Bank Challenge

Julián Darío Luna Patiño

Justification

Quality Attributes

Evaluated

- **Security - Confidentiality:**
  1. **Scenario:** An attacker tries to access sensitive bank account information.
  2. **Justification:** Protecting users' financial and personal information is essential. A security breach could have serious legal consequences and harm the bank's reputation.
- **Security - Integrity:**
  1. **Scenario:** An attacker tries to alter transactions or account balances.
  2. **Justification:** Maintaining transaction integrity is crucial to ensuring customers' funds are handled correctly.
- **Availability:**
  1. **Scenario:** A user tries to access their account or make a transaction during a peak demand period.
  2. **Justification:** Users expect the service to be always available, especially during high demand periods like payday or billing deadlines.
- **Performance:**
  1. **Scenario**: A user makes a transaction and expects an immediate system response.
  2. **Justification:** A slow banking system can frustrate users and cause them to lose trust in the bank.
- **Scalability:**
  1. **Scenario:** The number of banking system users increases significantly in a short period.
  2. **Justification:** The system must be able to handle growth in user numbers and transactions without degrading performance.
- **Resilience and Recoverability:**
  1. **Scenario**: A system component fails, such as a server crash.
  2. **Justification:** The system must be able to recover quickly from failures and ensure no transaction is lost.

- **Auditability:**
    1. **Scenario:** A detailed log of all transactions for a particular account is requested.
    2. Justification: Being able to track and audit all system actions is crucial to comply with regulations and investigate any suspicious activity.
- **Interoperability:**
    1. **Scenario:** Banking software needs to interact with other systems, such as payment systems or credit reporting systems.
    2. Justification: Banks often interact with a variety of other systems and services, and it is essential they do so seamlessly.
- **Usability:**
    1. **Scenario**: A new user tries to open an account or make a transaction for the first time.
    2. Justification: An intuitive and user-friendly design will enhance customer satisfaction and reduce the number of errors or issues.
- **Upgradability:**
    1. **Scenario:** A new feature needs to be implemented or a bug in the system fixed.
    2. Justification: The ability to update the system quickly and seamlessly is crucial for keeping it modern and secure.

Prioritized

1. **Scalability:**
    - **Scenario:** The system must be capable of handling a substantial increase in the number of users and transactions, moving from 30,000 users to 1 million in one year and three months.
    - **Justification:** As the user count grows, the system must be able to manage the additional load without degrading performance. This involves not only accommodating more users but also a proportional increase in transactions, queries, and related operations.
2. **Performance**:
    - **Scenario:** With a significant surge in demand, the system needs to maintain swift and consistent response times, even during demand spikes.
    - **Justification:** Slow performance can lead to a poor user experience, potentially deterring new users from joining

or retaining current ones. Maintaining optimal performance is crucial to ensure customer satisfaction and operational efficiency.

3. **Resilience and Recoverability:**
    - **Scenario:** Given the increase in demand and complexity, the system must be able to recover swiftly from failures and ensure service continuity.
    - **Justification:** As the user base expands, any system downtime or failure will have a magnified impact. It's crucial for the system to detect issues, self-recover, or at least have mechanisms in place for rapid recovery with minimal manual intervention.

Quality Scenarios

Scalability:

- Case in which it occurs: The number of active users increases from 30 thousand to 1 million in a period of 1 year and 3 months.
- Affected module: Complete system, including authentication, transaction management, user interface, database.
- Expected unit of measurement: Number of concurrent users that the system can handle without performance degradation.
- Ideal value: 1 million concurrent users.
- Trade-off:
    - Counter Security: As you optimize for scalability, you can introduce more entry points and more surface area for potential attacks.
    - Against Maintainability: Optimizing for scalability can lead to more complex solutions that may be more difficult to maintain or modify in the future.

Performance:

- Case in which it occurs: Users perform transactions, queries and other operations in the system.
- Module affected: Database operations, API, frontend.
- Expected unit of measurement: Response time in seconds.
- Ideal value: Response in less than 2 seconds for 95% of operations.
- Trade-off:
    - Against Security: Optimizing for performance may lead to skipping some security checks to speed up operations.
    - Against Cost: Improving performance could require more hardware resources or more expensive solutions.

Resilience:

- Case in which it occurs: Failure in one or more components of the system.
- Affected module: All modules, including services, databases and networks.
- Expected unit of measurement: Number of failures that the system can handle without service interruption.
- Ideal value: The system should be able to handle the failure of any individual component without service interruption.

- Trade-off:
    - Against Cost: Increasing resilience could require redundancy in hardware or services, which would increase costs.
    - Counter Performance: Some resilience strategies may introduce additional latency.

Recoverability:

- Case in which it occurs: After a catastrophic failure or a major error.
- Affected module: Complete system, including backups and logs.
- Expected unit of measurement: Time required to recover service after a failure.
- Ideal value: Complete recovery in less than 1 hour.
- Trade-off:
    - Against Cost: Maintaining frequent backups or rapid recovery solutions can be expensive.
    - Performance Against: Some recoverability solutions may introduce latency or degrade performance during normal operations.

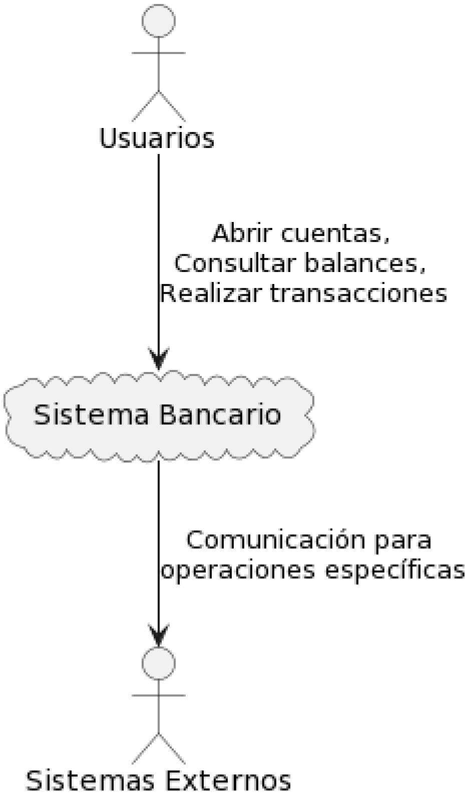
Patrones

  - **Scalability:**
    - **Recommended patterns::**
      - Design) Microservices: Decompose the application into smaller, independent services that communicate with each other. This allows individual components to be scaled based on demand.
      - (Design) Load Balancing: Distribute incoming traffic to the application across multiple instances to avoid overloading a single server.
      - (Design) Partitioned Database (Sharding) or CQRS: Divide the database into smaller fragments and distribute them to improve performance and scalability.
      - (Design) Message Queues: Decouple components and handle background operations, allowing the application to handle large volumes of traffic.
  - **Performance:**
    - **Recommended patterns:**
      - (Design) Cache: Use caching solutions (such as Redis or Memcached) to store frequently consulted data and reduce access time.
      - (Design) Database Query Optimization: Ensure that database queries are optimized and use indexes appropriately.
      - (implemented) Pagination: In very long queries, making sure to have pagination optimizes the system response time.
      - (Design) Content Delivery Network (CDN): Distribute content to multiple locations and serve content to the user from the nearest location.
      - (Design) Asynchronous Computing: Perform intensive operations in the background so as not to block the main flow of the application.

