

# Manual de usuario

## DESCARGA EL PROYECTO DE GITHUB CON EL COMANDO

```
git clone https://github.com/jormartinezl/PagoService.git
```

```
apple@Apples-MacBook-Pro descargar % git clone https://github.com/jormartinezl/PagoService.git
Clonando en 'PagoService'...
remote: Enumerating objects: 62, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 62 (delta 1), reused 8 (delta 1), pack-reused 53
Recibiendo objetos: 100% (62/62), 66.95 MiB | 7.27 MiB/s, listo.
Resolviendo deltas: 100% (1/1), listo.
```

## ABRE UNA TERMINAL Y ENTRA AL PROYECTO PAGOSERVICE

```
apple@Apples-MacBook-Pro descargar % cd PagoService
```

## EJECUTA EL COMANDO PARA EMPAQUETAR EL PROYECTO SIN EJECUTAR LAS PRUEBAS UNITARIAS

```
./mvnw clean package -DskipTests
```

```
[INFO] Replacing main artifact /Users/apple/Documents/descargar/PagoService/target
[INFO] The original artifact has been renamed to /Users/apple/Documents/descargar/
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.944 s
[INFO] Finished at: 2024-01-22T01:59:53-06:00
[INFO] -----
```

## GENERA IMAGEN DOCKER

```
docker-compose build java_app
```

```
[INFO] -----
apple@Apples-MacBook-Pro PagoService % docker-compose build java_app
[+] Building 0.6s (7/7) FINISHED
=> [java_app internal] load .dockerignore
=> => transferring context: 2B
=> [java_app internal] load build definition from Dockerfile
=> => transferring dockerfile: 299B
=> [java_app internal] load metadata for docker.io/library/openjdk:17-slim
=> [java_app internal] load build context
=> => transferring context: 76.73MB
=> CACHED [java_app 1/2] FROM docker.io/library/openjdk:17-slim
=> [java_app 2/2] COPY target/PagoService-0.0.1-SNAPSHOT.jar app.jar
=> [java_app] exporting to image
=> => exporting layers
=> => writing image sha256:eb8fcd86fd93257da744531278e68740477a253547737ba51b29fd72a4fd90cc
=> => naming to docker.io/library/pago_java_app:1.0.0
```

## VALIDA QUE SE GENERO LA IMAGEN

```
apple@Apples-MacBook-Pro PagoService % docker image ls
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
pago_java_app        1.0.0       eb8fcd86fd93  About a minute ago  479MB
mysql                1.0.0       1868a550c677  2 days ago     639MB
```

## GENERA Y LEVANTA EL ORQUESTADO

docker-compose up

```
openjdk:17-slim 8a5a217ec52a 21 months ago
apple@Apples-MacBook-Pro PagoService % docker-compose up
[+] Building 0.0s (0/0)
[+] Running 5/3
✓ Network pagoservice_default Created
✓ Container mysql2 Created
✓ Container zookeeper2 Created
✓ Container kafka2 Created
✓ Container java_app Created
Attaching to java_app, kafka2, mysql2, zookeeper2
zookeeper2: l... User
```

## EN UNA NUEVA TERMINAL VALIDA QUE SE LEVANTEN LOS 4 CONTENEDORES

docker ps -a

```
apple@Apples-MacBook-Pro prueba-tecnica-pago % docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
eeda263d56f8   pago_java_app:1.0.0                "java -jar app.jar"     2 minutes ago Up 2 minutes   0.0.0.0:8080->8080/tcp             java_app
bce90bcd9721   confluentinc/cp-kafka:7.4.0        "/etc/confluent/dock..." 2 minutes ago Up 2 minutes   0.0.0.0:9092->9092/tcp             kafka2
9fdedf2aa292   confluentinc/cp-zookeeper:7.4.0    "/etc/confluent/dock..." 2 minutes ago Up 2 minutes   2181/tcp, 2888/tcp, 3888/tcp       zookeeper2
e56bde1ef83b   mysql:8.0.29                        "docker-entrypoint.s..." 2 minutes ago Up 2 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp   mysql2
e67fd6f67acb3   phomwadin                           "/docker-entrypoint..." 2 days ago     Exited (0) 6 hours ago            mysql-phomwadin
```

