

Manuel de Usuario
del Aplicativo
BP-ArquitecturaMicroservicio-SP
SolangePico
Versión
1.0.0

INFORMACION DEL DOCUMENTO	
Título	Manuel de Usuario del Aplicativo BP-ArquitecturaMicroservicio-SP
Versión:	1.0.0
Autor:	Solange Pico
Estado:	Release

El propósito del siguiente manual de usuario es ejecutar la aplicación, la misma ha sido realizada en las siguientes versiones:

BACKEND	
Lenguaje de programación:	Java
IDE de desarrollo:	Spring Tool Suite 4-4-17-1
Framework:	Spring Boot 2.6.4 Maven
JDK:	1.8.0_111
Base de datos:	MySQL 8.0.23
UI Base de datos:	Workbench 8.0
Consumo Microservicio:	Postman

A continuación, correr el siguiente script para la creación de la base de datos:

```
CREATE DATABASE sparquitecturamicroserviciobp;
USE sparquitecturamicroserviciobp;

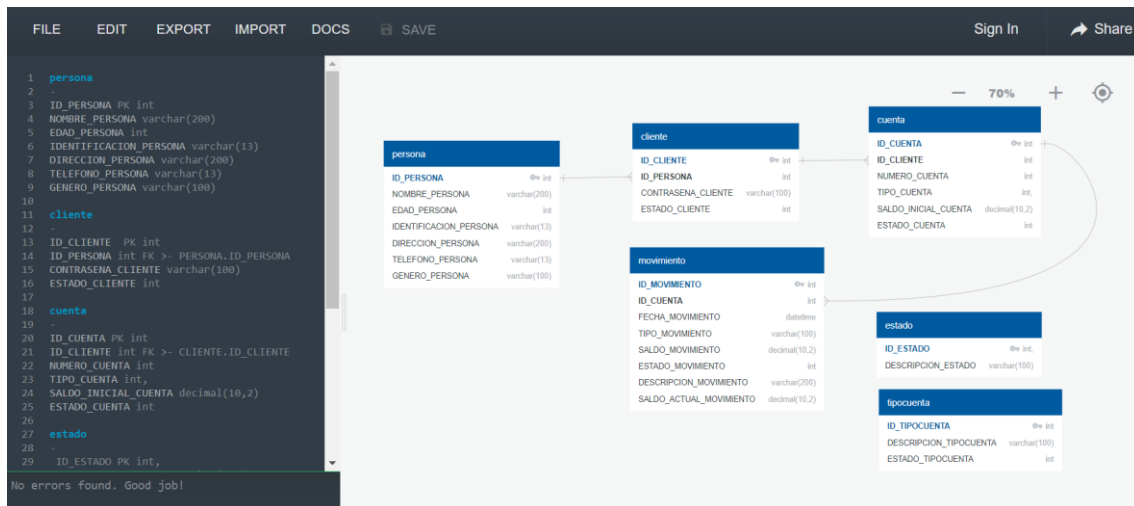
CREATE TABLE `persona` (
  `ID_PERSONA` int NOT NULL AUTO_INCREMENT,
  `NOMBRE_PERSONA` varchar(200) DEFAULT NULL,
  `EDAD_PERSONA` int DEFAULT NULL,
  `IDENTIFICACION_PERSONA` varchar(13) DEFAULT NULL,
  `DIRECCION_PERSONA` varchar(200) DEFAULT NULL,
  `TELEFONO_PERSONA` varchar(13) DEFAULT NULL,
  `GENERO_PERSONA` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`ID_PERSONA`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `cliente` (
  `ID_CLIENTE` int NOT NULL AUTO_INCREMENT,
  `ID_PERSONA` int DEFAULT NULL,
  `CONTRASENA_CLIENTE` varchar(100) DEFAULT NULL,
  `ESTADO_CLIENTE` int DEFAULT NULL,
  PRIMARY KEY (`ID_CLIENTE`),
  KEY `FK_RELATIONSHIP_3` (`ID_PERSONA`),
  CONSTRAINT `FK_RELATIONSHIP_3` FOREIGN KEY (`ID_PERSONA`) REFERENCES `persona`
  (`ID_PERSONA`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `cuenta` (