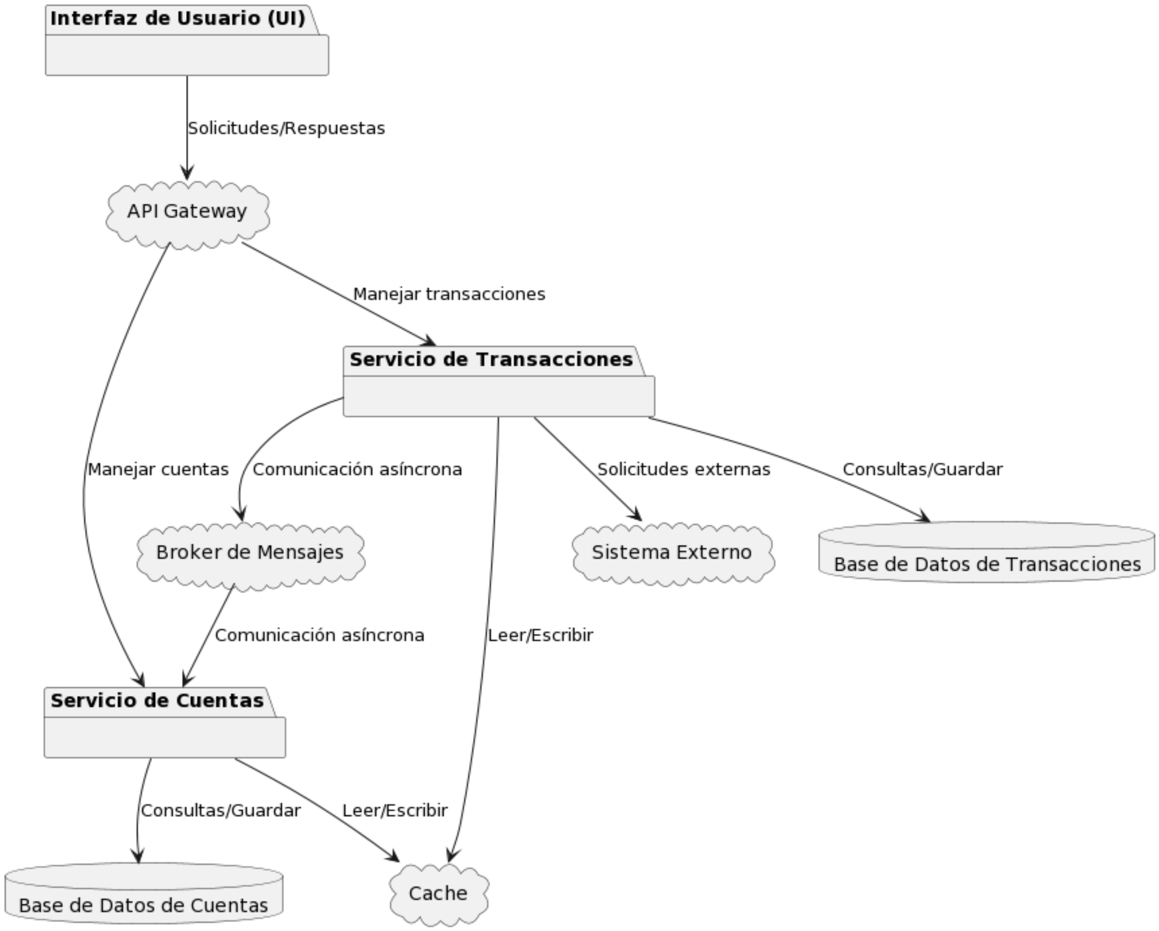
- **Resilience:**

    - **Recommended patterns:**
        - (Design) Circuit Breaker: Detect failures and prevent the system from making requests that are likely to fail, allowing the system to recover.
        - (Design) Bulkhead: Isolate elements of the system, so that if one fails, it does not cause a cascading failure throughout the system.
        - (Design) Retry Pattern: In case of failure in an operation, try the operation again a defined number of times.
        - (Design) Timeouts: Establish maximum waiting times for operations, preventing the system from being in a blocked state indefinitely.
- **Recoverability:**
    - **Recommended patterns:**
        - (Design) Backup and Restore: Make regular backup copies of data and ensure that there is a process in place to restore that data in case of failures.
        - (Design) Journaling: Record operations in a journal before they are applied, allowing the system to recover in case of failures.
        - (Design) Snapshot: Take snapshots of the system state at regular intervals to facilitate recovery.
        - (Design) Stateless Components: Design stateless components so that they can be easily replaced or recovered without data loss.

