

# DOCUMENTO DE INSTALACIÓN DEL CHALLENGE DE EVERTEC

**Por: Oscar Bohórquez**

**Fecha: 23-Agosto-2021**

## Objetivo:

El presente documento cuenta con las suficientes instrucciones para ejecutar el challenge de forma exitosa y fácil.

## Prerequisitos:

1. git cliente o clonar el proyecto desde un navegador
2. Docker instalado de la pagina oficial "<https://docs.docker.com/get-docker>", instalar la ultima versión de Docker en su Sistema Operativo.

## Instalación:

Repositorio con el challenge se encuentra en el siguiente link de github: <https://github.com/oscbohr/ddd-springboot-react-postgres.git>

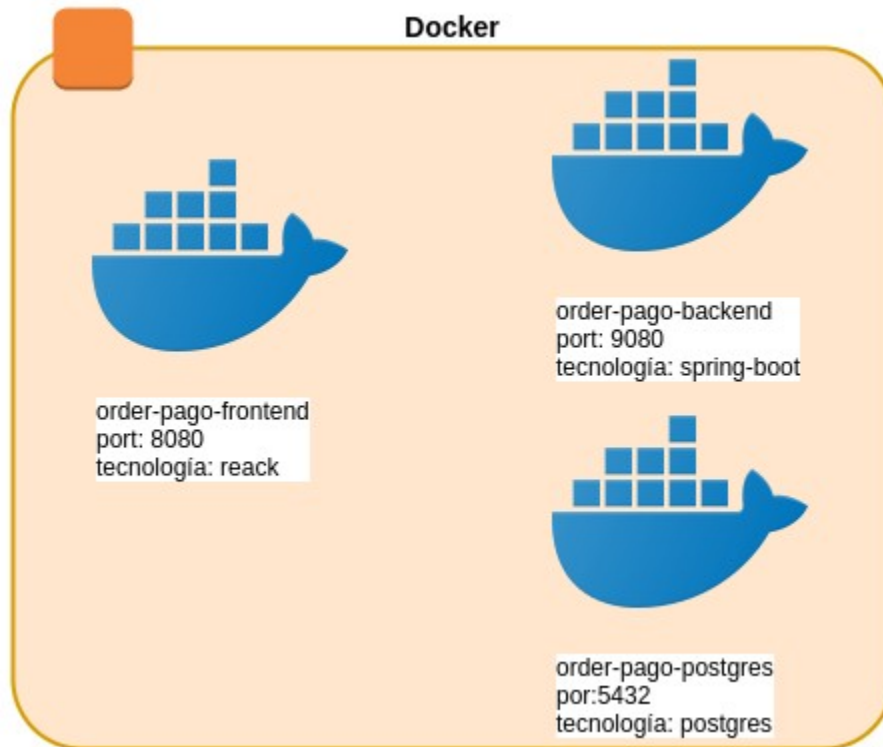
1. Clonar el repositorio dentro del directorio deseado y entrar al directorio descargado "ddd-springboot-react-postgres", revisar que se tengan la siguiente estructura de directorios y archivos:

```
ml-orderpago-challenge
doc
docker-compose.yml
db
```

comando: git clone <https://github.com/oscbohr/ddd-springboot-react-postgres.git>

comando: cd ddd-springboot-react-postgres

2. Crear la imagen y ejecutar los contenedores con docker-compose:



NOTA: la aplicación react está ejecutándose por el puerto: 8080, la base de datos postgres en el puerto 5432, y spring-boot en el puerto 9080; tener en cuenta que si la máquina donde se está ejecutando los contenedores ya tienen esos puertos en uso, debemos cambiarlos en el archivo Dockerfile de cada contenedor al igual que en el archivo "docker-compose.yml"

#### DockerFile: spring-boot

```
# set base image (host OS)
FROM openjdk:11.0.4-jre

# set the working directory in the container
WORKDIR /app

# copy the dependencies file to the working directory
COPY /ml-orderpago-challenge/backend/ChallengePlaceToPay/target/ChallengePlaceToPay-0.0.1-SNAPSHOT.jar
/app/ChallengePlaceToPay-0.0.1-SNAPSHOT.jar

# port Expose
EXPOSE 9080

# run
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/app/ChallengePlaceToPay-0.0.1-SNAPSHOT.jar"]
```

