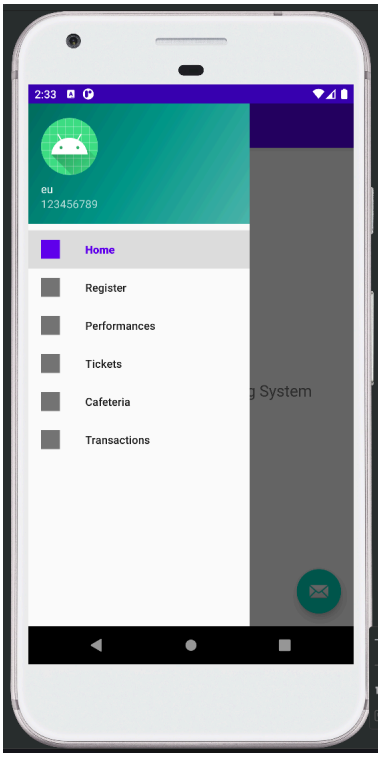


Fig.2 - Sidebar Menu

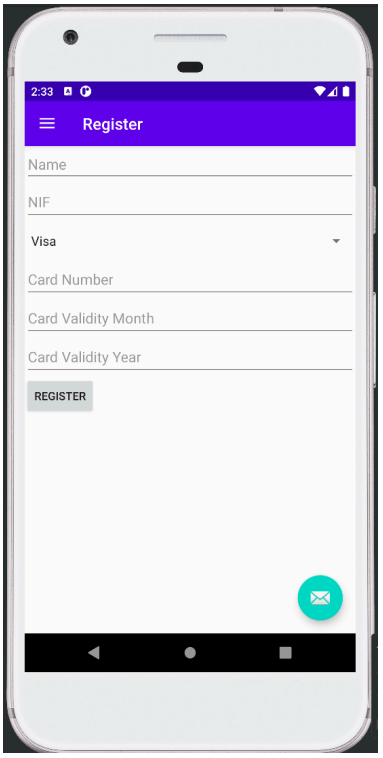


Fig.3 - Register Page

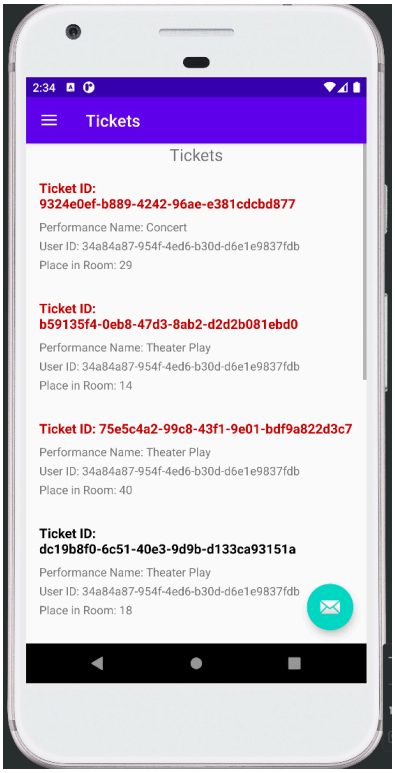


Fig.4 - Tickets Page

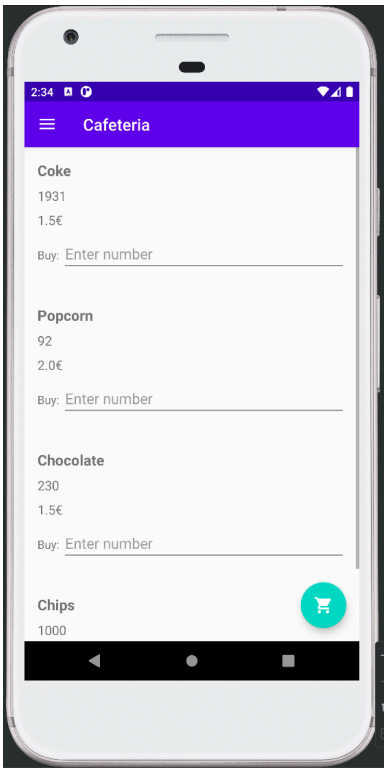


Fig.5 - Cafeteria Page

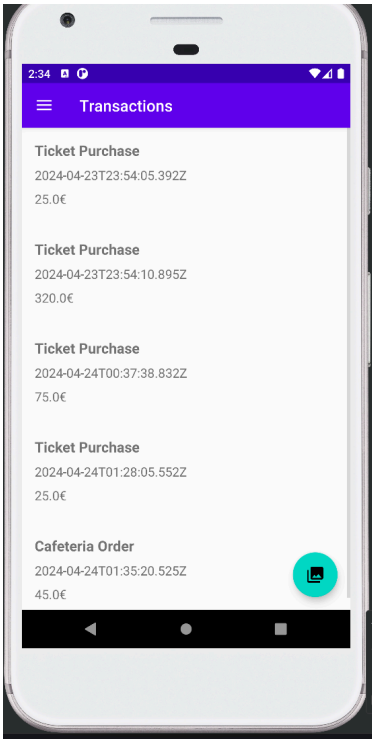