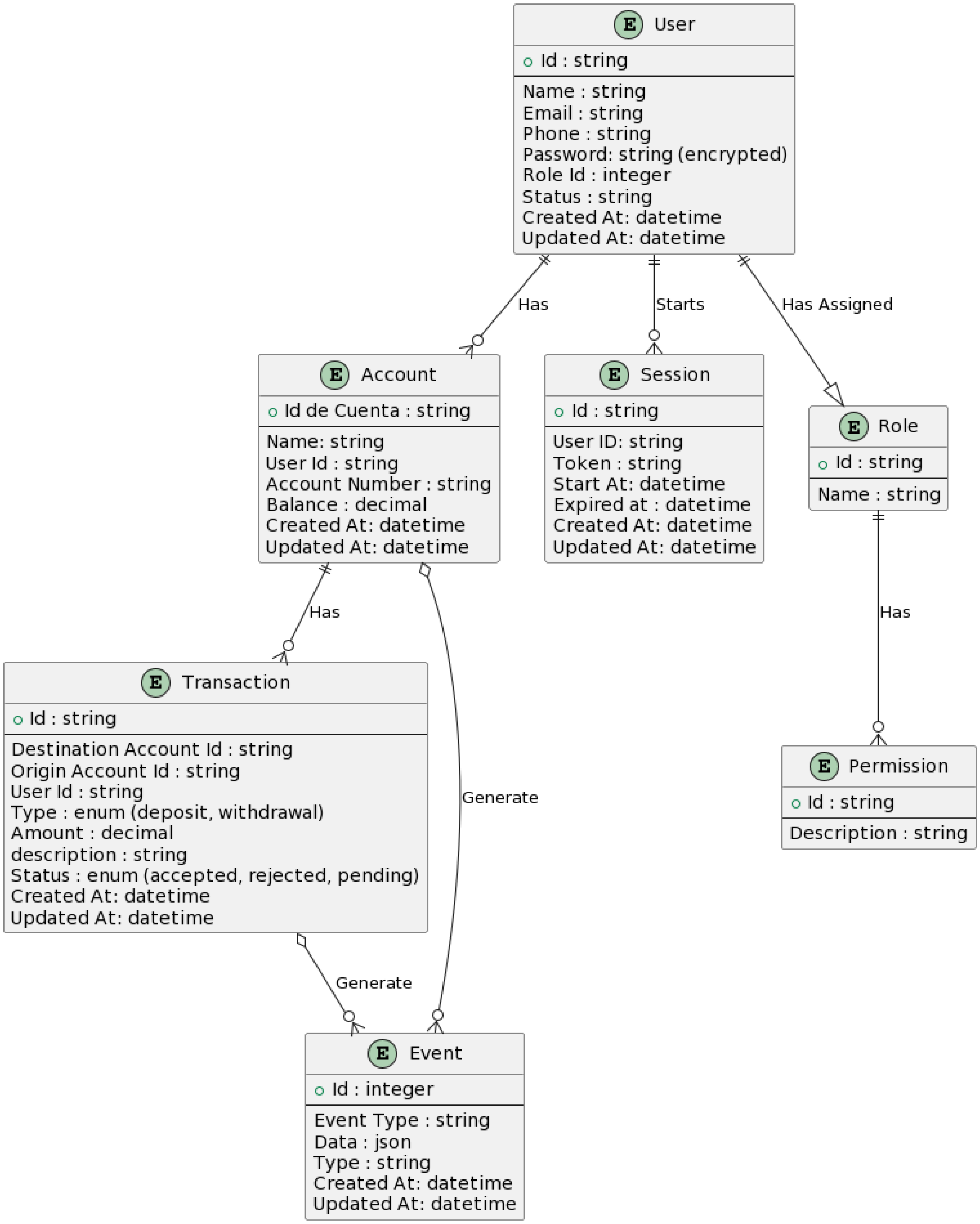
Context View



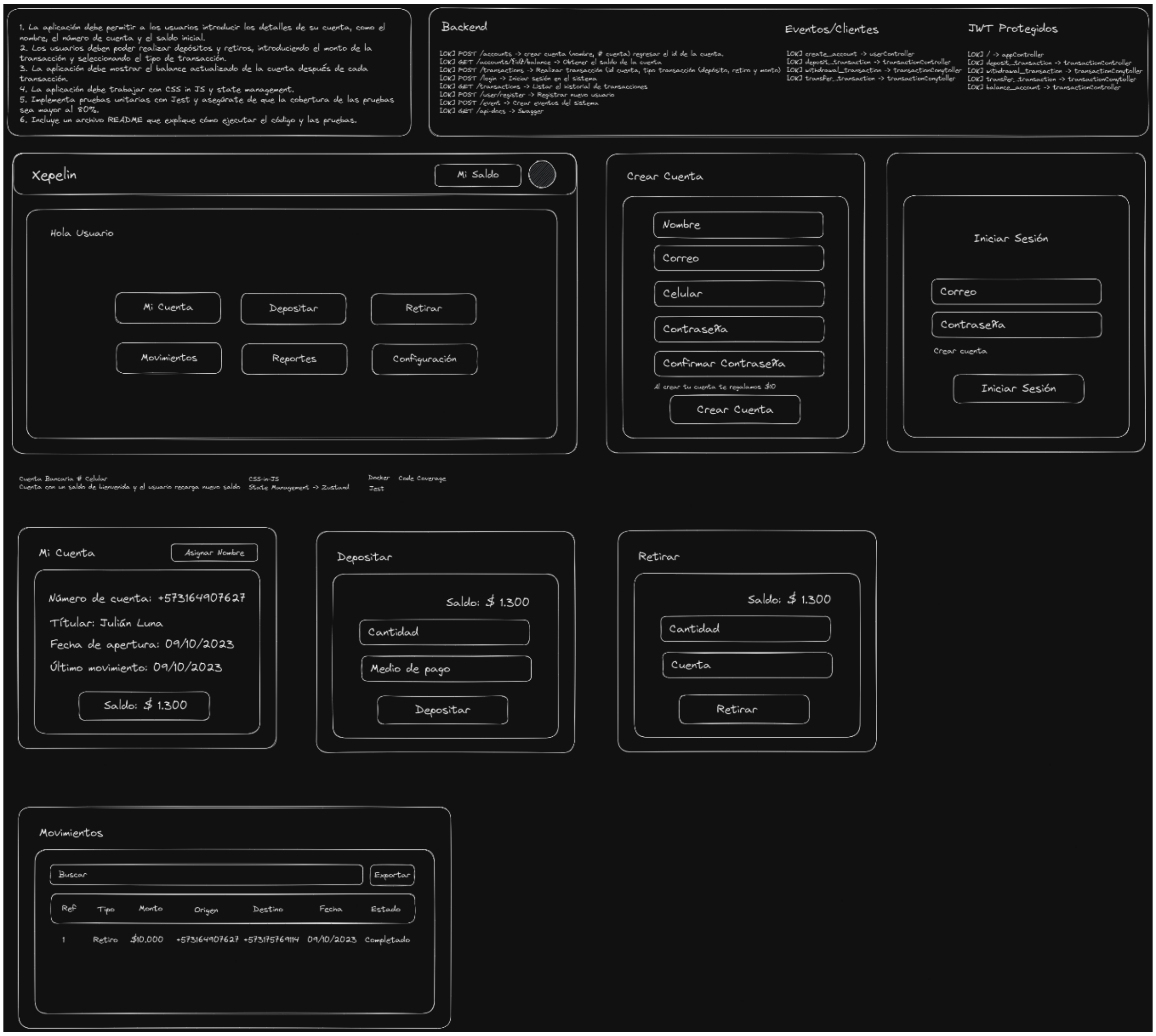
Vista de Componentes



Information View



Mockup



1. La aplicación debe permitir a los usuarios introducir los detalles de su cuenta, como el nombre, el número de cuenta y el saldo inicial.
2. Los usuarios deben poder realizar depósitos y retiros, introduciendo el monto de la transacción y seleccionando el tipo de transacción.
3. La aplicación debe mostrar el balance actualizado de la cuenta después de cada transacción.
4. La aplicación debe trabajar con CSS in JS y state management.
5. Implementa pruebas unitarias con Jest y asegúrate de que la cobertura de las pruebas sea mayor al 80%.
6. Incluye un archivo README que explique cómo ejecutar el código y las pruebas.

Backend

[OK] POST /accounts -> crear cuenta (nombre, # cuenta) regresar el id de la cuenta.
[OK] GET /accounts/{id}/balance -> Obtener el saldo de la cuenta
[OK] POST /transactions -> Realizar transacción (id cuenta, tipo transacción (depósito, retiro y monto)
[OK] POST /login -> Iniciar sesión en el sistema
[OK] GET /transactions -> Listar el historial de transacciones
[OK] POST /user/register -> Registrar nuevo usuario
[OK] POST /event -> Crear eventos del sistema
[OK] GET /api-docs -> Swagger

Eventos/Clientes

[OK] create_account -> userController
[OK] deposit_transaction -> transactionController
[OK] withdrawal_transaction -> transactionConytoller
[OK] transfer_transaction -> transactionConytoller

JWT Protegidos

[OK] / -> appController
[OK] deposit_transaction -> transactionController
[OK] withdrawal_transaction -> transactionConytoller
[OK] transfer_transaction -> transactionConytoller
[OK] balance_account -> transactionController

Al crear tu cuenta te regalamos $10

Cuenta Bancaria # Celular
Cuenta con un saldo de bienvenida y el usuario recarga nuevo saldo

CSS-in-JS
State Management -> Zustand

Docker
Jest

Code Coverage

Número de cuenta: +573164907627
Títular: Julián Luna
Fecha de apertura: 09/10/2023
Último movimiento: 09/10/2023

Ref	Tipo	Monto	Origen	Destino	Fecha	Estado
1	Retiro	$10.000	+573164907627	+573175769114	09/10/2023	Completado

To do

Frontend

1. User Interface

- Adjust the pagination buttons.
- Improvements in form validations.
- Pending implementation of notification modals.
- Mobile First.