#### DockerFile: react

```
#### Stage 1: Build the react application
FROM node:12.4.0-alpine as build
WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH

COPY ./ml-orderpago-challenge/frontend/challenge-place-to-pay-ui/package.json /app/package.json
RUN npm install --silent
RUN npm install react-scripts@3.0.1 -g --silent
COPY ./ml-orderpago-challenge/frontend/challenge-place-to-pay-ui /app/
RUN npm run build

# production environment
FROM nginx:1.16.0-alpine
COPY --from=build /app/build /usr/share/nginx/html
RUN rm /etc/nginx/conf.d/default.conf
COPY ./ml-orderpago-challenge/frontend/nginx.conf /etc/nginx/conf.d
EXPOSE 8080
CMD ["nginx", "-g", "daemon off;"]
```

## `docker-compose.yml`

```
versión: '3'

services:
  backend:
    build:
      context: .
      dockerfile: ./ml-orderpago-challenge/backend/Dockerfile
    container_name: orderpago-backend
    hostname: backend
    network_mode: bridge
    restart: always
    expose:
      - "9080"
    ports:
      - "9080:9080"
    depends_on:
      - "db"
    links:
      - "db"
  db:
    image: postgres:latest
    network_mode: bridge
    hostname: db
    container_name: orderpago-postgres
```

```
restart: always
expose:
  - "5432"
ports:
  - "5432:5432"
volumes:
  - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
environment:
  - POSTGRES_PASSWORD=123456
  - POSTGRES_USER=postgres
  - POSTGRES_DB=testdb
frontend:
build:
  context: .
  dockerfile: ./ml-orderpago-challenge/frontend/Dockerfile
container_name: orderpago-frontend
hostname: frontend
network_mode: bridge
restart: always
expose:
  - "8080"
ports:
  - "8080:8080"
depends_on:
  - "backend"
links:
  - "backend"
```

Ejecutar el comando: “**docker-compose up -d --build**”

```
lobohorquez@bobohorquez ddd-springboot-react-postgres]$ docker-compose up -d --build
Building backend
Step 1/5 : FROM openjdk:11.0.4-jre
--> fa6d3c4702db
Step 2/5 : WORKDIR /app
--> Using cache
--> 17402edbab28
Step 3/5 : COPY ./ml-orderpago-challenge/backend/ChallengePlaceToPay/target/ChallengePlaceToPay-0.0.1-SNAPSHOT.jar /app/ChallengePlaceToPay-0.0.1-SNAPSHOT.jar
--> c92c7f8e86f3
Step 4/5 : EXPOSE 9080
--> Running in 8571771639b8
Removing intermediate container 8571771639b8
--> 45fc96e8a4a8
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app/ChallengePlaceToPay-0.0.1-SNAPSHOT.jar"]
--> Running in 82da9310cf5a
Removing intermediate container 82da9310cf5a
--> 05c2eac47321
Successfully built 05c2eac47321
Successfully tagged ddd-springboot-react-postgres_backend:latest
Building frontend
Step 1/14 : FROM node:12.4.0-alpine as build
--> d4edda39fb81
Step 2/14 : WORKDIR /app
--> Running in 3d8714a5388c
Removing intermediate container 3d8714a5388c
--> db0a40fa0424
Step 3/14 : ENV PATH /app/node_modules/.bin:$PATH
--> Running in 5a63078520de
Removing intermediate container 5a63078520de
--> dc3b9d3b9a21
Step 4/14 : COPY ./ml-orderpago-challenge/frontend/challenge-place-to-pay-ui/package.json /app/package.json
--> cff0d9c080a5
Step 5/14 : RUN npm install --silent
--> Running in f850bbf514de
```

3. Verificar la correcta ejecución del contenedor de la aplicación frontend, aplicación backend y de la Base de Datos ejecutando:  
comando: `docker-compose ps -a`  
resultado:

```
root@ombohorquez10 ddd-springboot-react-postgres]#  
root@ombohorquez10 ddd-springboot-react-postgres]# docker-compose ps -a  
-----  
Name                                Command                                State      Ports  
-----  
orderpago-backend                  java -Djava.security.egd=f ...        Up         0.0.0.0:9080->9080/tcp  
orderpago-frontend                 nginx -g daemon off;                 Up         80/tcp, 0.0.0.0:8080->8080/tcp  
orderpago-postgres                 docker-entrypoint.sh postgres        Up         0.0.0.0:5432->5432/tcp  
root@ombohorquez10 ddd-springboot-react-postgres]#  
root@ombohorquez10 ddd-springboot-react-postgres]#  
root@ombohorquez10 ddd-springboot-react-postgres]#  
root@ombohorquez10 ddd-springboot-react-postgres]#
```