## ENTRA A LA CONSOLA DE MYSQL

docker exec -it mysql2 mysql -uuser -p

## INTRODUCIR LA CONTRASEÑA

password123#

```
apple@Apples-MacBook-Pro prueba-tecnica-pago % docker exec -it mysql2 mysql -uuser -p
Enter password: █
```

**SELECCIONA LA BASE DE DATOS Y CREA LA TABLA PAGO**  
USE pagos\_db;

### CREA LA TABLA PAGO

```
CREATE TABLE pago (id bigint not null auto_increment,  
beneficiario varchar(255) not null, cantidad integer not null,  
concepto varchar(255) not null, emisor varchar(255) not null,  
estatus varchar(255) not null, monto decimal(20,2) not null,  
primary key (id));
```

**VALIDA QUE SE CREO LA TABLA**  
SHOW TABLES;

```
mysql> USE pagos_db;  
Database changed  
mysql> CREATE TABLE pago (id bigint not null auto_increment, beneficiario varchar(255) not null, cantidad integer not null, con  
,2) not null, primary key (id));  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_pagos_db |  
+-----+  
| pago                |  
+-----+  
1 row in set (0.01 sec)
```

**SAL DE LA CONSOLA DE MYSQL**  
CTRL + D

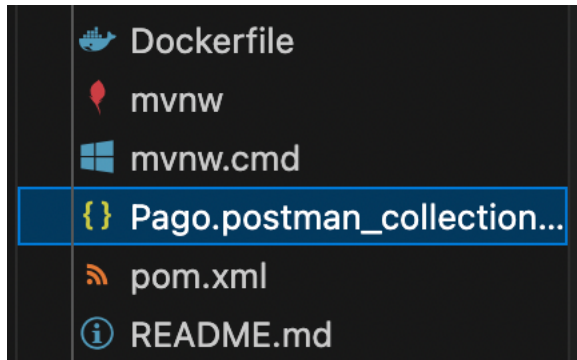
**PUEDES VISUALIZAR LA DOCUMENTACIÓN DE LAS APIS EN LA SIGUIENTE LIGA**  
<http://localhost:8080/swagger-ui/index.html#/>

The screenshot displays the Swagger UI for an API named 'pago-controller'. At the top, the Swagger logo is visible, along with the text 'Reported by SMARTBEAR' and the path '/v3/api-docs'. Below this, the 'OpenAPI definition' is shown, with a version indicator 'v0' and 'OAS 3.0'. A 'Servers' section contains a dropdown menu with the value 'http://localhost:8080 - Generated server url'. The main part of the interface lists four API endpoints for the 'pago-controller':

