

WORKSHOP II

SOFIA LOZANO MARTINEZ
20211020088

JOSÉ JESÚS CÉSPEDES RIVERA
20211020118

CARLOS ANDRÉS SIERRA VIRGUEZ



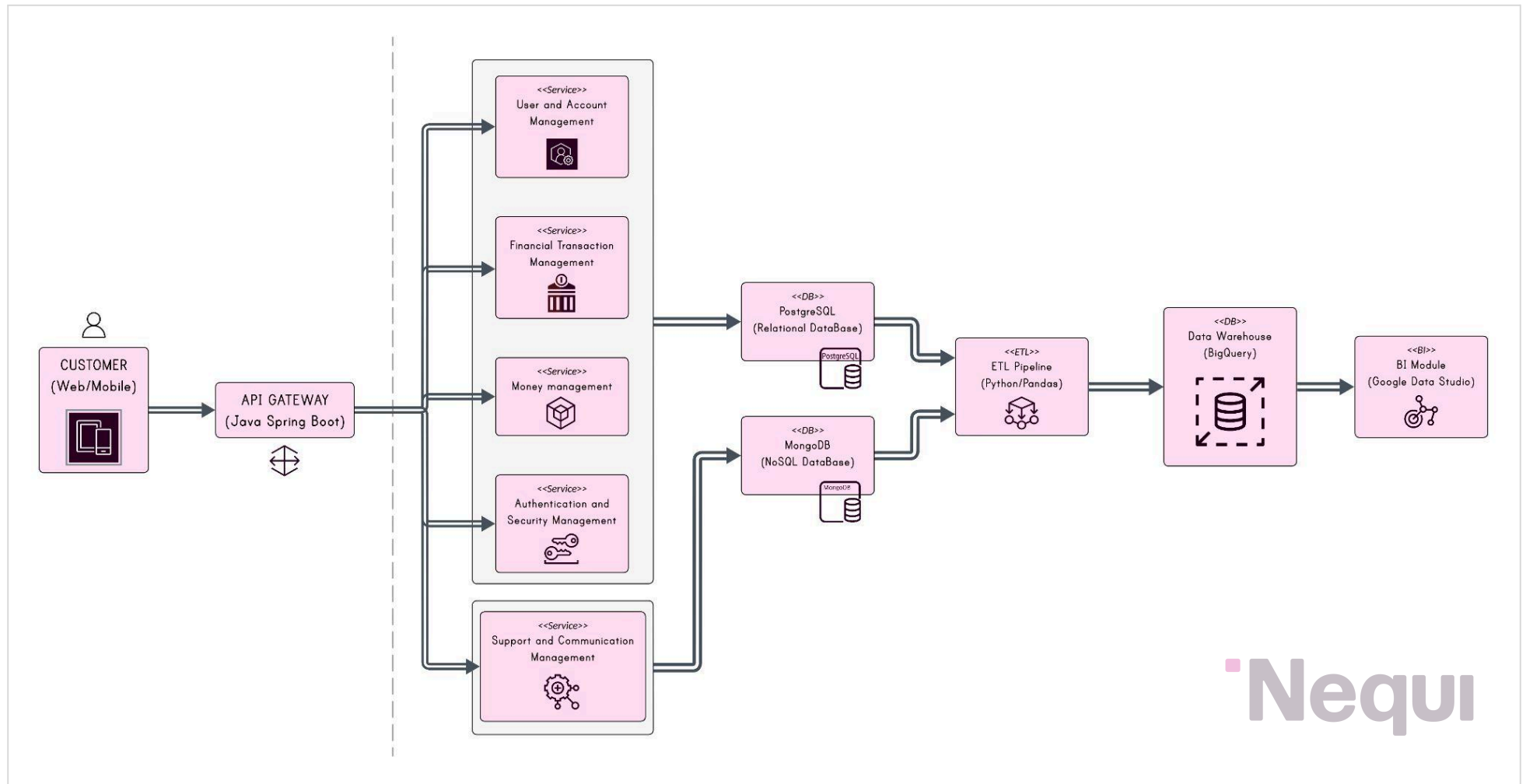
**UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS**

FRANCISCO JOSÉ DE CALDAS DISTRICT UNIVERSITY

FACULTY OF ENGINEERING
CURRICULAR PROJECT: SYSTEMS ENGINEERING
DATABASE II

BOGOTÁ, D.C., MAY 27
2025

DATA SYSTEM STRUCTURE



CUSTOMER (Web/Mobile)

Role: Represents the interface with the end user. Through mobile devices or browsers, customers interact with system features such as transactions, balance inquiries, and support.

