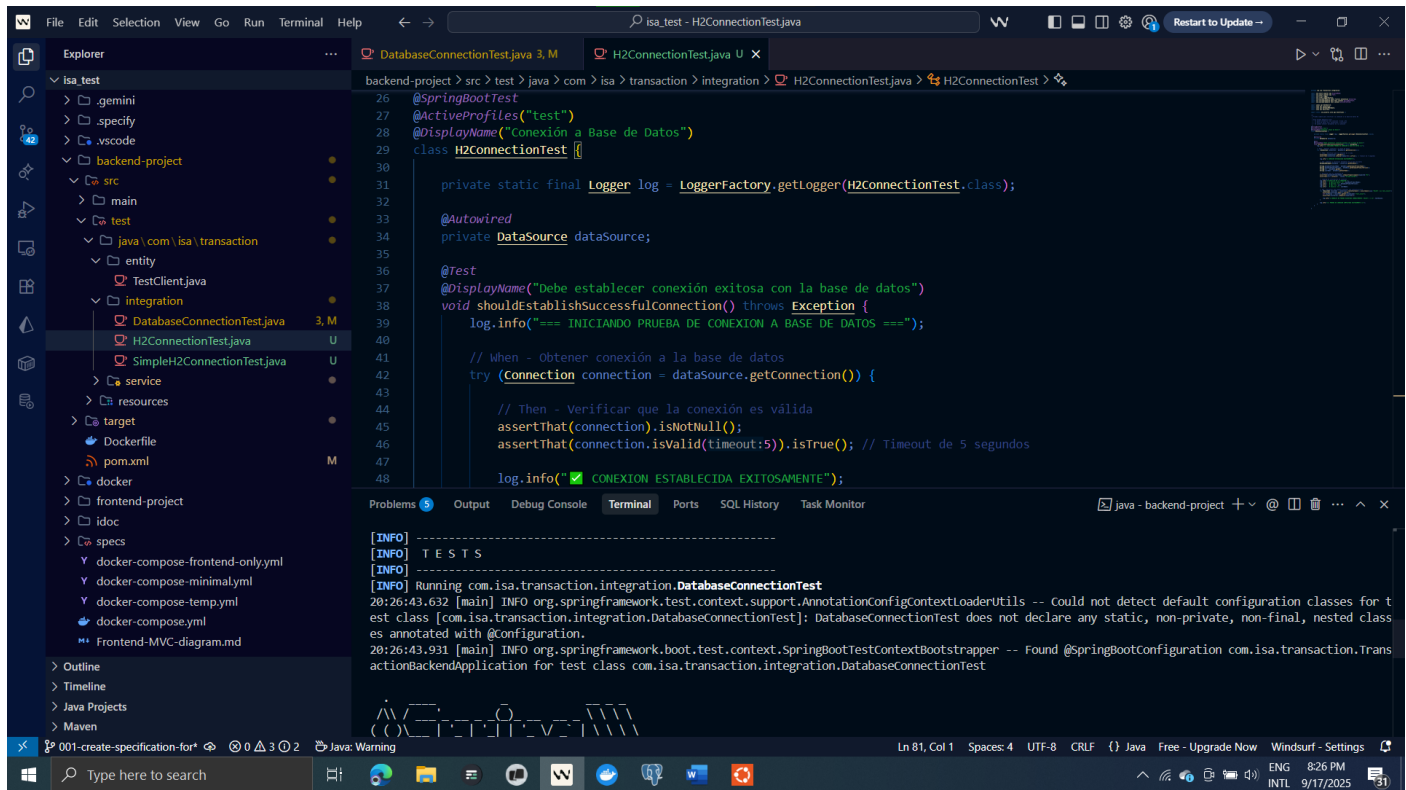


Documentacion de ejecucion de pruebas unitarias

a. Pruebas Backend

- Comprobar la conexión a la base de datos



The screenshot shows the Visual Studio Code editor with the file `H2ConnectionTest.java` open. The file is part of the `backend-project` and is located in the `src > test > java > com > isa > transaction > integration` directory. The code defines a `H2ConnectionTest` class that extends `SpringBootTest` and uses `ActiveProfiles("test")`. It contains a `shouldEstablishSuccessfulConnection()` method that tests the database connection. The terminal output shows the test running successfully, with the message `CONEXION ESTABLECIDA EXITOSAMENTE`.