  `ID_CUENTA` int NOT NULL AUTO_INCREMENT,
  `ID_CLIENTE` int DEFAULT NULL,
  `NUMERO_CUENTA` int DEFAULT NULL,
  `TIPO_CUENTA` int DEFAULT NULL,
  `SALDO_INICIAL_CUENTA` float(10,2) DEFAULT NULL,
  `ESTADO_CUENTA` int DEFAULT NULL,
  PRIMARY KEY (`ID_CUENTA`),
  KEY `FK_RELATIONSHIP_2` (`ID_CLIENTE`),
  CONSTRAINT `FK_RELATIONSHIP_2` FOREIGN KEY (`ID_CLIENTE`) REFERENCES `cliente`
  (`ID_CLIENTE`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `movimiento` (  
  `ID_MOVIMIENTO` int NOT NULL AUTO_INCREMENT,  
  `ID_CUENTA` int DEFAULT NULL,  
  `FECHA_MOVIMIENTO` datetime DEFAULT NULL,  
  `TIPO_MOVIMIENTO` varchar(100) DEFAULT NULL,  
  `SALDO_MOVIMIENTO` float(10,2) DEFAULT NULL,  
  `ESTADO_MOVIMIENTO` int DEFAULT NULL,  
  `DESCRIPCION_MOVIMIENTO` varchar(200) DEFAULT NULL,  
  `SALDO_ACTUAL_MOVIMIENTO` float(10,2) DEFAULT NULL,  
  PRIMARY KEY (`ID_MOVIMIENTO`),  
  KEY `FK_REFERENCE_3` (`ID_CUENTA`),  
  CONSTRAINT `FK_REFERENCE_3` FOREIGN KEY (`ID_CUENTA`) REFERENCES `cuenta`  
  (`ID_CUENTA`))  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
  
CREATE TABLE `estado` (  
  `ID_ESTADO` int NOT NULL AUTO_INCREMENT,  
  `DESCRIPCION_ESTADO` varchar(100) DEFAULT NULL,  
  PRIMARY KEY (`ID_ESTADO`))  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
  
CREATE TABLE `tipocuenta` (  
  `ID_TIPOCUENTA` int NOT NULL AUTO_INCREMENT,  
  `DESCRIPCION_TIPOCUENTA` varchar(100) DEFAULT NULL,  
  `ESTADO_TIPOCUENTA` int DEFAULT NULL,  
  PRIMARY KEY (`ID_TIPOCUENTA`))  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
  
INSERT INTO `tipocuenta` VALUES (1,'Ahorro',1),(2,'Corriente',1);  
INSERT INTO `estado` VALUES (1,'Activo'),(2,'Inactivo');
```