2. Functionality and Logic

- Allow selecting dynamic pagination.
- Detect token expiration.
- Pending assigning account name.

3. Tests

- Unit and coverage tests.
- Optimization and Maintenance
- Improve aliased routes.

Backend

1. Security

- Improve the implementation of the JWT Guard module.
- Delimit that only if the user is the owner of the resource can they carry out actions.
- Implement the session, role and permission tables to improve security and registration.

2. Tests

- Implement unit and coverage tests.
- Improve the implementation of unit and coverage tests.

3. Documentation and Monitoring

- Improve the definition of swagger.
- Implement monitoring tool (Grafana).

4. Database and ORM

- Implement ORM relationships.
- Implement migrations.
- Improve account registration, instead of saving account numbers, saving the account UUID and doing the conversion.

5. Errors and Logging

- Improve the detail and implementation of error handling cases in various parts of the system.
- Implement logger.

6. Architecture and Design

- Implement interfaces in controllers, userCases, etc.
- Deploy Kafka to replace clients.

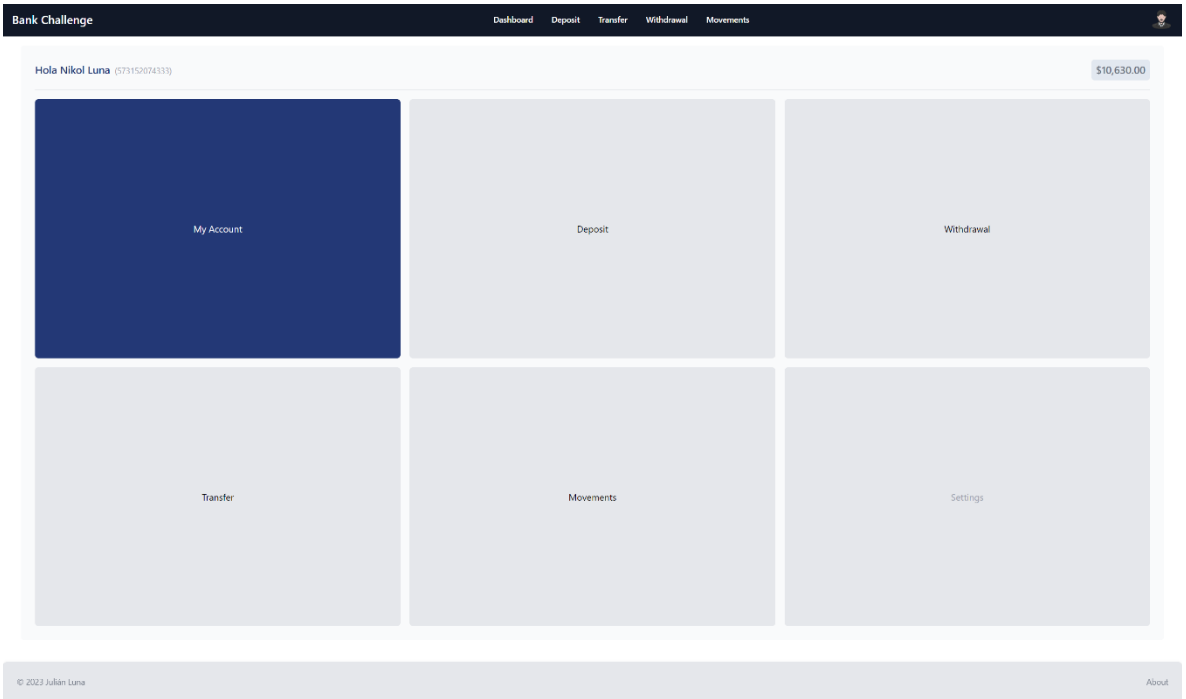
7. General Improvements

- Implement monitoring tool (Grafana).

Architecture

- AWS Diagram

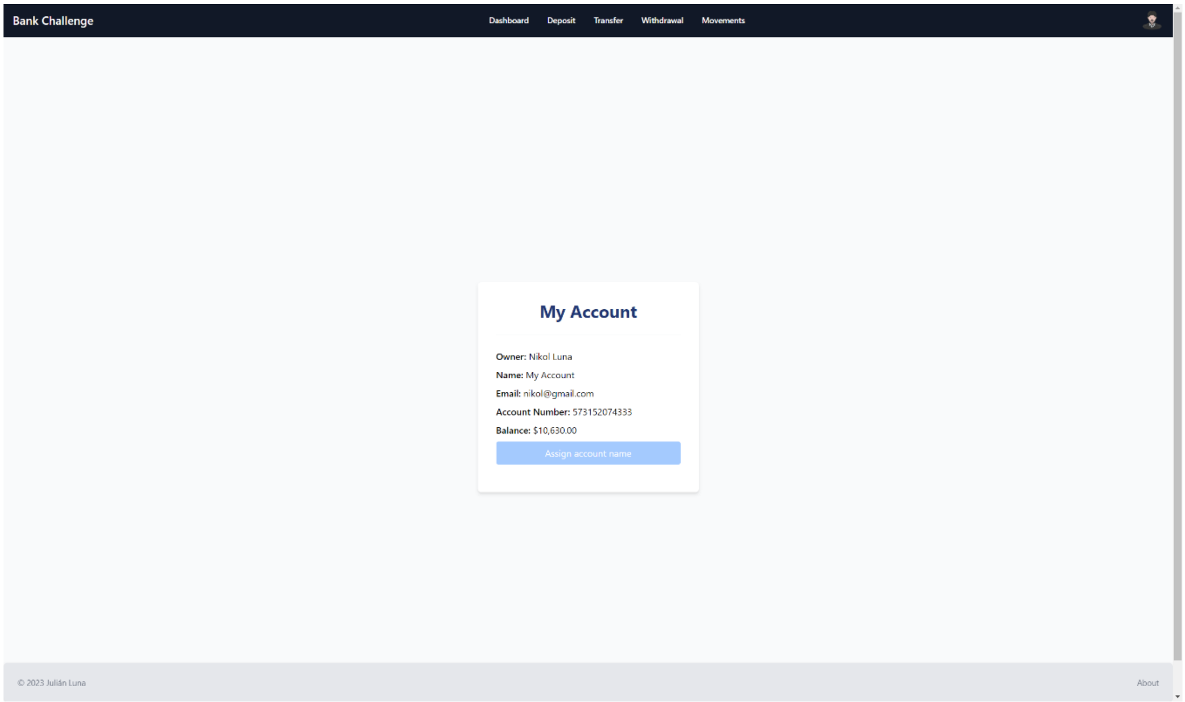
ScreenShots



Hola Nikol Luna (573152074333)