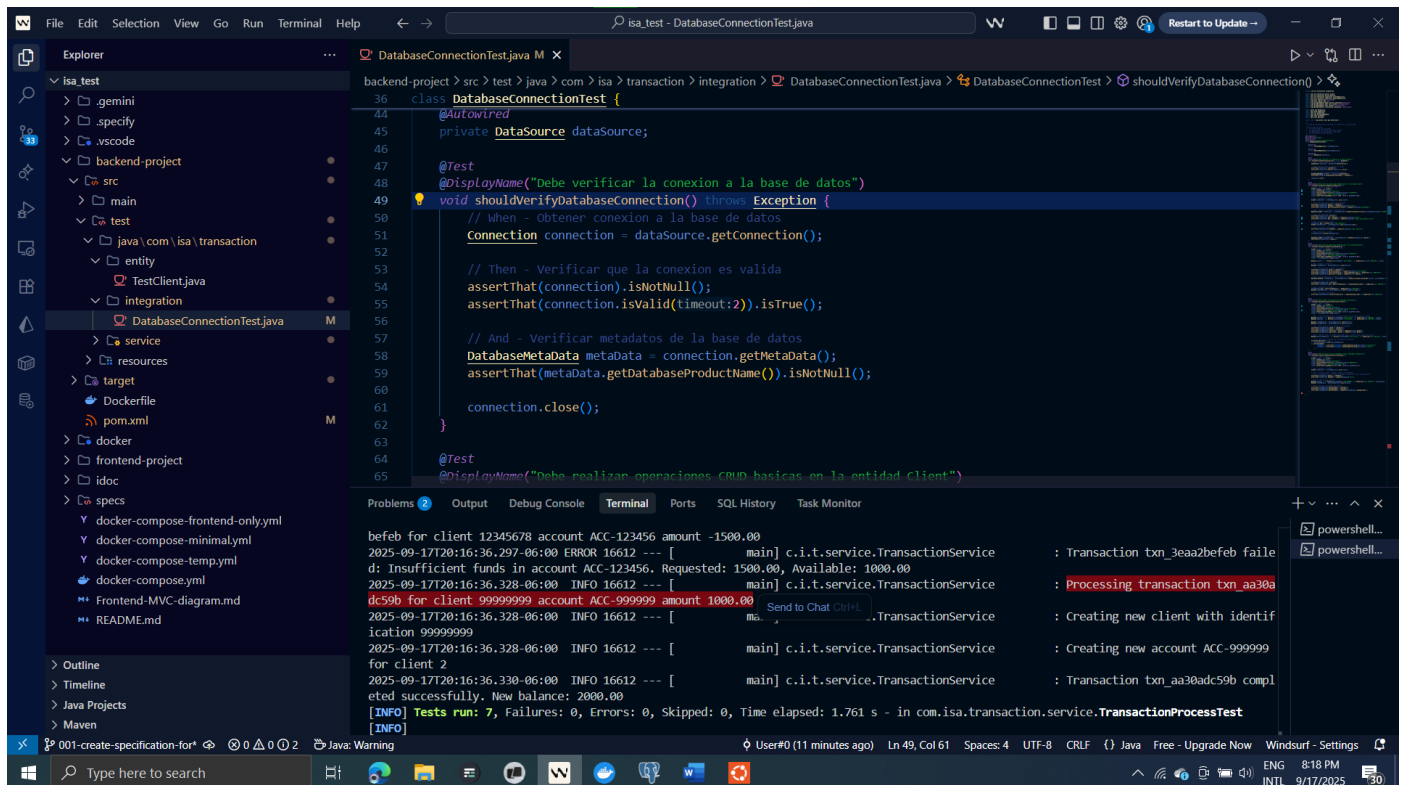
```
26 @SpringBootTest
27 @ActiveProfiles("test")
28 @DisplayName("Conexión a Base de Datos")
29 class H2ConnectionTest {
30
31     private static final Logger log = LoggerFactory.getLogger(H2ConnectionTest.class);
32
33     @Autowired
34     private DataSource dataSource;
35
36     @Test
37     @DisplayName("Debe establecer conexión exitosa con la base de datos")
38     void shouldEstablishSuccessfulConnection() throws Exception {
39         log.info("=== INICIANDO PRUEBA DE CONEXION A BASE DE DATOS ===");
40
41         // When - Obtener conexión a la base de datos
42         try (Connection connection = dataSource.getConnection()) {
43
44             // Then - Verificar que la conexión es válida
45             assertThat(connection).isNotNull();
46             assertThat(connection.isValid(timeout:5)).isTrue(); // Timeout de 5 segundos
47
48             log.info("✅ CONEXION ESTABLECIDA EXITOSAMENTE");
49         }
50     }
51 }
```

Terminal Output:

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.isa.transaction.integration.DatabaseConnectionTest
20:26:43.632 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default configuration classes for test class [com.isa.transaction.integration.DatabaseConnectionTest]: DatabaseConnectionTest does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
20:26:43.931 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfiguration com.isa.transaction.TransactionBackendApplication for test class com.isa.transaction.integration.DatabaseConnectionTest

[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.isa.transaction.integration.DatabaseConnectionTest
20:26:43.931 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfiguration com.isa.transaction.TransactionBackendApplication for test class com.isa.transaction.integration.DatabaseConnectionTest
```

- Validar el proceso de acreditación y débito de transacciones



The screenshot shows the Visual Studio Code editor with the file `DatabaseConnectionTest.java` open. The file is part of the `backend-project` and is located in the `src > test > java > com > isa > transaction > integration` directory. The code defines a `DatabaseConnectionTest` class that extends `SpringBootTest` and uses `ActiveProfiles("test")`. It contains a `shouldVerifyDatabaseConnection()` method that tests the database connection. The terminal output shows the test running successfully, with the message `Tests run: 7, Failures: 0, Errors: 0, Skipped: 0`.

```
36 class DatabaseConnectionTest {
37
38     @Autowired
39     private DataSource dataSource;
40
41     @Test
42     @DisplayName("Debe verificar la conexión a la base de datos")
43     void shouldVerifyDatabaseConnection() throws Exception {
44         // When - Obtener conexión a la base de datos
45         Connection connection = dataSource.getConnection();
46
47         // Then - Verificar que la conexión es válida
48         assertThat(connection).isNotNull();
49         assertThat(connection.isValid(timeout:2)).isTrue();
50
51         // And - Verificar metadatos de la base de datos
52         DatabaseMetaData metaData = connection.getMetaData();
53         assertThat(metaData.getDatabaseProductName()).isNotNull();
54
55         connection.close();
56     }
57
58     @Test
59     @DisplayName("Debe realizar operaciones CRUD basicas en la entidad Client")
60     void shouldPerformCRUDOperations() throws Exception {
61         // ...
62     }
63 }
```

Terminal Output:

```
befeb for client 12345678 account ACC-123456 amount -1500.00
2025-09-17T20:16:36.297-06:00 ERROR 16612 --- [main] c.i.t.service.TransactionService : Transaction txn_3eaa2befeb failed: Insufficient funds in account ACC-123456. Requested: 1500.00, Available: 1000.00
2025-09-17T20:16:36.328-06:00 INFO 16612 --- [main] c.i.t.service.TransactionService : Processing transaction txn_aa308dc59b for client 99999999 account ACC-999999 amount 1000.00
2025-09-17T20:16:36.328-06:00 INFO 16612 --- [main] c.i.t.service.TransactionService : Creating new client with identification 99999999
2025-09-17T20:16:36.328-06:00 INFO 16612 --- [main] c.i.t.service.TransactionService : Creating new account ACC-999999 for client 2
2025-09-17T20:16:36.330-06:00 INFO 16612 --- [main] c.i.t.service.TransactionService : Transaction txn_aa308dc59b completed successfully. New balance: 2000.00
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.761 s - in com.isa.transaction.service.TransactionProcessTest
[INFO]
```

b. Pruebas Frontend JSF

- Comprobar la validación de formularios y binding de datos en Managed Beans

The screenshot shows an IDE with the following components:

- Explorer:** A project structure for 'isa_test' with folders for 'backend-project', 'docker', 'frontend-project', 'src', 'main', 'java', 'resources', 'webapp', 'WEB-INF', 'test', and 'resources'. The file 'FormValidationTest.java' is selected under 'test'.
- Editor:** Displays the code for 'FormValidationTest.java'. The code includes a test class with methods for setting up test data, performing assertions on the transaction bean, and verifying the transaction type.
- Terminal:** Shows the output of the test run, indicating that all tests passed successfully.

```
class FormValidationTest {  
    @Test  
    @DisplayName("Debe validar correctamente los campos del formulario")  
    void shouldValidateFormFields() {  
        // Given - Datos válidos  
        String validClientId = "12345678";  
        String validAccountNumber = "ACC-123456";  
        BigDecimal validAmount = new BigDecimal(val:"100.50");  
  
        // When - Asignar datos al bean (binding de datos)  
        transactionBean.setClientIdentification(validClientId);  
        transactionBean.setAccountNumber(validAccountNumber);  
        transactionBean.setAmount(validAmount);  
  
        // Then - Verificar binding correcto  
        assertThat(transactionBean.getClientIdentification()).isEqualTo(validClientId);  
        assertThat(transactionBean.getAccountNumber()).isEqualTo(validAccountNumber);  
        assertThat(transactionBean.getAmount()).isEqualTo(validAmount);  
  
        // And - Verificar tipo de transacción calculado correctamente  
        assertThat(transactionBean.getTransactionType()).isEqualTo(expected:"DEPOSITO");  
        assertThat(transactionBean.getTransactionTypeText()).isEqualTo(expected:"Crédito (Depósito)");  
    }  
}
```

Terminal Output:

```
INFO: Campo monto limpiado  
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.036 s - in com.isa.transaction.frontend.FormValidationTest  
[INFO] Results:  
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0  
[INFO] --- war:3.3.1:war (default-war) @ frontend-project ---  
[INFO] Packaging webapp  
[INFO] Assembling webapp [frontend-project] in [C:\Coding\Me\isa_test\frontend-project\target\transaction-frontend]  
[INFO] Processing war project  
[INFO] Copying webapp resources [C:\Coding\Me\isa_test\frontend-project\src\main\webapp]
```

- Verificar la correcta inyección de dependencias entre Managed Beans y servicios

The screenshot shows an IDE with the following components:

- Explorer:** A project structure for 'isa_test' with folders for 'backend-project', 'docker', 'frontend-project', 'src', 'main', 'java', 'resources', 'webapp', 'WEB-INF', 'test', and 'resources'. The file 'DependencyInjectionTest.java' is selected under 'test'.
- Editor:** Displays the code for 'DependencyInjectionTest.java'. The code includes a test class with methods for setting up test data, performing assertions on the transaction bean, and verifying the transaction type.
- Terminal:** Shows the output of the test run, indicating that all tests passed successfully.

