Manual de usurio

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
```

GENERA Y LEVANTA EL ORQUESTADO

```
docker-compose up
```

```
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
```

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
```

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;
```

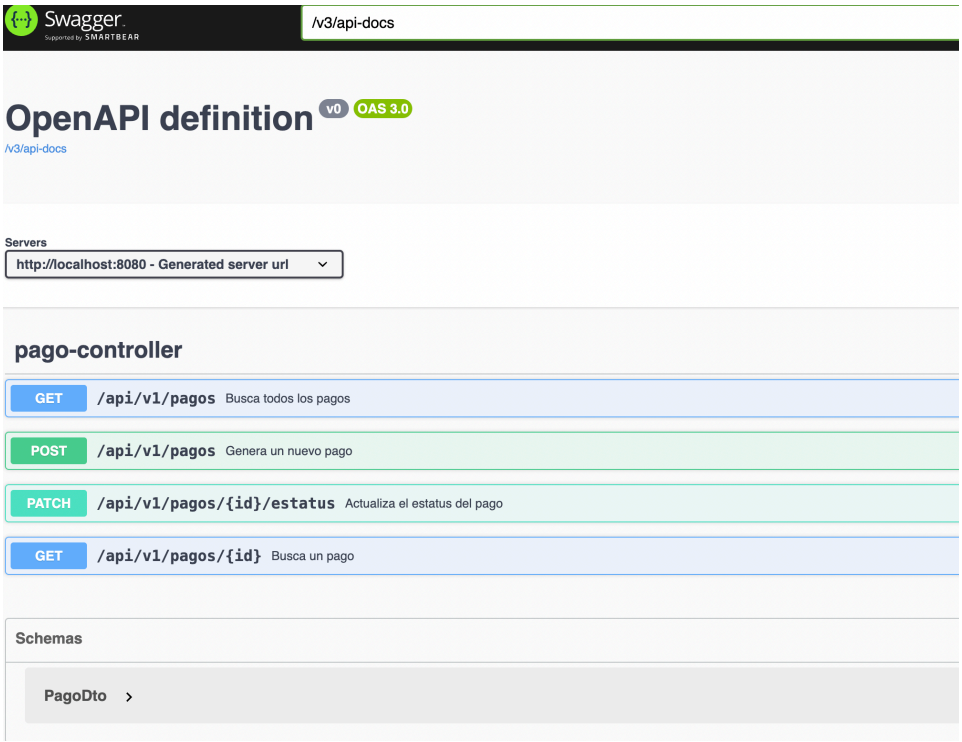
```
mysql> USE pagos_db;
Database changed
mysql> CREATE TABLE pago (id bigint not null auto_increment, beneficiario varchar(255) not null, cantidad integer not null, con