API GATEWAY (Java Spring Boot)

Role: Single point of entry for customer requests. Responsible for authenticating, authorizing, and redirecting requests to the appropriate microservice, centralizing access logic.

Technology: Java Spring Boot was chosen for its ease of developing robust microservices and its mature ecosystem.

Microservices

Each specialized in a functional domain:

- User and Account Management: Management of user and account information.
- Financial Transaction Management: Processes payments, transfers, and financial transactions.
- Money Management: Allows the user to plan and visualize their financial situation.
- Authentication and Security Management: Responsible for credential validation and overall system security.
- Support and Communication Management: Channels user support requests and communicates with other support services.
 - A NoSQL database (MongoDB) was integrated for this component, allowing flexibility in storing free text (such as support messages), open requests, and conversation tracking. This decision responds to the need to process and store semi-structured data with high variability.

Databases

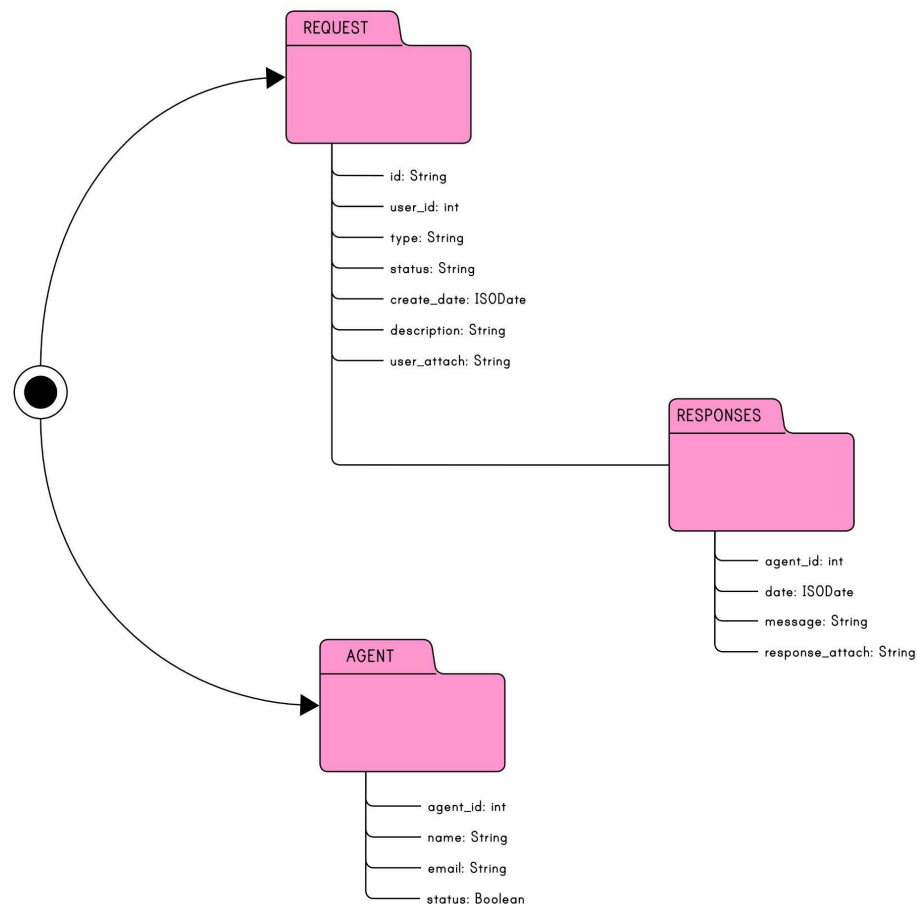
PostgreSQL (Relational):

- Role: Stores critical structured and transactional data (accounts, transactions, profiles).
- Justification: Requires referential integrity, ACID transactions, and well-defined relationships between entities.

MongoDB (NoSQL):

- Role: Stores semi-structured information, including logs, preferences, and dynamic data from the support module.
- Justification: Its document model allows greater flexibility and performance for data with changing schemas.

This model represents the data structure of the user support system, implemented with a document-oriented database (MongoDB). Its purpose is to manage Request (Petitions, Complaints and Claims) and their corresponding responses by customer service agents.



- **Main collection (REQUEST):** Each document in this collection represents a request made by a user, with the following key elements:
 - user_id: Reference to the user's identifier (from the relational system).
 - type: Type of request (complaint, request, etc.).
 - status: Status of the case (in_progress, resolved, etc.).
 - description: Detailed text of the request.
 - user_attachment: Embedded list of files sent by the user.
 - responses: Embedded list of responses from agents, each one with:
 - agent_id: Agent identifier (reference to agent collection).
 - message: Textual response.
 - attachment: Files sent by the agent in its response.