4. Detener e iniciar los contenedores para que tomen el nombre de host configurado en el archivo `docker-compose.yml`. Para ello ejecute: `docker-compose stop`, y seguido `docker-compose start`

```
[root@ombohorquez10 ddd-springboot-react-postgres]#  
[root@ombohorquez10 ddd-springboot-react-postgres]# docker-compose stop  
Stopping orderpago-frontend ... done  
Stopping orderpago-backend ... done  
Stopping orderpago-postgres ... done  
[root@ombohorquez10 ddd-springboot-react-postgres]#  
[root@ombohorquez10 ddd-springboot-react-postgres]#  
[root@ombohorquez10 ddd-springboot-react-postgres]# docker-compose start  
Starting db ... done  
Starting backend ... done  
Starting frontend ... done  
[root@ombohorquez10 ddd-springboot-react-postgres]#
```

5. Verificar los logs de los contenedores ejecutando los siguientes comandos:
  1. `docker logs orderpago-backend`
  2. `docker logs orderpago-frontend`
  3. `docker logs orderpago-postgres`

```
[root@ombohorquez10 ddd-springboot-react-postgres]#  
[root@ombohorquez10 ddd-springboot-react-postgres]#  
[root@ombohorquez10 ddd-springboot-react-postgres]# docker logs orderpago-backend
```

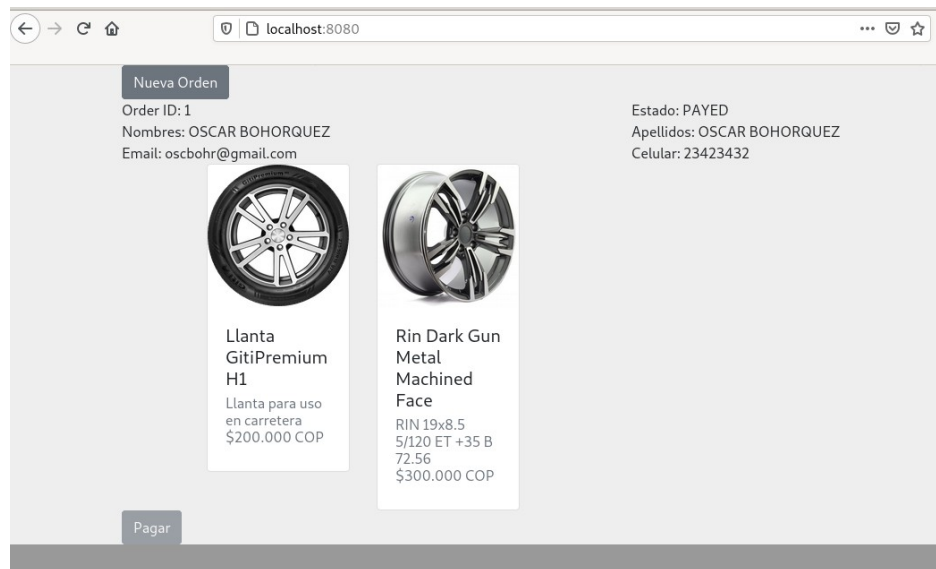
6. Ingresar al contenedor de la Base de Datos y verificar la estructura de tablas. Para ello ingrese los siguientes comandos:

1. `"docker exec -it orderpago-postgres /bin/bash"`
2. `"su postgres"`
3. `"psql"`
4. `"\c testdb"`
5. `"\dt"`

```
[root@ombohorquez10 ddd-springboot-react-postgres]# docker exec -it orderpago-postgres /bin/bash  
root@db:/# su postgres  
postgres@db:/$ psql  
psql (12.2 (Debian 12.2-2.pgdg100+1))  
Type "help" for help.  
  
postgres=# \c testdb  
You are now connected to database "testdb" as user "postgres".  
testdb=# \dt  
          List of relations  
Schema |      Name      | Type  | Owner  
-----+-----+-----+-----  
public | articulo        | table | postgres  
public | orders          | table | postgres  
public | transaccion_orders | table | postgres  
(3 rows)  
  
testdb=#
```

NOTA: Si la estructura de tablas no se encuentra, ejecutar sobre la misma terminal el script `"init.sql"` localizado en el directorio `"db"` del repositorio descargado.