Modelo Entidad – Relación:



Una vez creada la base de datos debe ser una estructura así:



BackEnd:

Para acceder al aplicativo ingresa al siguiente repositorio:

<https://github.com/SolangePico95/BP-ArquitecturaMicroservicio-SP.git>

Se lo descarga en su entorno de desarrollo en mi caso Spring Tool Suite (última versión), abrimos el proyecto desde la carpeta donde lo descargamos, esperamos a que descargue las dependencias y corremos el programa.

Para verificar que la conexión este en lo correcto nos vamos al archivo:

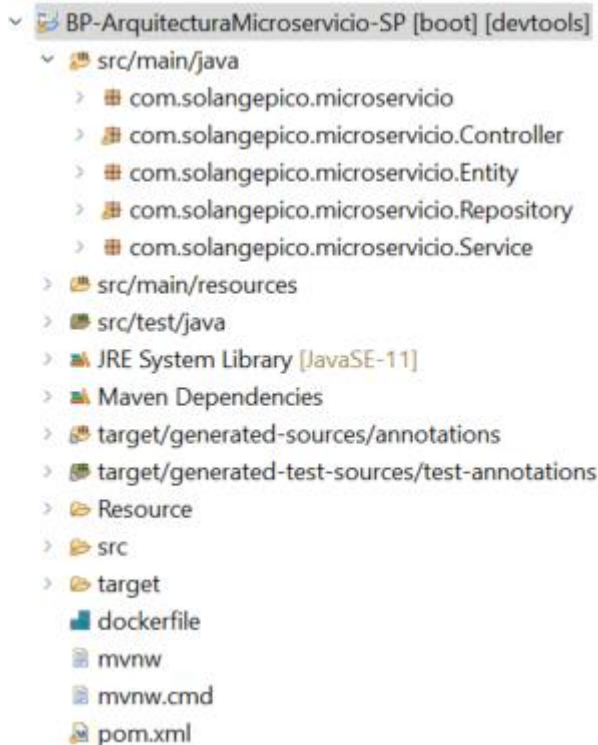
application.properties

y vemos la conexión, caso contrario modificar para que se conecte a nuestra base:

```
1 #::: NOMBRE DEL MICROSERVICIO ::::
2 spring.application.name=BP-ArquitecturaMicroservicio-Sp
3
4 #::: PUERTO ::::
5 server.port=8081
6
7 #::: DATASOURCE MYSQL ::::
8 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
9 spring.datasource.url=jdbc:mysql://localhost:3306/sparquitecturamicroserviciobp?serverTimezone=America/Mexico_City
10 spring.datasource.username=root
11 spring.datasource.password=1234
12
13 spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect
14 logging.level.org.hibernate.SQL=debug
15 spring.jpa.open-in-view=false
16 spring.jpa.show-sql = true
17
```

En mi caso el puerto :8080 ya lo tenía ocupado por el cual use el :8081, aconsejo usar este para ejecutar las validaciones en Postman.

Estructura del proyecto:



Funcionamiento:

Inicio de la aplicación:

La aplicación se desplegará en el puerto 8081

PERSONA:

Crear Persona:

The screenshot shows a REST client interface with a POST request to `http://localhost:8081/api/persona/crearPersona`. The request body is a JSON object representing a person. Below the request, the response is shown as `1 Persona creada.`

Below the REST client, a SQL query is shown: `5 • SELECT * FROM sparquitecturamicroserviciobp.persona;`

The result of the SQL query is displayed in a table with 7 columns: `ID_PERSONA`, `NOMBRE_PERSONA`, `EDAD_PERSONA`, `IDENTIFICACION_PERSONA`, `DIRECCION_PERSONA`, `TELEFONO_PERSONA`, and `GENERO_PERSONA`. The table contains two rows of data.

ID_PERSONA	NOMBRE_PERSONA	EDAD_PERSONA	IDENTIFICACION_PERSONA	DIRECCION_PERSONA	TELEFONO_PERSONA	GENERO_PERSONA
1	John Lennon	40	1707122980	Liverpool	0987142849	Masculino
2	Paul McCartney	40	1726244039	NewYork	0987654323	Masculino

Listar Persona:

The screenshot shows a REST client interface with a GET request to `http://localhost:8081/api/persona/listarPersonas`. The response is shown as a JSON array of two person objects.

The response body is a JSON array of two person objects, each with the following fields: `id_persona`, `nombre_persona`, `edad_persona`, `identificacion_persona`, `direccion_persona`, `telefono_persona`, and `genero_persona`.

```
1 {
2   "id_persona": 1,
3   "nombre_persona": "John Lennon",
4   "edad_persona": 40,
5   "identificacion_persona": "1707122980",
6   "direccion_persona": "Liverpool",
7   "telefono_persona": "0987142849",
8   "genero_persona": "Masculino"
9 },
10 {
11   "id_persona": 2,
12   "nombre_persona": "Paul McCartney",
13   "edad_persona": 40,
14   "identificacion_persona": "1726244039",
15   "direccion_persona": "NewYork",
16   "telefono_persona": "0987654323",
17   "genero_persona": "Masculino"
18 }
19 ]
20
```

Editar Persona:

Pruebas Microservicios Solange Pico / Persona / EditarPersona

PUT

http://localhost:8081/api/persona/editarPersona/1707122980

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

-{

2

...."nombre_persona": "John Lennon Ono",

3

...."edad_persona": 42,

4

...."identificacion_persona": "1707122980",

5

...."direccion_persona": "New York",

6

...."telefono_persona": "0999999999",

7

...."genero_persona": "Masculino"

8

}

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

Persona actualizado.