- **Agente Collection:** Contains the information of the support agents:
 - id
 - name
 - email
 - status

ETL Pipeline (Python/Pandas)

Role: Extracts information from PostgreSQL and MongoDB, performs transformations (cleansing, aggregation), and loads it into the data warehouse.

Justification: Python and Pandas are powerful tools for data manipulation and allow for easy automation of the process.

Data Warehouse (BigQuery)

Role: Central repository for analyzing large volumes of historical and aggregated data.

Justification: Its ability to scale horizontally and execute SQL queries at high speed makes it ideal for BI.

BI Module (Google Data Studio)

Role: Visualization tool for generating dynamic dashboards and reports from BigQuery.

Justification: Intuitive interface, native integration with BigQuery, and easy sharing of visualizations.

Data Flow

1. The client initiates an interaction from their mobile app or website.
2. The request is managed by the API Gateway, which channels it to the corresponding microservice.
3. The microservices access PostgreSQL for structured operations (users, accounts, transactions) and MongoDB for unstructured data (support, events).
4. The ETL Pipeline extracts data from both databases, transforms it, and loads it into the Data Warehouse (BigQuery).
5. The analytics team accesses it through the BI Module (Google Data Studio) to visualize key metrics and generate reports.

INFORMATION TO BE RETRIEVED BY THE SYSTEM

The following outlines the main types of information the system must retrieve in order to support the functionalities described in the user stories. This information aligns with the modeled entities and responds to business needs.

1. User Registration and Authentication (HU01, HU08)

To allow the registration of new users and ensure secure authentication, the system must be able to retrieve the following information:

- Verify whether a user is already registered by document or phone number (User.user_id, User.phone).
- Retrieve authentication data, such as the securely stored PIN and biometric method, if configured (Authentication.PIN, Authentication.biometrics).
- Retrieve the devices registered by the user and their current status (Device.status), to validate whether the device is trusted or if it is a new access attempt.

2. Balance Inquiry and Account Movements (HU02)

To allow users to check their available balance and add money to their account, the system must retrieve:

- The current balance of the account (Account.balance).
- The transaction history over a specific period.
- The status of the account (Account.status), indicating whether it is active or blocked.

3. Money Transfers (HU03)

To facilitate the sending and receiving of money between users, the system must retrieve:

- The user's available balance (Account.balance) to validate whether the transfer can be completed.
- The recipient's information, including name and account status (User.full_name, Account.status), in case of a Nequi-to-Nequi transfer.
- The history of transfers made and received (Movement with type transfer), to be displayed in the user interface.

4. Payments to Merchants and Services (HU04, HU07)

This functionality allows users to make payments at merchants and for public services directly from their Nequi account. The system must retrieve:

- The available balance in the main account (Account.balance) to validate the availability of funds.

- The active card associated with the account, including its status and available limit (Card.status, Card.limit), if the payment is to be made using a card.
- The history of completed payments, including those made to merchants and services (Movement with type bill_payment).
- The reference or name of the merchant or service paid (Movement.reference), to display transaction details to the user.

5. Expense Management with Pockets and Cushion (HU05)

To provide users with tools that help them better organize their money through the use of pockets and the cushion, the system must retrieve:

- The list of pockets created by the user (Poken) and the available balance in each (Poken.balance).
- The current amount saved in the cushion (Cushion.balance) and whether it is locked or not (Cushion.locked).

6. Credit Management (HU06)

To support credit application and management functionality, the system must retrieve:

- The credit records associated with the user (Credit by account_id).
- The requested amount, disbursement status, and repayment terms (Credit.amount, Credit.status, Credit.term).
- The credit history and disbursement movements related to the loan (Movement with type credit).
- The account balance to verify if the funds from the credit have been received (Account.balance).

7. Support Management (HU08)

To provide user assistance and ensure proper tracking of their requests, the system must retrieve:

- The support tickets associated with the user (Support by user_id).
- The type of request submitted (query, report, response) (Support.type).
- The current status of the ticket (Support.status), to determine whether it is in progress, resolved, or closed.
- The full description of the ticket and its update history if applicable (Support.description).

QUERY PROPOSALS

POSTGRESQL - SQL

1. User Registration and Authentication

Allows the system to verify if a user already exists and manage secure authentication. Provides information about user identity, login credentials, and registered trusted devices.