```
class DependencyInjectionTest {  
    @Test  
    @DisplayName("Inyección de Dependencias")  
    void shouldCreateTransactionBeanCorrectly() {  
        // Then- Verificar creación del Managed Bean  
        assertThat(transactionBean).isNotNull();  
        assertThat(transactionBean.getClass().getName()).contains(...values:"TransactionBean");  
    }  
}
```

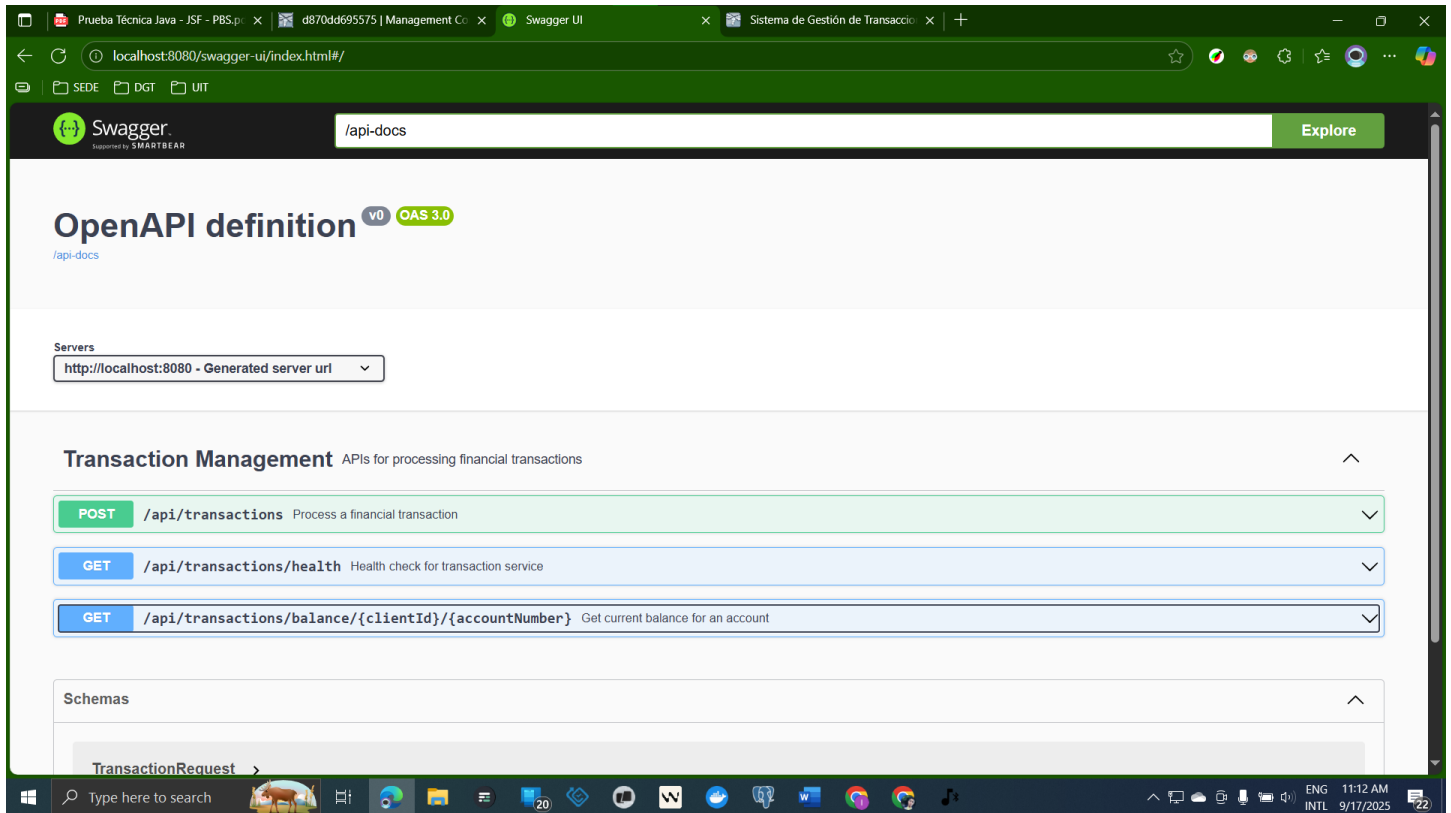
Terminal Output:

```
INFO: Campo monto limpiado  
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.036 s - in com.isa.transaction.frontend.FormValidationTest  
[INFO] Results:  
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0  
[INFO] --- war:3.3.1:war (default-war) @ frontend-project ---  
[INFO] Packaging webapp  
[INFO] Assembling webapp [frontend-project] in [C:\Coding\Me\isa_test\frontend-project\target\transaction-frontend]  
[INFO] Processing war project  
[INFO] Copying webapp resources [C:\Coding\Me\isa_test\frontend-project\src\main\webapp]
```

Documentacion de ejecución de funcionamiento

Bakend testing

Swagger Apis



Api transactions execution

The image shows the Swagger UI for the `/api/transactions` endpoint. The endpoint is a **POST** request that processes a financial transaction. The description states: "Processes a credit or debit transaction for a client account. If the account doesn't exist, it will be created automatically for the client. Returns immediately with transaction ID and processes asynchronously." There are no parameters for this endpoint. The request body is required and set to `application/json`. The example request body is a JSON object:

```
{  "credit": true,  "debit": false,  "clientIdentification": "060543377",  "accountNumber": "ACC-123456",  "amount": 211}
```

 At the bottom, there is an **Execute** button and a **Clear** button.

Database data

The image shows the pgAdmin 4 interface. The left sidebar displays the **Object Explorer** for the `testhtc` database. The **Tables (4)** folder is expanded, showing `balance`, `balance_transaction`, `client`, and `Columns`. The `Columns` table is selected. The main pane shows the **Query** editor with the following SQL query:

```
select * from testhtc.client;
select * from testhtc.balance;
select * from testhtc.balance_transaction;
```

 The **Data Output** pane shows the results of the query, displaying a single row of data. The status bar at the bottom indicates "Total rows: 1" and "Query complete 00:00:00.134".

	current_balance numeric (15,2)	client_id bigint	created_at timestamp with time zone	id [PK] bigint	updated_at timestamp with time zone	account_number character varying (10)
1	211.00	1	2025-09-17 17:07:57.855643+		2025-09-17 17:10:03.737599+	ACC-123456

Insufficient funds

The image shows the Swagger UI for the `/api/transactions` endpoint. The endpoint is a **POST** method. The description states: "Processes a credit or debit transaction for a client account. If the account doesn't exist, it will be created automatically for the client. Returns immediately with transaction ID and processes asynchronously." There are no parameters. The request body is required and set to `application/json`. The body contains the following JSON:

```
{  "credit": false,  "debit": true,  "clientIdentification": "060543377",  "accountNumber": "ACC-123456",  "amount": -500}
```

At the bottom, there is an **Execute** button and a **Clear** button.

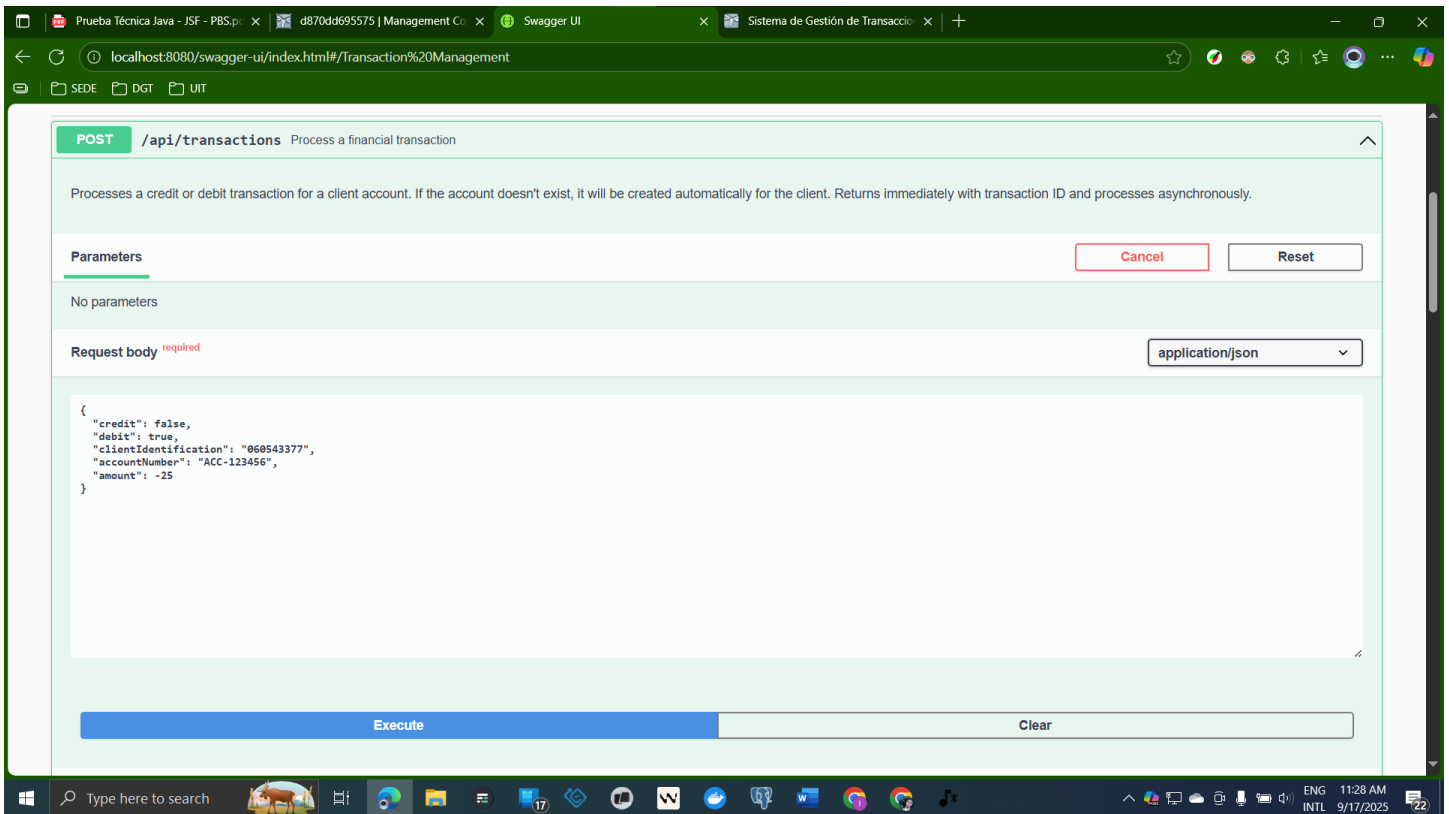
The image shows the Swagger UI for the `/api/transactions` endpoint, displaying the response for a failed transaction. The response status is **409** (Conflict). The response body is:

```
{  "success": false,  "error": true,  "data": null,  "message": "Insufficient funds in account ACC-123456. Requested: 500, Available: 211.00",  "code": "INSUFFICIENT_FUNDS"}
```

The response headers are:

```
connection: keep-alive
content-type: application/json
date: Wed, 17 Sep 2025 17:18:46 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Substraction founds



POST /api/transactions Process a financial transaction

Processes a credit or debit transaction for a client account. If the account doesn't exist, it will be created automatically for the client. Returns immediately with transaction ID and processes asynchronously.