{

"id_persona": 1,

"nombre_persona": "John Lennon Ono",

"edad_persona": 42,

"identificacion_persona": "1707122980",

"direccion_persona": "New York",

"telefono_persona": "0999999999",

"genero_persona": "Masculino"

}

Eliminar Persona:

Pruebas Microservicios Solange Pico / Persona / EliminarPersona

DELETE

http://localhost:8081/api/persona/eliminarPersona/1726244039

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
--	-----	-------

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

Persona eliminada.

5

SELECT * FROM sparquitecturamicroserviciobp.persona;

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	ID_PERSONA	NOMBRE_PERSONA	EDAD_PERSONA	IDENTIFICACION_PERSONA	DIRECCION_PERSONA	TELEFONO_PERSONA	GENERO_PERSONA
1	1	John Lennon Ono	42	1707122980	New York	0999999999	Masculino
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

CLIENTE:

Crear Cliente:

Pruebas Microservicios Solange Pico / Cliente / CrearCliente

POST

http://localhost:8081/api/cliente/crearCliente

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   ...."contrasena_cliente": "1234",
3   ...."estado_cliente": 1,
4   ...."persona":
5   ....{
6   ....  ...."nombre_persona": "Paul McCartney",
7   ....  ...."edad_persona": 40,
8   ....  ...."identificacion_persona": "1726244039",
9   ....  ...."genero_persona": "Masculino",
10  ....  ...."telefono_persona": "0999999999",
11  ....  ...."direccion_persona": "Irlanda"
12  ....}
13 }
```

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

Cliente creado.

6 • SELECT * FROM sparquitecturamicroserviciobp.cliente;

Result Grid

Filter Rows:

Edit:

Export/Import:

ID_CLIENTE	ID_PERSONA	CONTRASENA_CLIENTE	ESTADO_CLIENTE
1	3	1234	1
2	1	1234	1
NULL	NULL	NULL	NULL


Listar Cliente:

Pruebas Microservicios Solange Pico / Cliente / ListarClientes

GET ▼ http://localhost:8081/api/cliente/listarClientes

Params Authorization Headers (6) Body Pre-request Script Tests

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```
2  {
3      "id_cliente": 1,
4      "contrasena_cliente": "1234",
5      "estado_cliente": 1,
6      "id_persona": 3,
7      "persona": {
8          "id_persona": 3,
9          "nombre_persona": "Paul McCartney",
10         "edad_persona": 40,
11         "identificacion_persona": "1726244039",
12         "direccion_persona": "NewYork",
13         "telefono_persona": "0987654323",
14         "genero_persona": "Masculino"
15     }
16 },
17 {
18     "id_cliente": 2,
19     "contrasena_cliente": "1234",
20     "estado_cliente": 1,
21     "id_persona": 1,
22     "persona": {
23         "id_persona": 1,
24         "nombre_persona": "John Lennon Ono",
25         "edad_persona": 42,
26         "identificacion_persona": "1707122980",
27         "direccion_persona": "New York",
28         "telefono_persona": "0999999999",
29         "genero_persona": "Masculino"
30     }
31 }
```


Editar Cliente:

Pruebas Microservicios Solange Pico / Cliente / **EditarCliente**

PUT ▼ http://localhost:8081/api/cliente/editarCliente/1726244039

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼

```
1 {
2   "contrasena_cliente": "1234567890",
3   "estado_cliente": 1,
4   "persona": {
5     "nombre_persona": "James McCartney",
6     "edad_persona": 35,
7     "identificacion_persona": "1726244039",
8     "genero_persona": "Masculino",
9     "direccion_persona": "Av. 1234",
10    "telefono_persona": "0987654321"
11  }
12 }
13 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text ▼ ≡

1 Cliente actualizado.

6 • **SELECT * FROM sparquitecturamicroserviciobp.cliente;**

Result Grid ≡ ↺ Filter Rows: Edit: ✎ ➕ ➖ Export

	ID_CLIENTE	ID_PERSONA	CONTRASENA_CLIENTE	ESTADO_CLIENTE
▶	1	3	1234567890	1

Eliminar Cliente:

Pruebas Microservicios Solange Pico / Cliente / **EliminarCliente**

DELETE ▼ http://localhost:8081/api/cliente/eliminarCliente/1726244039

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
--	-----	-------

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text ▼ ≡

1 Cliente eliminado.

6 • **SELECT * FROM sparquitecturamicroserviciobp.cliente;**

Result Grid ≡ ↺ Filter Rows: Edit: ✎ ➕ ➖ Export/Import: 📁 📤

	ID_CLIENTE	ID_PERSONA	CONTRASENA_CLIENTE	ESTADO_CLIENTE
▶	1	3	1234567890	2
	2	1	1234	1
*	NULL	NULL	NULL	NULL

Nota: No se elimina como tal sino se actualiza el estado a inactivo.

CUENTA:

Crear Cuenta:

Pruebas Microservicios Solange Pico / Cuenta / CrearCuenta

POST

http://localhost:8081/api/cuenta/crearCuenta

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1  {
2    "numero_cuenta": 1313131313,
3    "tipo_cuenta": "1",
4    "saldo_inicial_cuenta": 800,
5    "estado_cuenta": 1,
6    "cliente": {
7      "contrasena_cliente": "1234567890",
8      "estado_cliente": 1,
9      "id_persona": 3,
10     "persona": {
11       "nombre_persona": "Paul McCartney",
12       "edad_persona": 35,
13       "identificacion_persona": "1726244039",
14       "genero_persona": "Masculino",
15       "direccion_persona": "Av 1234",
16       "telefono_persona": "0999999999"
17     }
18   }
19 }
```

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

Text

1

Cuenta creada.

7

SELECT * FROM sparquitecturamicroserviciobp.cuenta;

Result Grid

Filter Rows:

Edit

Export/Import

Wrap Cell Conte

	ID_CUENTA	ID_CLIENTE	NUMERO_CUENTA	TIPO_CUENTA	SALDO_INICIAL_CUENTA	ESTADO_CUENTA
▶	1	1	1313131313	1	800.00	1
*	NULL	NULL	NULL	NULL	NULL	NULL

Listar Cuenta:

Pruebas Microservicios Solange Pico / Cuenta / ListarCuentas

GET ⌵ http://localhost:8081/api/cuenta/listarCuentas

Params Authorization Headers (6) Body Pre-request Script Tests Setting

Query Params

KEY	VALUE
-----	-------

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ⌵ ≡

```
1 [
2   {
3     "id_cuenta": 1,
4     "id_cliente": 1,
5     "numero_cuenta": 1313131313,
6     "tipo_cuenta": "1",
7     "saldo_inicial_cuenta": 800.0,
8     "estado_cuenta": 1,
9     "cliente": {
10      "id_cliente": 1,
11      "contrasena_cliente": "1234567890",
12      "estado_cliente": 1,
13      "id_persona": 3,
14      "persona": {
15        "id_persona": 3,
16        "nombre_persona": "Paul McCartney",
17        "edad_persona": 40,
18        "identificacion_persona": "1726244039",
19        "direccion_persona": "NewYork",
20        "telefono_persona": "0987654323",
21        "genero_persona": "Masculino"
22      }
23    }
24  }
25 ]
```

7 • `SELECT * FROM sparquitecturamicroserviciobp.cuenta;`

ID_CUENTA	ID_CLIENTE	NUMERO_CUENTA	TIPO_CUENTA	SALDO_INICIAL_CUENTA	ESTADO_CUENTA
1	1	1313131313	1	800.00	1
2	1	1313131314	2	900.00	1
NULL	NULL	NULL	NULL	NULL	NULL

Editar Cuenta:

Pruebas Microservicios Solange Pico / Cuenta / EditarCuenta

PUT ▼ http://localhost:8081/api/cuenta/editarCuenta/1726244039

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "numero_cuenta": 1313131313,
3   "tipo_cuenta": "1",
4   "saldo_inicial_cuenta": 1500,
5   "estado_cuenta": 1,
6   "cliente": {
7     "contrasena_cliente": "1234567890",
8     "estado_cliente": 1,
9     "id_persona": 3,
10    "persona": {
11      "nombre_persona": "Paul McCartney",
12      "edad_persona": 35,
13      "identificacion_persona": "1726244039",
14      "genero_persona": "Masculino",
15      "direccion_persona": "Av 1234asb",
16      "telefono_persona": "09876789878"
17    }
18  }
19 }
20 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text ▼