- Query to verify if the user is already registered by document or phone number.

```
SELECT user_id, ID_document, phone
FROM User
WHERE ID_document = 'DOCUMENT' OR phone = 'PHONE_NUMBER';
```
- Query to retrieve the user's credentials (PIN and biometrics) from the Authentication entity.

```
SELECT PIN, biometrics
FROM Authentication
WHERE user_id = 'USER_ID';
```
- Query to retrieve all devices associated with the user, including their status and usage dates.

```
SELECT device_id, registration_date, last_used, status
FROM Device
WHERE user_id = 'USER_ID';
```

2. Balance Inquiry and Account Movements

Allows the user to check their balance and see how their account is moving—what their expenses have been and what their sources of income are.

- Query to retrieve the current balance and status (active or blocked) of the account.

```
SELECT balance, status
FROM Account
WHERE account_id = 'ACCOUNT_ID';
```
- Query to retrieve the account's transaction history over a specific period.

```
SELECT movement_id, amount, date_time, type_movement, status,
reference
FROM Movement
WHERE source_account_id = 'ACCOUNT_ID'
AND date_time BETWEEN '2025-05-01' AND '2025-05-31'
ORDER BY date_time DESC;
```

3. Money Transfers

Allows validation of available funds for transfer and provides recipient details. Also offers a transfer history useful for personal tracking or resolving disputes.

- Query to verify the available balance before making a transfer.

```
SELECT balance
FROM Account
WHERE account_id = 'ACCOUNT_ID';
```
- Query to retrieve the recipient's name and account status in a Nequi-to-Nequi transfer.

```
SELECT u.full_name, a.status
FROM User u
JOIN Account a ON u.user_id = a.user_id
WHERE a.account_id = 'DESTINATION_ACCOUNT_ID';
```
- Query to display transfers made or received, useful for the user interface.

```
SELECT movement_id, amount, date_time, reference
FROM Movement
WHERE source_account_id = 'ACCOUNT_ID'
AND type_movement = 'transfer'
ORDER BY date_time DESC;
```

4. Payments to Merchants and Services

Enables control over card usage, verifies limits, and retrieves payment history to merchants and public services. Provides transparency by showing the amount, provider, and date of each transaction.

- Query to retrieve the balance before making a payment.

```
SELECT balance
FROM Account
WHERE account_id = 'ACCOUNT_ID';
```
- Query to validate if the card is active and what its limit is.

```
SELECT card_id, status, limit
FROM Card
WHERE account_id = 'ACCOUNT_ID' AND status = 'active';
```
- Query to retrieve payments made to public services, along with their reference.

```
SELECT m.movement_id, m.amount, m.date_time, m.reference,
bp.service_company
FROM Movement m
JOIN BillPayment bp ON m.movement_id = bp.movement_id
WHERE m.source_account_id = 'ACCOUNT_ID'
ORDER BY m.date_time DESC;
```

5. Expense Management with Pockets and Cushion

Allows the user to organize their money into categories (pockets) or reserve it (cushion). Offers visibility into how the user manages and distributes their finances.

- Query to check the amount stored in the cushion and whether it is locked.

```
SELECT balance, locked
FROM Cushion
WHERE account_id = 'ACCOUNT_ID';
```

6. Credit Management

Provides access to active or requested loans, including terms and current status. Useful for tracking debt, due dates, interest rates, and the user's financial compliance.

- Query to retrieve the credits requested by the user, along with their status and conditions.

```
SELECT movement_id, credit_type, interest, term, due_date,
       credit_status
FROM Credit
WHERE movement_id IN (
    SELECT movement_id
    FROM Movement
    WHERE source_account_id = 'ACCOUNT_ID'
);
```

- Query to display disbursements or transactions related to granted credits.

```
SELECT amount, date_time, status
FROM Movement
WHERE source_account_id = 'ACCOUNT_ID' AND type_movement =
'credit';
```

MONGODB - NOSQL

7. Support Management

- The support tickets associated with the user (Support by user_id).
db.request.find({ user_id: "user_id" })
- The type of request submitted (query, report, response) (Support.type).
db.request.find({ type: "type" })
db.request.distinct("type") – This query allows you to view all types of existing requests.
- The current status of the ticket (Support.status), to determine whether it is in progress, resolved, or closed.

```
db.pqr.aggregate([
    { $group: { _id: "$status", count: { $sum: 1 } } }
])
```

WORKSHOP 1 CHANGES

The money management component was added, which is composed of the new entities added to the model: pocket, mattress, and card, which are related to the account entity.