,2) not null, primary key (id));
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW TABLES;
+--------------------+
| Tables_in_pagos_db |
+--------------------+
| pago               |
+--------------------+
1 row in set (0.01 sec)
```

SAL DE LA CONSOLA DE MYSQL

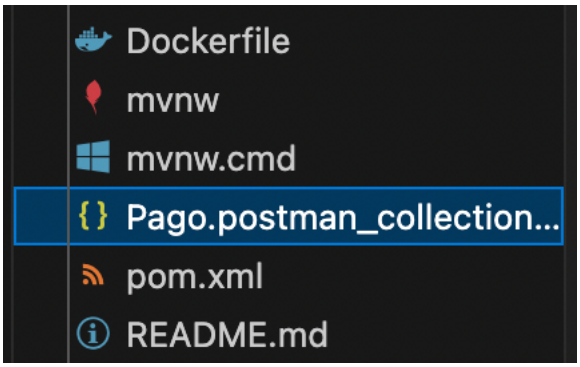
```
CTRL + D
```

PUEDES VISUALIZAR LA DOCUMENTACIÓN DE LAS APIS EN LA SIGUIENTE LIGA

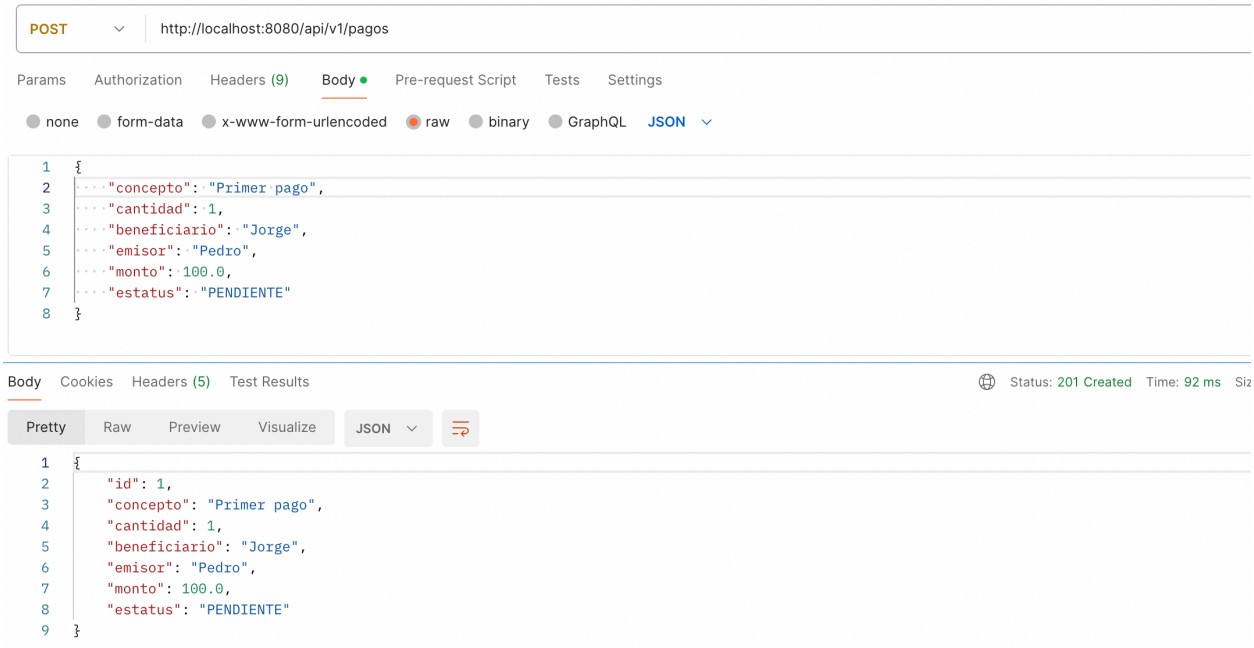
```