Parameters

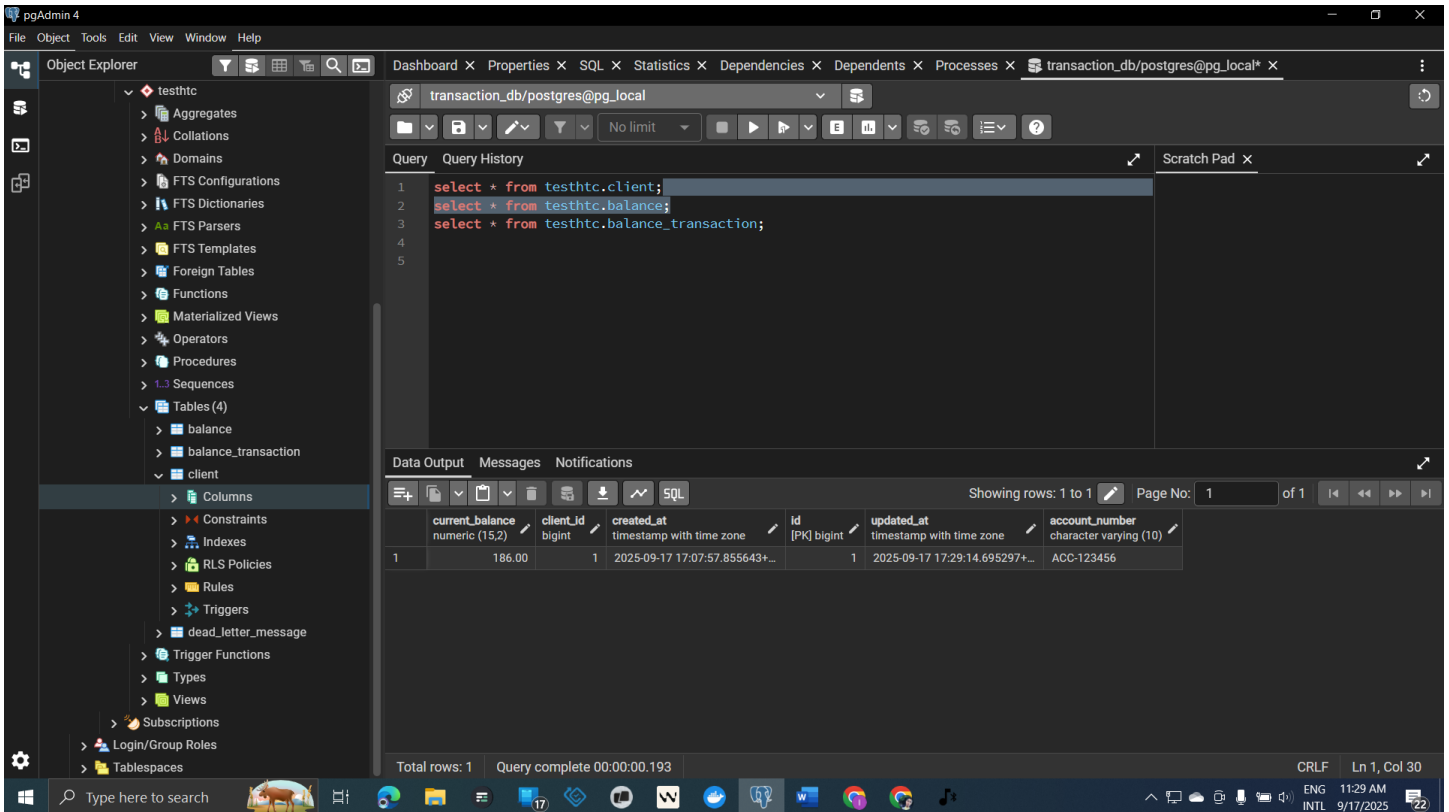
No parameters

Request body *required*

application/json

```
{
  "credit": false,
  "debit": true,
  "clientId": "060543377",
  "accountNumber": "ACC-123456",
  "amount": -25
}
```

Execute **Clear**



pgAdmin 4

File Object Tools Edit View Window Help

Dashboard Properties SQL Statistics Dependencies Dependents Processes transaction_db/postgres@pg_local