Additionally, attributes were added to the device (login and status) and transaction (payment type) entities.

Initial Database Architecture

1. Components:

1. **User and Account Management**

Responsible for storing and managing user personal information and account details, including balances, limits, types, and account status.

2. **Financial Transaction Management**

Records and monitors all transactions carried out by users, such as transfers, payments, credit ("Crédito Propulsor" and "Crédito Salvavidas"), and withdrawals, enabling detailed tracking of financial flows.

3. **Money management**

Manage the ways in which the user organizes, restricts, or uses their money within their Nequi account. This component doesn't represent a single entity, but rather a set of interrelated entities that operate within an Account, such as: wallet, mattress, and virtual card.

4. **Authentication and Security Management**

Controls access mechanisms to the system, including the use of PIN, biometrics, and device verification, ensuring data integrity and confidentiality.

5. **Support and Communication Management**

Manages interactions between users and the support area, including inquiries, reports, responses, and the customer communication history.

2. Entities:

1. **User and Account Management**

- a. User
- b. Account
- c. Device

2. **Financial Transaction Management**

- a. Movement
- b. Transfer
- c. BillPayment
- d. TopUp

- e. Withdrawal
- f. Credit

3. Money Management

- a. Pocket
- b. Cushion
- c. Card

4. Authentication and Security Management

- a. Authentication

3. Attributes per Entity:

a. User

- user_id (PK)
- full_name
- ID_document
- phone
- email
- country_of_residence
- registration_date
- status

b. Account

- account_id (PK)
- user_id (FK)
- type (low amount, savings)
- balance
- monthly_limit
- status
- creation_date
- exempt_4x1000

c. Device

- device_id (PK)
- user_id (FK)
- registration_date
- last_used
- status

d. Movement

- movement_id (PK)
- source_account_id (FK)

- amount
- date_time
- status
- type
- reference
- source_type

e. Transfer

- movement_id (PK, FK)
- destination_account_id (FK)
- type

f. Withdrawal

- movement_id (PK, FK)
- channel

g. BillPayment

- movement_id (PK, FK)
- service_company
- reference_number

h. Credit (Loan)

- movement_id (PK, FK)
- credit_type (lifeline, booster)
- interest
- term
- due_date
- credit_status

i. Pocken

- pocked_id (PK)
- account_id(FK)
- name
- balance
- creation_date
- status

j. Cushion

- cushion_id(FK)
- account_id
- balance
- locked
- creation_date

k. Card

- card_id
- account_id(FK)
- card_number
- expiry_date
- cvv_hash
- status
- issued_date
- last_used
- provider
- limit
- currency



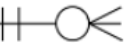
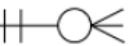
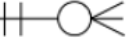
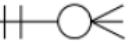
l. Authentication

- auth_id (PK)
- user_id (FK)
- PIN
- biometrics

4. Relationships


	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>
<i>a</i>		✓	✓									✓
<i>b</i>	✓			✓					✓	✓	✓	
<i>c</i>	✓											
<i>d</i>		✓			✓	✓	✓	✓				
<i>e</i>				✓								
<i>f</i>				✓								
<i>g</i>				✓								
<i>h</i>				✓								
<i>i</i>		✓										
<i>j</i>		✓										
<i>k</i>		✓										
<i>l</i>	✓											

5. Relationships types

- User (a)  Account (b): (1:1)
A user has one account, and one account belongs to only one user.
- User (a)  Device (c): (1:N)
A user can register one or multiple devices, but each device is associated with only one user.
- Account (b)  Movement (d): (1:N)
An account can have zero or multiple movements, but each movement is associated with only one account.
- Movement (d)  Transfer (e): (1:1)
Each transfer is a specific type of movement.
- Movement (d)  Withdrawal (f): (1:1)
Each withdrawal is a specific type of movement.
- Movement (d)  BillPayment (g): (1:1)
Each bill payment is a specific type of movement.

7. Movement (d)  Credit (h): (1:1)

Each loan is a specific type of movement.

8. Account (b)  Pocket(i): (1:N)


A user can have zero or multiple pockets, but each pocket is associated with only one account.

9. Account (b)  Cushion(j): (1:1)

A user can have zero or one cushion, but each cushion is associated with only one account.

10. Account (b)  Card(k): (1:1)

A user can have zero or one card, but each card belongs to a single account.

11. User (a)  Authentication (l): (1:1)

Each user has a unique authentication record, and each authentication belongs to only one user.

6. First Entity-Relationship Model Draw