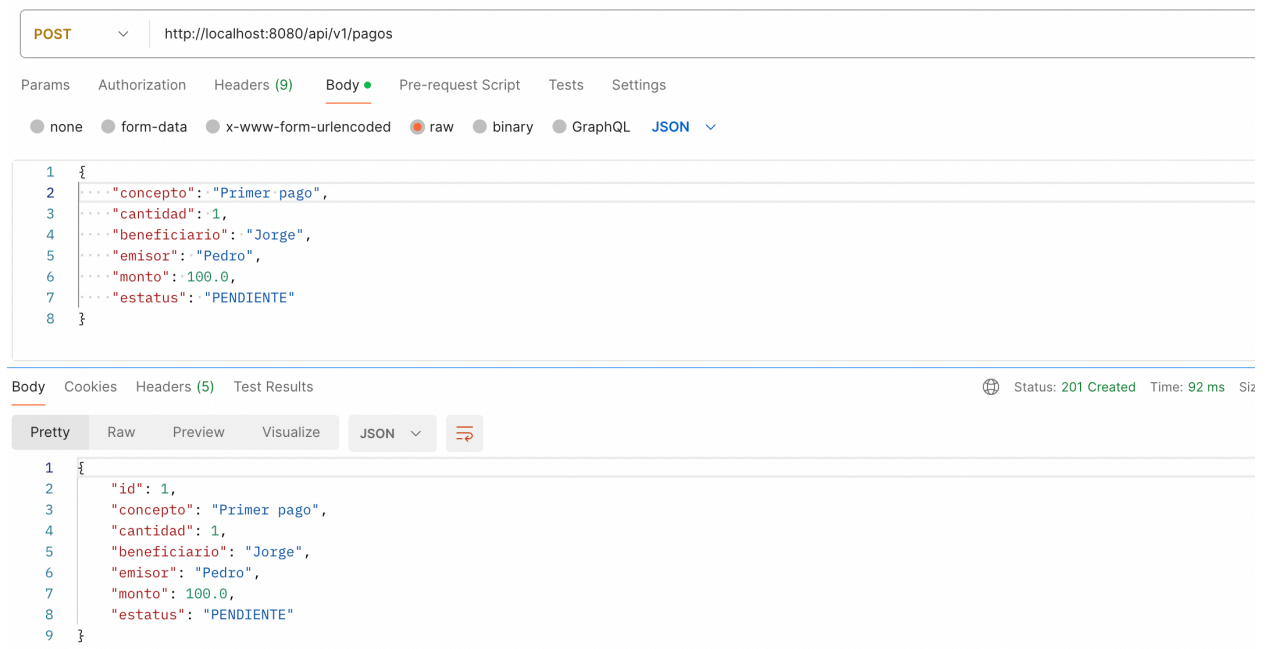
- GET** /api/v1/pagos: Busca todos los pagos
- POST** /api/v1/pagos: Genera un nuevo pago
- PATCH** /api/v1/pagos/{id}/estatus: Actualiza el estatus del pago
- GET** /api/v1/pagos/{id}: Busca un pago

At the bottom, there is a 'Schemas' section with a link to 'PagoDto'.

**IMPORTA LA COLECCIÓN DE POSTMAN, ESTA ESTA EN LA RAÍZ DE NUESTRO PROYECTO PAGO.POSTMAN\_COLLECTION.JSON**



**UNA VEZ IMPORTADO NUESTRA COLECCIÓN DE POSTMAN EJECUTA EL MÉTODO POST PARA GENERAR UN NUEVO PAGO**



VALIDA EL REGISTRO CON EL MÉTODO GET

GET

http://localhost:8080/api/v1/pagos/1

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

BodyCookiesHeaders (5)Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"id": 1,

3

"concepto": "Primer pago",

4

"cantidad": 1,

5

"beneficiario": "Jorge",

6

"emisor": "Pedro",

7

"monto": 100.00,

8

"estatus": "PENDIENTE"

9

}

ACTUALIZA EL ESTATUS PARA QUE SE REGISTRE EN KAFKA

PATCH

http://localhost:8080/api/v1/pagos/1/estatus?estatus=EN\_PROCESO

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description
<input checked="" type="checkbox"/>	estatus	EN_PROCESO	
	Key	Value	Description

BodyCookiesHeaders (5)Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"id": 1,

3

"concepto": "Primer pago",

4

"cantidad": 1,

5

"beneficiario": "Jorge",

6

"emisor": "Pedro",

7

"monto": 100.00,

8

"estatus": "EN\_PROCESO"

9

}

Status: 200 OKTime:

## ENTRA A LA CONSOLA DE KAFKA

`docker exec -it kafka2 bash`

## INICIA EL CONSUMIDOR EN LA CONSOLA DE KAFKA PARA EL TÓPICO ESTATUS-TOPIC

`kafka-console-consumer --topic estatus-topic --from-beginning --bootstrap-server localhost:9092`

```
apple@Apples-MacBook-Pro prueba-tecnica-pago % docker exec -it kafka2 bash
[appuser@bce90bcd9721 ~]$ kafka-console-consumer --topic estatus-topic --from-beginning --bootstrap-server localhost:9092
{"id":1,"concepto":"Primer pago","cantidad":1,"beneficiario":"Jorge","emisor":"Pedro","monto":100.00,"estatus":"EN_PROCESO"}
```