$10,630.00

© 2023 Julián Luna



My Account

Owner: Nikol Luna

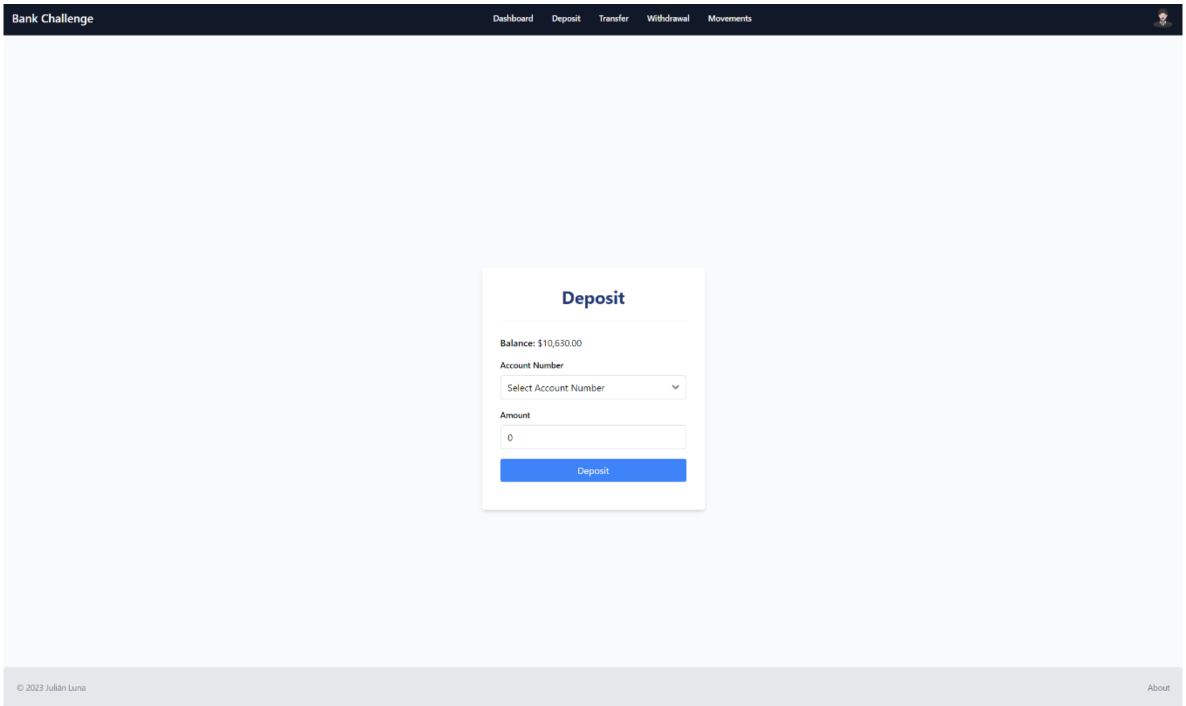
Name: My Account

Email: nikol@gmail.com

Account Number: 573152074333

Balance: $10,630.00

© 2023 Julián Luna

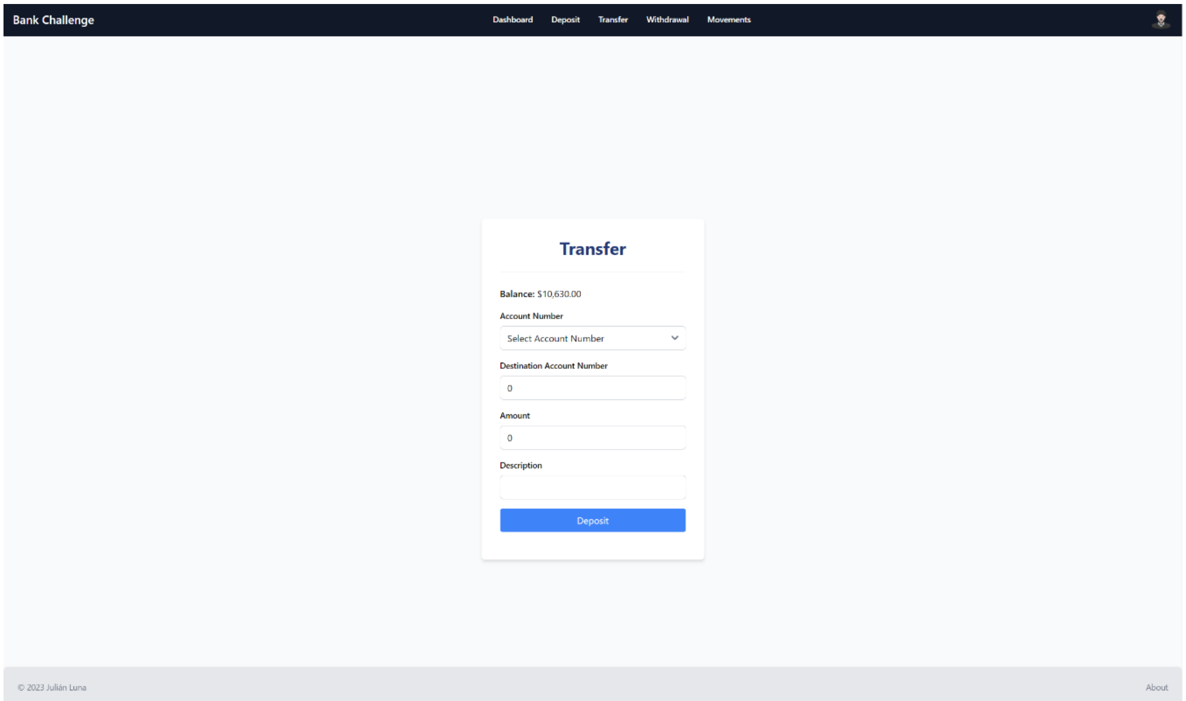


Deposit

Balance: $10,630.00

Account Number: Select Account Number

Amount: 0



Transfer

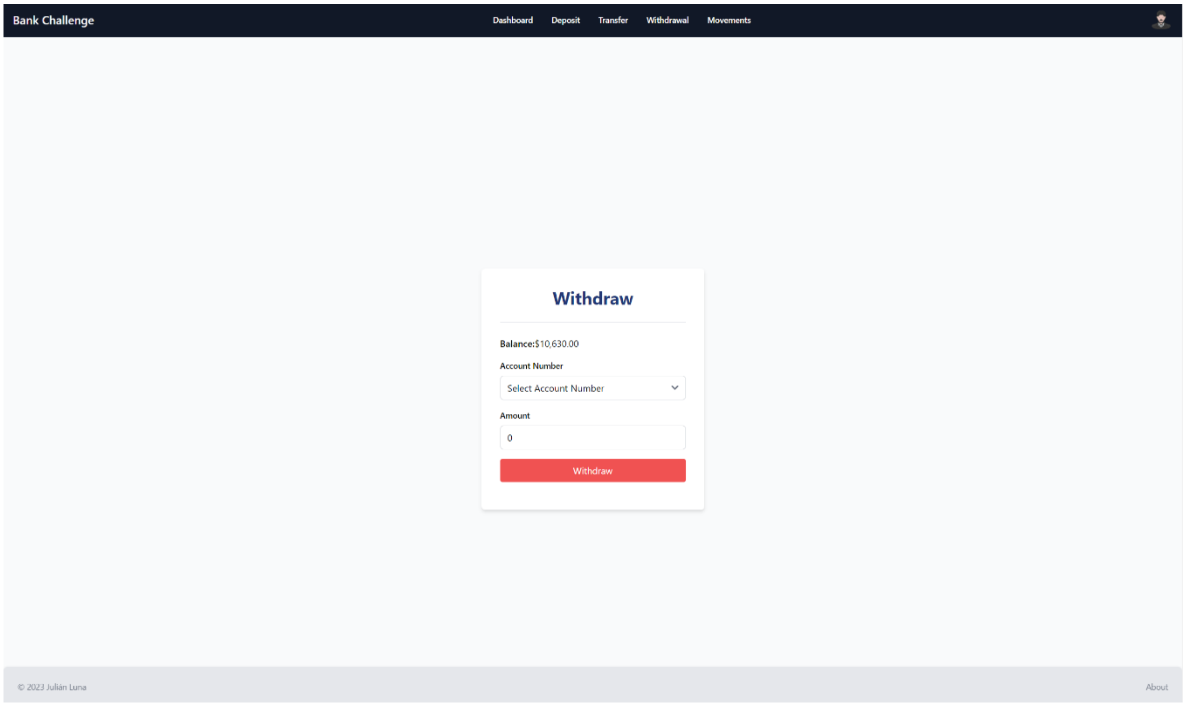
Balance: $10,630.00

Account Number: Select Account Number

Destination Account Number: 0

Amount: 0

Description:



Withdraw

Balance:$10,630.00

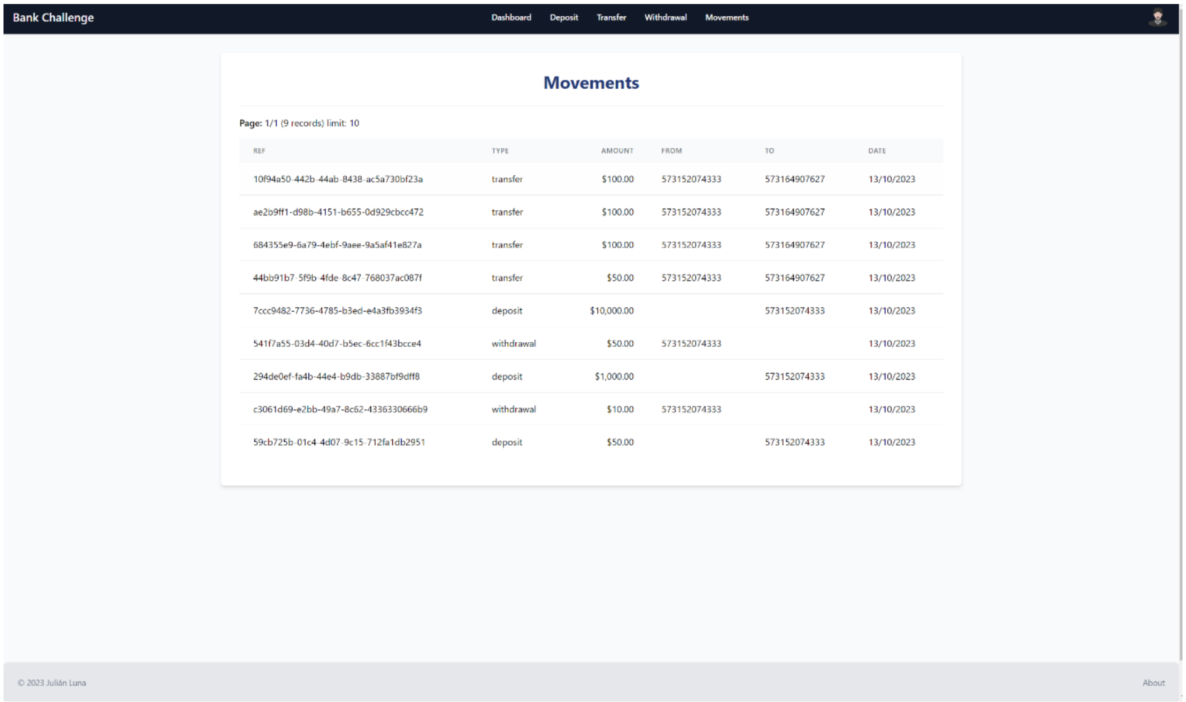
Account Number

Select Account Number

Amount

0

© 2023 Julián Luna

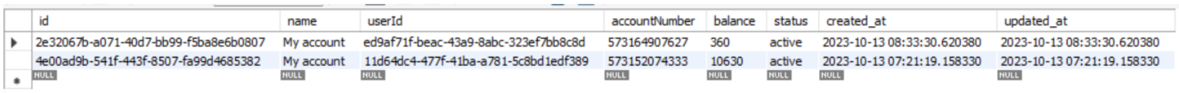


Movements

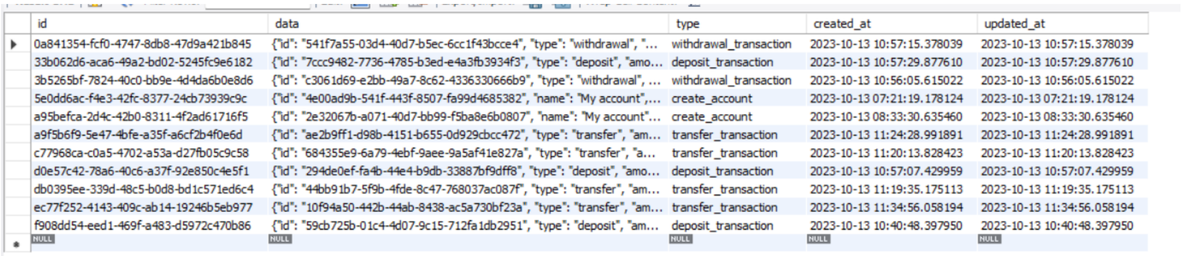
Page: 1/1 (9 records) limit: 10

REF	TYPE	AMOUNT	FROM	TO	DATE
10f94a50-442b-44ab-8438-ac5a730bf23a	transfer	$100.00	573152074333	573164907627	13/10/2023
ae2b9ff1-d98b-4151-b655-0d929cbcc472	transfer	$100.00	573152074333	573164907627	13/10/2023
684355e9-6a79-4ebf-9aee-9a5af41e827a	transfer	$100.00	573152074333	573164907627	13/10/2023
44bb91b7-5f9b-4fde-8c47-768037ac087f	transfer	$50.00	573152074333	573164907627	13/10/2023
7ccc9482-7736-4785-b3ed-e4a3fb3934f3	deposit	$10,000.00		573152074333	13/10/2023
541f7a55-03d4-40d7-b5ec-6cc1f43bcce4	withdrawal	$50.00	573152074333		13/10/2023
294de0ef-fa4b-44e4-b9db-33887bf9dff8	deposit	$1,000.00		573152074333	13/10/2023
c3061d69-e2bb-49a7-8c62-4336330666b9	withdrawal	$10.00	573152074333		13/10/2023
59cb725b-01c4-4d07-9c15-712fa1db2951	deposit	$50.00		573152074333	13/10/2023