Fig.6 - Transactions Page

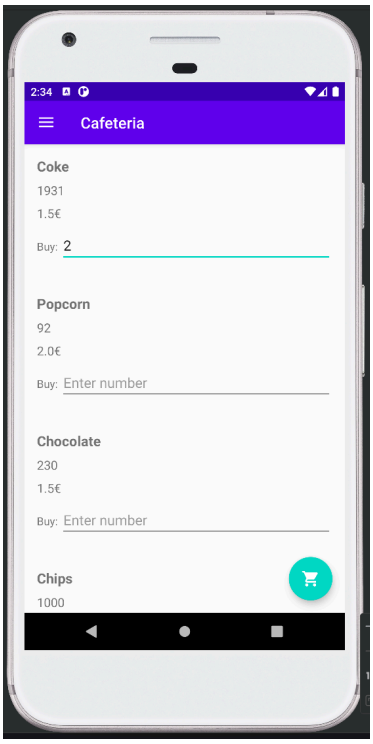


Fig.7 - Input Quantity

Fig.8 - QRCode

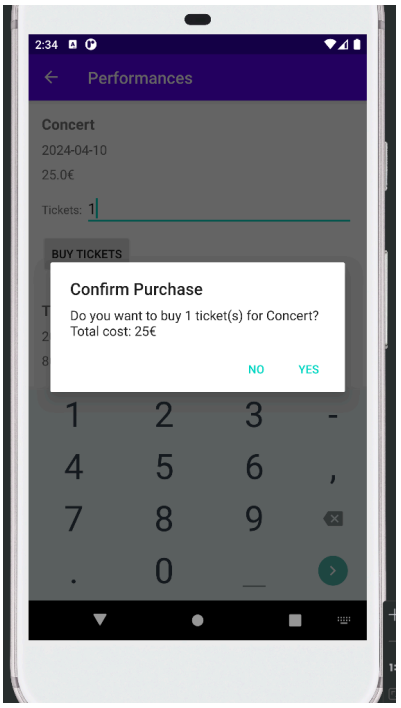


Fig.9 - Confirm Purchase Page

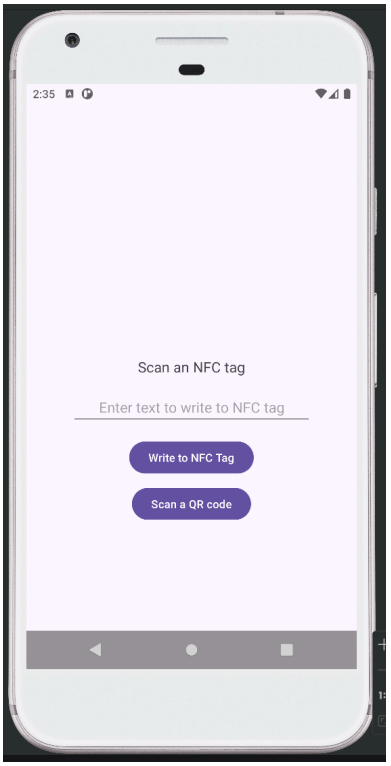


Fig.10 - Scan QRCode Page

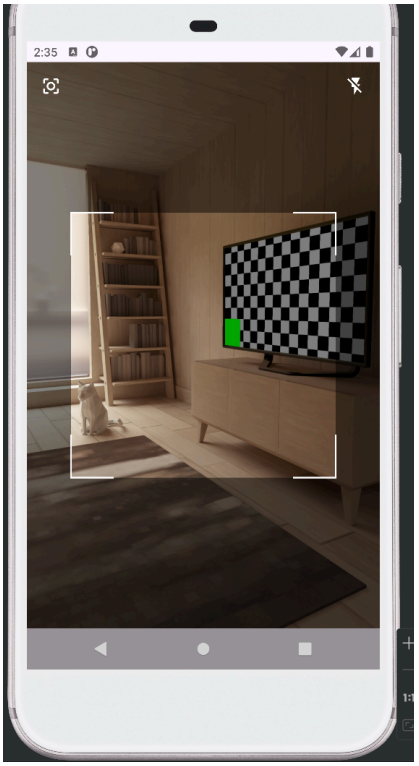


Fig.11 - Scanning