transaction_db/postgres@pg_local

Query History

```
1 select * from testhtc.client;
2 select * from testhtc.balance;
3 select * from testhtc.balance_transaction;
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1 of 1

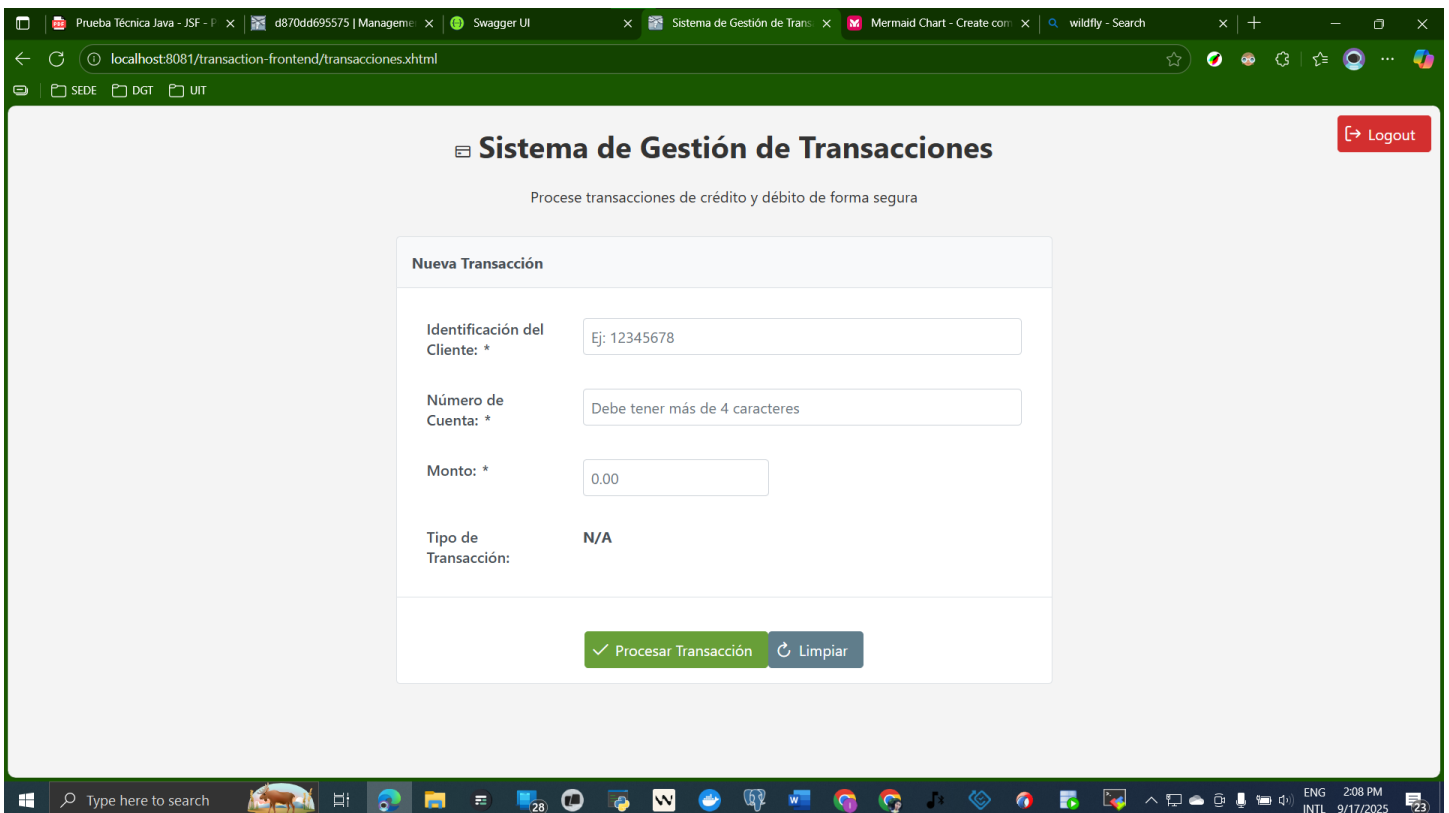
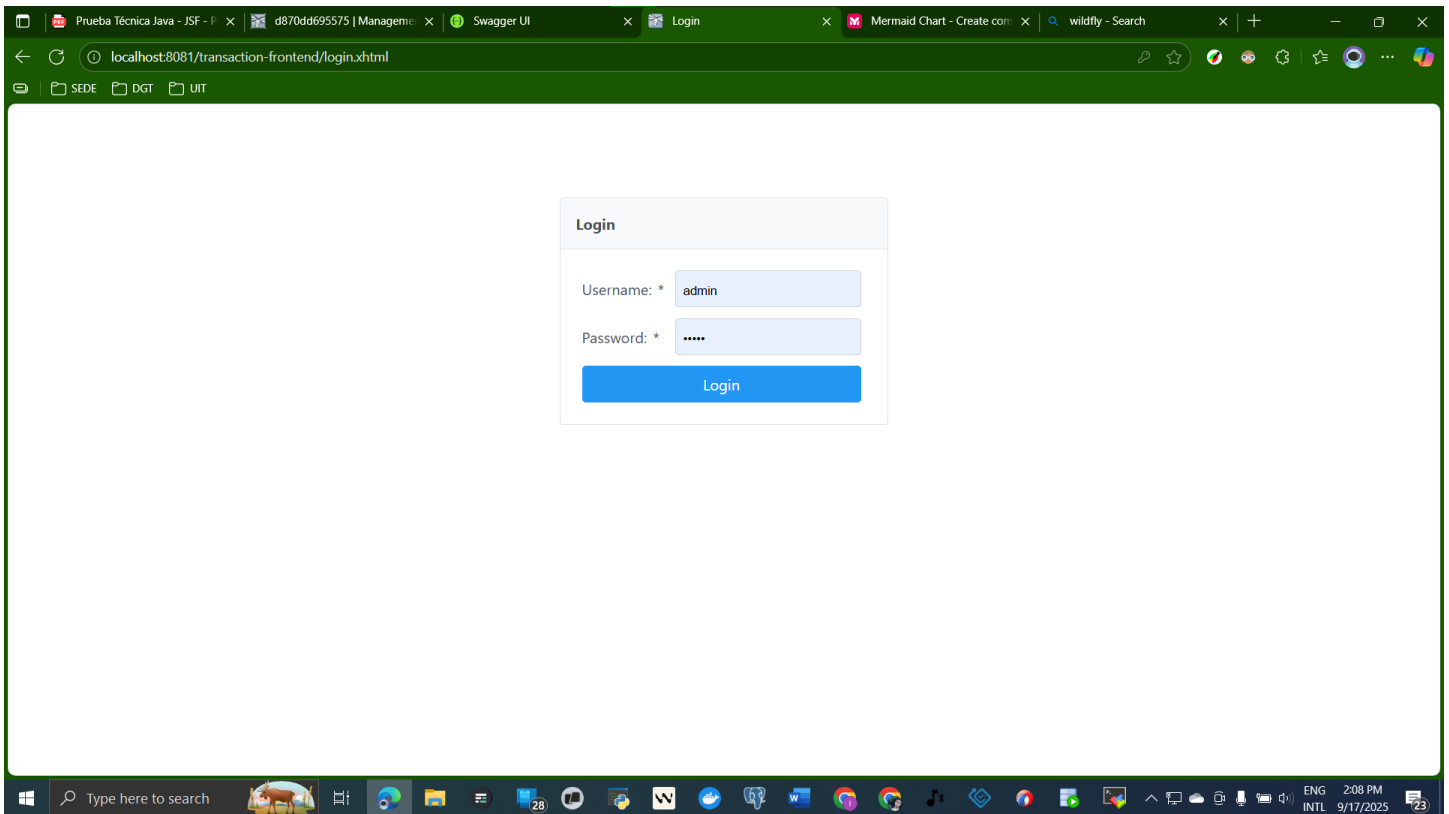
current_balance	client_id	created_at	id	updated_at	account_number
numeric (15,2)	bigint	timestamp with time zone	[PK] bigint	timestamp with time zone	character varying (10)
186.00	1	2025-09-17 17:07:57.855643+...	1	2025-09-17 17:29:14.695297+...	ACC-123456

Total rows: 1 Query complete 00:00:00.193

CRLF Ln 1, Col 30

Fontend implementation

Fake login for declared navigation admin/admin



Logging System

Frontend

The screenshot shows the Docker Desktop interface with the 'transaction-frontend' container selected. The container is running and has a status of 'Running (5 hours ago)'. The logs show a series of transactions being submitted and validated. The logs are as follows:

```
20:00:36,414 INFO [com.isa.transaction.frontend.bean.TransactionBean] (default task-31) Submitting transaction for client 060543377, account AAC-060543, amount -5000.00
20:00:36,415 INFO [com.isa.transaction.frontend.service.TransactionRestClient] (default task-31) Submitting transaction: TransactionRequest{clientIdentification='060543377', accountNumber='AAC-060543', amount=-5000.00}
20:00:41,458 WARNING [com.isa.transaction.frontend.service.TransactionRestClient] (default task-31) Transaction validation error: Insufficient funds in account AAC-060543. Requested: 5000.00, Available: 200.00
20:00:41,458 WARNING [com.isa.transaction.frontend.bean.TransactionBean] (default task-31) Transaction failed: Insufficient funds in account AAC-060543. Requested: 5000.00, Available: 200.00
20:00:55,233 INFO [com.isa.transaction.frontend.bean.TransactionBean] (default task-31) Submitting transaction for client 060543377, account AAC-060543, amount 50.00
20:00:55,234 INFO [com.isa.transaction.frontend.service.TransactionRestClient] (default task-31) Submitting transaction: TransactionRequest{clientIdentification='060543377', accountNumber='AAC-060543', amount=50.00}
20:01:00,285 INFO [com.isa.transaction.frontend.service.TransactionRestClient] (default task-31) Transaction submitted successfully: ApiResponse{data=TransactionResponse{transactionId='txn_b85845266f', status='ACCEPTED'}, message='Transaction has been accepted and is being processed', code='ACCEPTED'}
20:01:00,285 INFO [com.isa.transaction.frontend.bean.TransactionBean] (default task-31) Transaction successful: txn_b85845266f
20:01:14,328 INFO [com.isa.transaction.frontend.bean.TransactionBean] (default task-31) Submitting transaction for client 060543377, account AAC-060543, amount -9000.00
20:01:14,329 INFO [com.isa.transaction.frontend.service.TransactionRestClient] (default task-31) Submitting transaction: TransactionRequest{clientIdentification='060543377', accountNumber='AAC-060543', amount=-9000.00}
20:01:19,368 WARNING [com.isa.transaction.frontend.service.TransactionRestClient] (default task-31) Transaction validation error: Insufficient funds in account AAC-060543. Requested: 9000.00, Available: 250.00
20:01:19,368 WARNING [com.isa.transaction.frontend.bean.TransactionBean] (default task-31) Transaction failed: Insufficient funds in account AAC-060543. Requested: 9000.00, Available: 250.00
```

Backend

The screenshot shows the Docker Desktop interface with the 'transaction-backend' container selected. The container is running and has a status of 'Running (4 hours ago)'. The logs show a series of transactions being submitted and validated. The logs are as follows:

```
ssl.secure.random.implementation = null
ssl.trustmanager.algorithm = PKIX
ssl.truststore.certificates = null
ssl.truststore.location = null
ssl.truststore.password = null
ssl.truststore.type = JKS
transaction.timeout.ms = 60000
transactional.id = null
value.serializer = class org.springframework.kafka.support.serializer.JsonSerializer

2025-09-17T20:01:14.353Z INFO 1 --- [ntio-8080-exec-1] o.a.k.clients.producer.KafkaProducer : [Producer clientId=producer-9] Instantiated an idempotent producer.
2025-09-17T20:01:19.359Z WARN 1 --- [ntio-8080-exec-1] org.apache.kafka.clients.ClientUtils : Couldn't resolve server kafka:29092 from bootstrap.servers as
2025-09-17T20:01:19.359Z INFO 1 --- [ntio-8080-exec-1] o.a.k.clients.producer.KafkaProducer : [Producer clientId=producer-9] Closing the Kafka producer with timeoutMillis = 0 ms.
2025-09-17T20:01:19.360Z INFO 1 --- [ntio-8080-exec-1] o.apache.kafka.common.metrics.Metrics : Metrics scheduler closed
2025-09-17T20:01:19.360Z INFO 1 --- [ntio-8080-exec-1] o.apache.kafka.common.metrics.Metrics : Closing reporter org.apache.kafka.common.metrics.JmxReporter
2025-09-17T20:01:19.360Z INFO 1 --- [ntio-8080-exec-1] o.apache.kafka.common.metrics.Metrics : Metrics reporters closed
2025-09-17T20:01:19.360Z INFO 1 --- [ntio-8080-exec-1] o.a.kafka.common.utils.AppInfoParser : App info kafka.producer for producer-9 unregistered
2025-09-17T20:01:19.361Z ERROR 1 --- [ntio-8080-exec-1] c.i.t.service.TransactionService : Failed to send failure event for transaction txn_292ba177c3:
2025-09-17T20:01:19.362Z ERROR 1 --- [ntio-8080-exec-1] c.i.t.controller.GlobalExceptionHandler : Transaction exception: Insufficient funds in account AAC-060543. Requested: 9000.00, Available: 250.00 [INSUFFICIENT_FUNDS]
```