1 Cuenta actualizado.
2 • **SELECT * FROM sparquitecturamicroserviciobp.cuenta;**

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Conte

ID_CUENTA	ID_CLIENTE	NUMERO_CUENTA	TIPO_CUENTA	SALDO_INICIAL_CUENTA	ESTADO_CUENTA
1	1	1313131313	1	1500.00	1
2	1	1313131314	2	900.00	1
NULL	NULL	NULL	NULL	NULL	NULL

Eliminar Cuenta:

Pruebas Microservicios Solange Pico / Cuenta / EliminarCuenta

DELETE ▼ http://localhost:8081/api/cuenta/eliminarCuenta/1313131313

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
-----	-------

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text ▼

1 Cuenta eliminada.

12 • **SELECT * FROM sparquitecturamicroserviciobp.cuenta;**

ID_CUENTA	ID_CLIENTE	NUMERO_CUENTA	TIPO_CUENTA	SALDO_INICIAL_CUENTA	ESTADO_CUENTA
1	1	1313131313	1	1500.00	2
2	1	1313131314	2	900.00	1
NULL	NULL	NULL	NULL	NULL	NULL

MOVIMIENTO:

Crear Movimiento:

Pruebas Microservicios Solange Pico / Movimiento / CrearMovimiento

POST

http://localhost:8081/api/movimiento/crearMovimiento

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

Text

1 Movimiento creado.

14 • SELECT * FROM sparquitecturamicroserviciobp.movimiento;

ID_MOVIMIENTO	ID_CUENTA	FECHA_MOVIMIENTO	TIPO_MOVIMIENTO	SALDO_MOVIMIENTO	ESTADO_MOVIMIENTO	DESCRIPCION_MOVIMIENTO
1	2	2023-01-17 00:04:49	debito	900.00	1	Retiro de: 10.0
2	2	2023-01-17 00:06:12	debito	890.00	1	Retiro de: 10.0
NULO	NULO	NULO	NULO	NULO	NULO	NULO

Listar Movimiento:

Pruebas Microservicios Solange Pico / Movimiento / ConsultarHistorial

POST

http://localhost:8081/api/movimiento/historialMovimiento

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

CONFIDENCIAL

Archivo Docker:

Aquí primero se creo el empaquetado del proyecto (.jar) y se creo la imagen desplegada en openjdk: 11

```
dockerfile x
1 FROM openjdk:11
2 COPY "./target/microservicios-0.0.1-SNAPSHOT.jar" "microservicio.jar"
3 EXPOSE 8081
4 ENTRYPOINT [ "java", "-jar", "microservicio.jar" ]
```

Pruebas Unitarias:

Se realizo dos clases para realizar las pruebas unitarias de Cuenta y Persona:

```
> src/test/java
  > com.solangepico.microservicio
    > CuentaTest.java
    > MicroservicioApplicationTests.java
    > PersonaTest.java
```

Persona:

```
13 import org.springframework.test.annotation.Rollback;
14
15 import com.solangepico.microservicio.Entity.Cuenta;
16 import com.solangepico.microservicio.Entity.Persona;
17 import com.solangepico.microservicio.Repository.PersonaRepository;
18
19 /**
20  * @author SolangePico
21  * @version 1.0.0 17/01/2023 ClaseTest que verificara los metodos de Persona..
22  */
23
24 @DataJpaTest
25 @AutoConfigureTestDatabase(replace = Replace.NONE)
26 public class PersonaTest {
27
28     @Autowired
29     private PersonaRepository personaRepo;
30
31     @Test
32     @Rollback(false)
33     public void testGuardarPersona() {
34         Persona perso = new Persona("Paul McCartney", 40, "1756789567", "Masculino", "0987654323", "NewYork");
35         Persona persoGuar = personaRepo.save(perso);
36         assertNotNull(persoGuar);
37     }
38 }
39
40 }
```