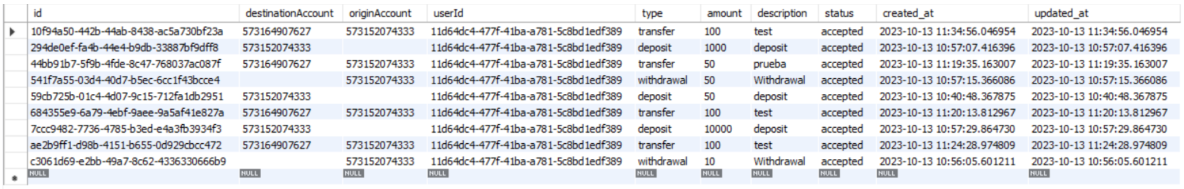
© 2023 Julián Luna



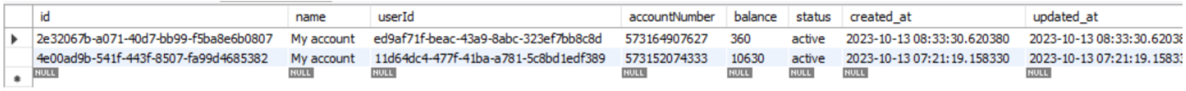
id	name	userId	accountNumber	balance	status	created_at	updated_at
2e32067b-a071-40d7-bb99-f5ba8e6b0807	My account	ed9af71f-beac-43a9-8abc-323ef7bb8c8d	573164907627	360	active	2023-10-13 08:33:30.620380	2023-10-13 08:33:30.620380
4e00ad9b-541f-443f-8507-fa99d4685382	My account	11d64dc4-477f-41ba-a781-5c8bd1edf389	573152074333	10630	active	2023-10-13 07:21:19.158330	2023-10-13 07:21:19.158330
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



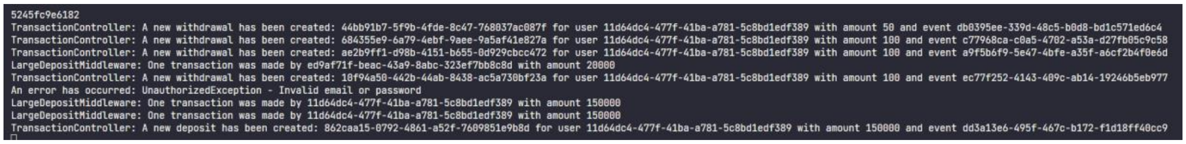
id	data	type	created_at	updated_at
0a841354-fcf0-4747-8db8-47d9a421b845	{"id": "541f7a55-03d4-40d7-b5ec-6cc1f43bcce4", "type": "withdrawal", "...	withdrawal_transaction	2023-10-13 10:57:15.378039	2023-10-13 10:57:15.378039
33b062d6-aca6-49a2-bd02-5245fc9e6182	{"id": "7ccc9482-7736-4785-b3ed-e4a3fb3934f3", "type": "deposit", "amo...	deposit_transaction	2023-10-13 10:57:29.877610	2023-10-13 10:57:29.877610
3b5265bf-7824-40c0-bb9e-4d4da6b0e8d6	{"id": "c3061d69-e2bb-49a7-8c62-4336330666b9", "type": "withdrawal", ...	withdrawal_transaction	2023-10-13 10:56:05.615022	2023-10-13 10:56:05.615022
5e0dd6ac-f4e3-42fc-8377-24cb73939c9c	{"id": "4e00ad9b-541f-443f-8507-fa99d4685382", "name": "My account",...	create_account	2023-10-13 07:21:19.178124	2023-10-13 07:21:19.178124
a95befca-2d4c-42b0-8311-4f2ad61716f5	{"id": "2e32067b-a071-40d7-bb99-f5ba8e6b0807", "name": "My account"...	create_account	2023-10-13 08:33:30.635460	2023-10-13 08:33:30.635460
a9f5b6f9-5e47-4bfe-a35f-a6cf2b4f0e6d	{"id": "ae2b9ff1-d98b-4151-b655-0d929cbcc472", "type": "transfer", "am...	transfer_transaction	2023-10-13 11:24:28.991891	2023-10-13 11:24:28.991891
c77968ca-c0a5-4702-a53a-d27fb05c9c58	{"id": "684355e9-6a79-4ebf-9aee-9a5af41e827a", "type": "transfer", "a...	transfer_transaction	2023-10-13 11:20:13.828423	2023-10-13 11:20:13.828423
d0e57c42-78a6-40c6-a37f-92e850c4e5f1	{"id": "294de0ef-fa4b-44e4-b9db-33887bf9dff8", "type": "deposit", "amo...	deposit_transaction	2023-10-13 10:57:07.429959	2023-10-13 10:57:07.429959
db0395ee-339d-48c5-b0d8-bd1c571ed6c4	{"id": "44bb91b7-5f9b-4fde-8c47-768037ac087f", "type": "transfer", "am...	transfer_transaction	2023-10-13 11:19:35.175113	2023-10-13 11:19:35.175113
ec77f252-4143-409c-ab14-19246b5eb977	{"id": "10f94a50-442b-44ab-8438-ac5a730bf23a", "type": "transfer", "am...	transfer_transaction	2023-10-13 11:34:56.058194	2023-10-13 11:34:56.058194
f908dd54-eed1-469f-a483-d5972c470b86	{"id": "59cb725b-01c4-4d07-9c15-712fa1db2951", "type": "deposit", "am...	deposit_transaction	2023-10-13 10:40:48.397950	2023-10-13 10:40:48.397950
NULL	NULL	NULL	NULL	NULL