http://localhost:8080/swagger-ui/index.html#/
```



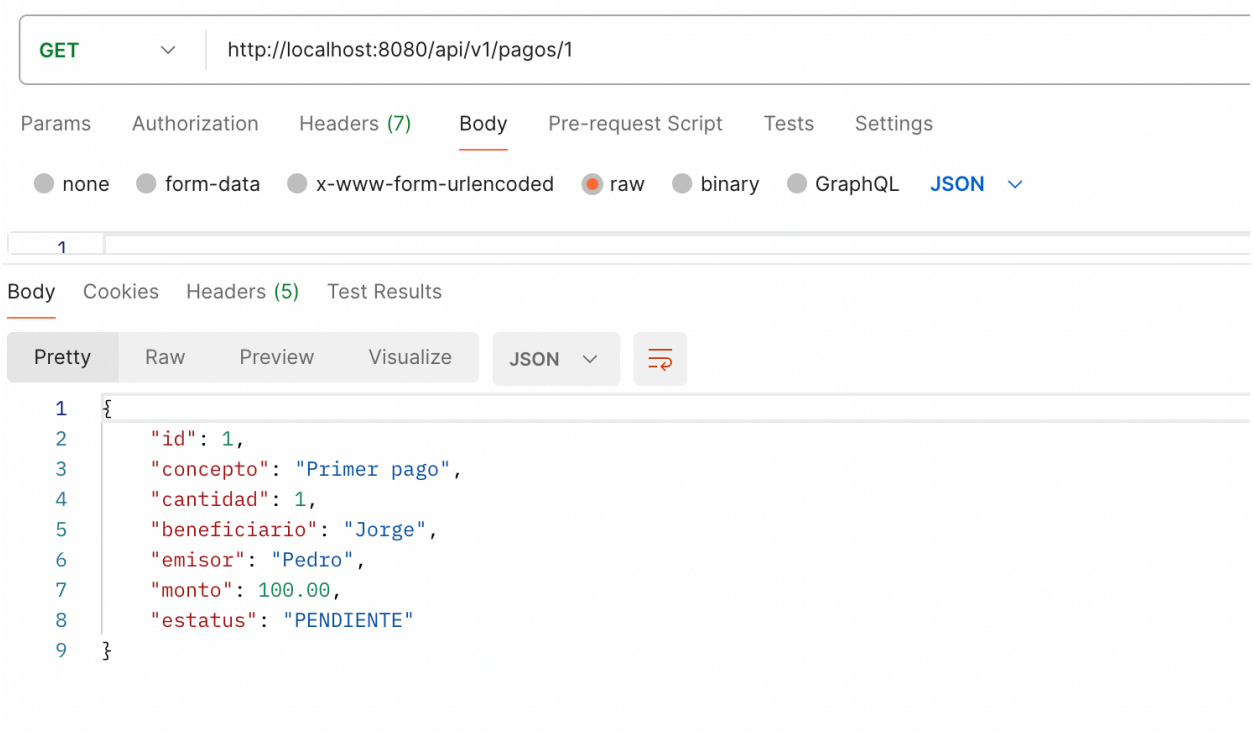
IMPORTA LA COLECCIÓN DE POSTMAN, ESTA ESTA EN LA RAÍZ DE NUESTRO PROYECTO PAGO.POSTMAN_COLLECTION.JSON



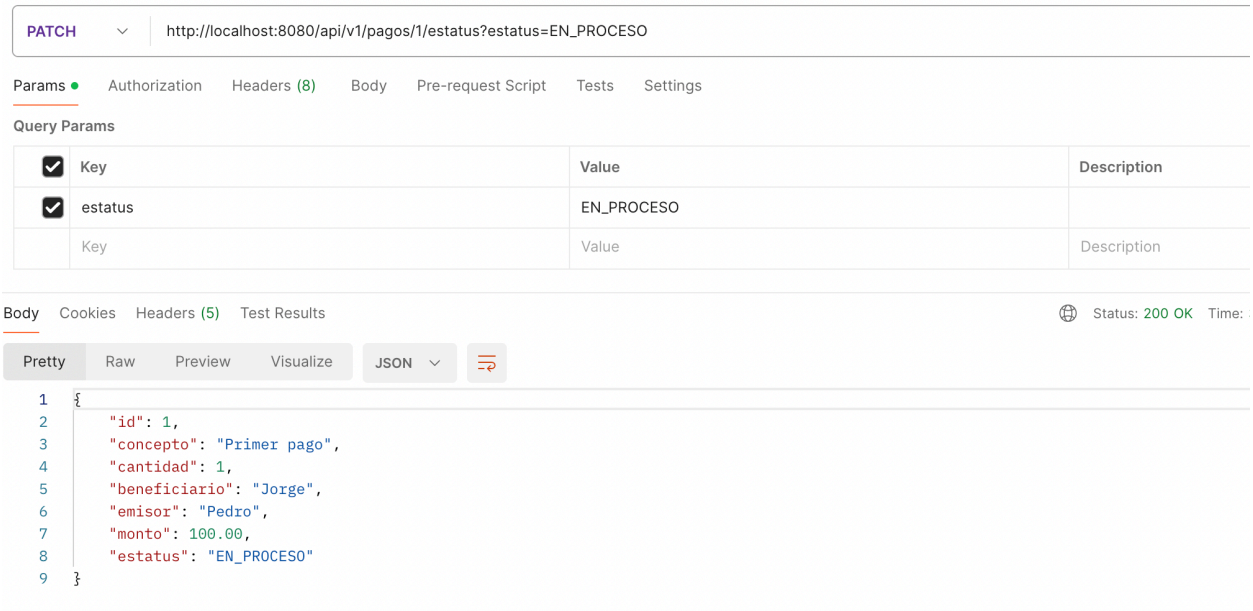
UNA VEZ IMPORTADO NUESTRA COLECCIÓN DE POSTMAN EJECUTA EL MÉTODO POST PARA GENERAR UN NUEVO PAGO



VALIDA EL REGISTRO CON EL MÉTODO GET



ACTUALIZA EL ESTATUS PARA QUE SE REGISTRE EN KAFKA



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
```