Pruebas Microservicios Solange Pico / Persona / listarPersonas

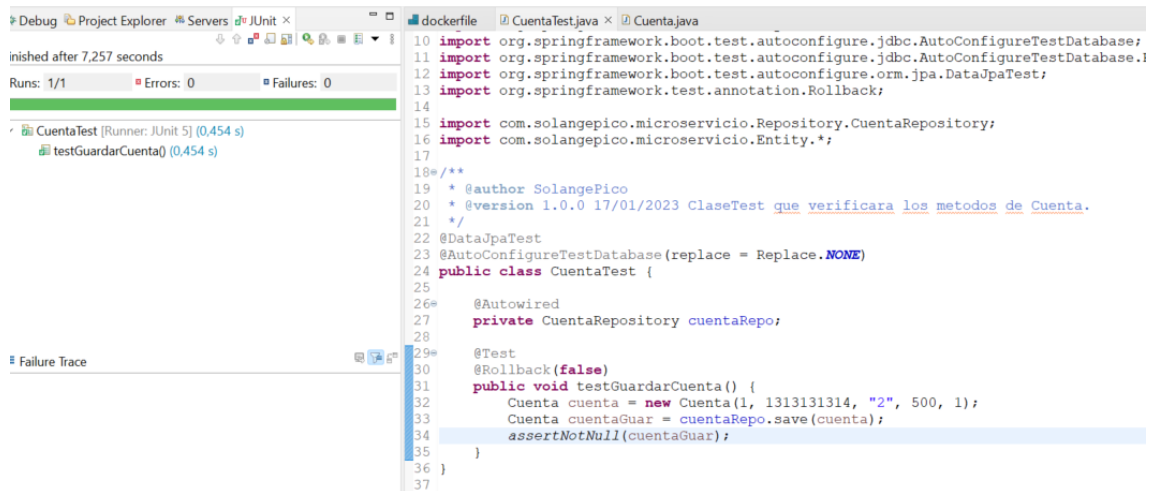
```
GET http://localhost:8081/api/persona/listarPersonas

Params Authorization Headers (6) Body Pre-request Script Tests

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↕

13 "nombre_persona": "Paul McCartney",
14 "edad_persona": 40,
15 "identificacion_persona": "1726244039",
16 "direccion_persona": "NewYork",
17 "telefono_persona": "0987654323",
18 "genero_persona": "Masculino"
19 },
20 {
21     "id_persona": 9,
22     "nombre_persona": "Paul McCartney",
23     "edad_persona": 40,
24     "identificacion_persona": "1756789567",
25     "direccion_persona": "Masculino",
26     "telefono_persona": "0987654323",
27     "genero_persona": "NewYork"
28 }
29 ]
```



```
10 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
11 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
12 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
13 import org.springframework.test.annotation.Rollback;
14
15 import com.solangepico.microservicio.Repository.CuentaRepository;
16 import com.solangepico.microservicio.Entity.*;
17
18 /**
19  * @author SolangePico
20  * @version 1.0.0 17/01/2023 ClaseTest que verificara los metodos de Cuenta.
21  */
22 @DataJpaTest
23 @AutoConfigureTestDatabase(replace = Replace.NONE)
24 public class CuentasTest {
25
26     @Autowired
27     private CuentaRepository cuentaRepo;
28
29     @Test
30     @Rollback(false)
31     public void testGuardarCuenta() {
32         Cuenta cuenta = new Cuenta(1, 1313131314, "2", 500, 1);
33         Cuenta cuentaGuar = cuentaRepo.save(cuenta);
34         assertNotNull(cuentaGuar);
35     }
36 }
37
```

Pruebas Microservicios Solange Pico / Cuenta / ListarCuentas

GET

http://localhost:8081/api/cuenta/listarCuentas

Params

Authorization

Headers (6)

Body

Pre-request Script

Test Results

KEY	VALUE
Key	Value

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
48 {
49     "id_cuenta": 3,
50     "id_cliente": 1,
51     "numero_cuenta": 1313131314,
52     "tipo_cuenta": "2",
53     "saldo_inicial_cuenta": 500.0,
54     "estado_cuenta": 1,
55     "cliente": {
56         "id_cliente": 1,
57         "contrasena_cliente": "1234567890",
58         "estado_cliente": 1,
59         "id_persona": 3,
60         "persona": {
```