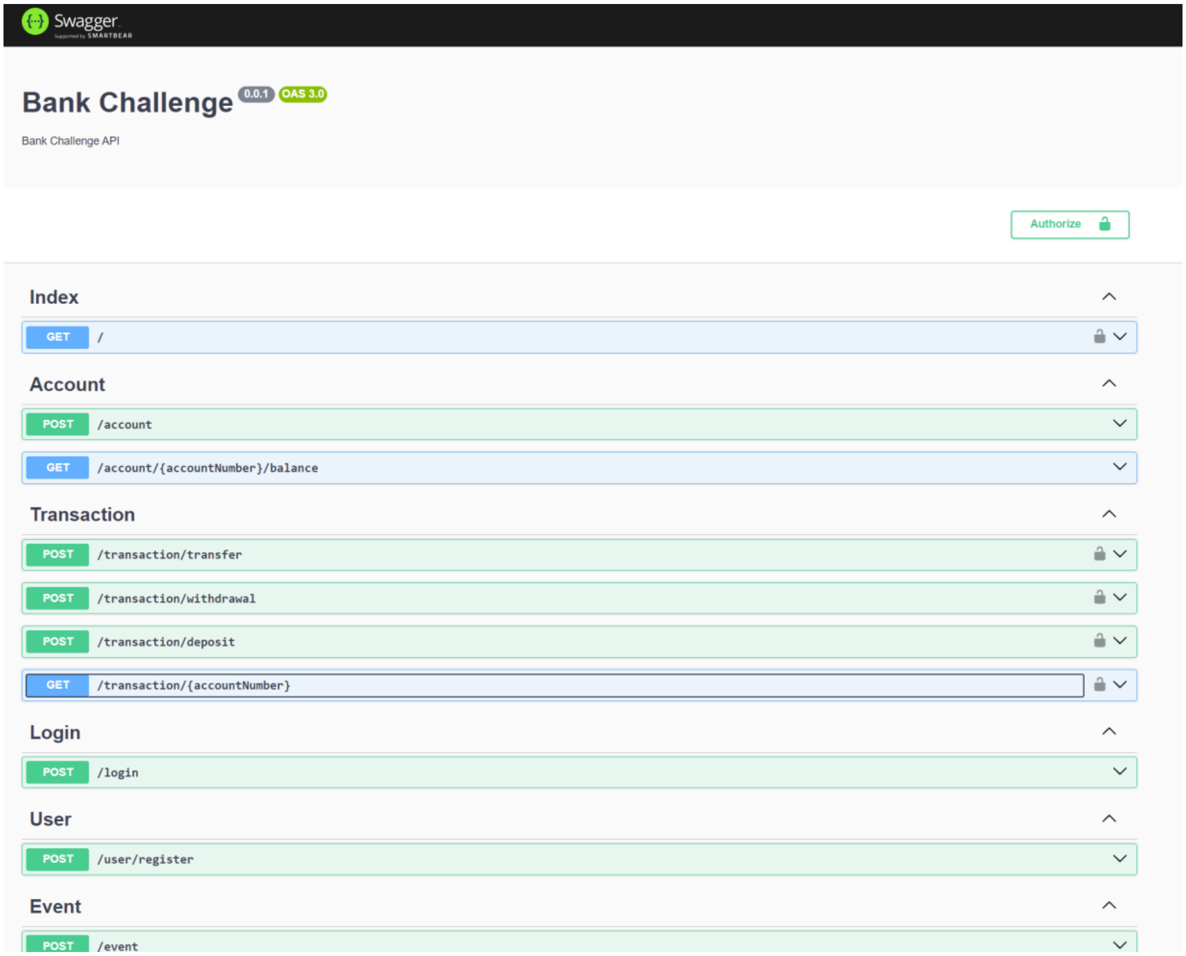
id	destinationAccount	originAccount	userId	type	amount	description	status	created_at	updated_at
10f94a50-442b-44ab-8438-ac5a730bf23a	573164907627	573152074333	11d64dc4-477f-41ba-a781-5c8bd1edf389	transfer	100	test	accepted	2023-10-13 11:34:56.046954	2023-10-13 11:34:56.046954
294de0ef-fa4b-44e4-b9db-33887bf9dff8	573152074333		11d64dc4-477f-41ba-a781-5c8bd1edf389	deposit	1000	deposit	accepted	2023-10-13 10:57:07.416396	2023-10-13 10:57:07.416396
44bb91b7-5f9b-4fde-8c47-768037ac087f	573164907627	573152074333	11d64dc4-477f-41ba-a781-5c8bd1edf389	transfer	50	prueba	accepted	2023-10-13 11:19:35.163007	2023-10-13 11:19:35.163007
541f7a55-03d4-40d7-b5ec-6cc1f43bcce4		573152074333	11d64dc4-477f-41ba-a781-5c8bd1edf389	withdrawal	50	Withdrawal	accepted	2023-10-13 10:57:15.366086	2023-10-13 10:57:15.366086
59cb725b-01c4-4d07-9c15-712fa1db2951	573152074333		11d64dc4-477f-41ba-a781-5c8bd1edf389	deposit	50	deposit	accepted	2023-10-13 10:40:48.367875	2023-10-13 10:40:48.367875
684355e9-6a79-4ebf-9aee-9a5af41e827a	573164907627	573152074333	11d64dc4-477f-41ba-a781-5c8bd1edf389	transfer	100	test	accepted	2023-10-13 11:20:13.812967	2023-10-13 11:20:13.812967
7ccc9482-7736-4785-b3ed-e4a3fb3934f3	573152074333		11d64dc4-477f-41ba-a781-5c8bd1edf389	deposit	10000	deposit	accepted	2023-10-13 10:57:29.864730	2023-10-13 10:57:29.864730
ae2b9ff1-d98b-4151-b655-0d929cbcc472	573164907627	573152074333	11d64dc4-477f-41ba-a781-5c8bd1edf389	transfer	100	test	accepted	2023-10-13 11:24:28.974809	2023-10-13 11:24:28.974809
c3061d69-e2bb-49a7-8c62-4336330666b9		573152074333	11d64dc4-477f-41ba-a781-5c8bd1edf389	withdrawal	10	Withdrawal	accepted	2023-10-13 10:56:05.601211	2023-10-13 10:56:05.601211
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



id	name	userId	accountNumber	balance	status	created_at	updated_at
2e32067b-a071-40d7-bb99-f5ba8e6b0807	My account	ed9af71f-beac-43a9-8abc-323ef7bb8c8d	573164907627	360	active	2023-10-13 08:33:30.620380	2023-10-13 08:33:30.62038
4e00ad9b-541f-443f-8507-fa99d4685382	My account	11d64dc4-477f-41ba-a781-5c8bd1edf389	573152074333	10630	active	2023-10-13 07:21:19.158330	2023-10-13 07:21:19.1583
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



```
5245fc9e6182
TransactionController: A new withdrawal has been created: 44bb91b7-5f9b-4fde-8c47-768037ac087f for user 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 50 and event db0395ee-339d-48c5-b0d8-bd1c571ed6c4
TransactionController: A new withdrawal has been created: 684355e9-6a79-4ebf-9aee-9a5af41e827a for user 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 100 and event c77968ca-c0a5-4702-a53a-d27fb05c9c58
TransactionController: A new withdrawal has been created: ae2b9ff1-d98b-4151-b655-0d929cbcc472 for user 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 100 and event a9f5b6f9-5e47-4bfe-a35f-a6cf2b4f0e6d
LargeDepositMiddleware: One transaction was made by ed9af71f-beac-43a9-8abc-323ef7bb8c8d with amount 20000
TransactionController: A new withdrawal has been created: 10f94a50-442b-44ab-8438-ac5a730bf23a for user 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 100 and event ec77f252-4143-409c-ab14-19246b5eb977
An error has occurred: UnauthorizedException - Invalid email or password
LargeDepositMiddleware: One transaction was made by 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 150000
LargeDepositMiddleware: One transaction was made by 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 150000
TransactionController: A new deposit has been created: 862caa15-0792-4861-a52f-7609851e9b8d for user 11d64dc4-477f-41ba-a781-5c8bd1edf389 with amount 150000 and event dd3a13e6-495f-467c-b172-f1d18ff40cc9
```



Bank Challenge 0.0.1 OAS 3.0

Bank Challenge API

Index

GET /

Account

POST /account

GET /account/{accountNumber}/balance

Transaction

POST /transaction/transfer

POST /transaction/withdrawal

POST /transaction/deposit

GET /transaction/{accountNumber}

Login

POST /login

User

POST /user/register

Event

POST /event