7. Ejecutar la aplicación: Abrir el browser de su preferencia (Firefox o Chrome) e ingresar la siguiente url “<http://localhost:8080/>”. Se visualizaran las ordenes emitidas por la tienda (o en blanco). Para crear una nueva Orden, oprimir el botón “Nueva Orden”



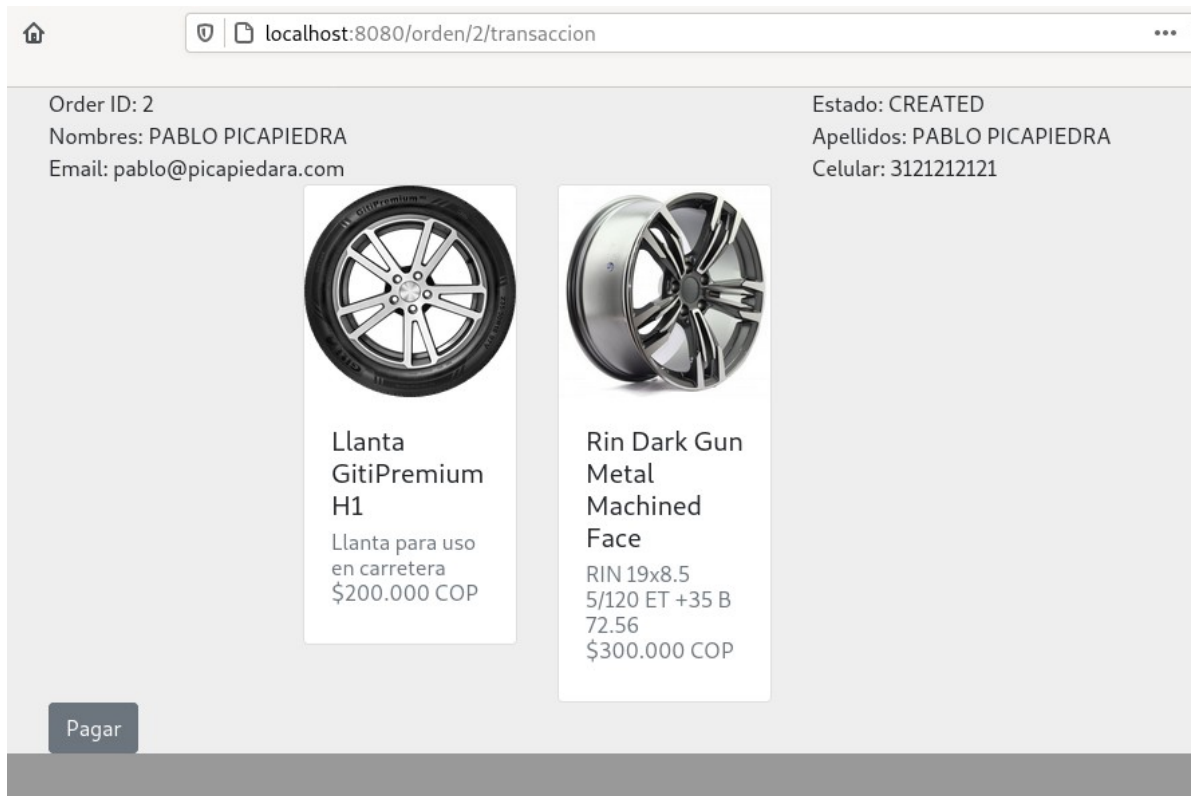
Ingresa la información del cliente y oprima el botón “submit”

The screenshot shows a web browser window at the URL `localhost:8080/crearOrden`. The page displays a form to create a new order with the following fields:

- Nombres:** pablo
- Apellidos:** picapiedra
- Email:** pablo@picapiedara.com
- Celular:** 3121212121


A **Submit** button is located at the bottom of the form.

Se visualizará la Orden con su información y estado con la opción de “Pagar”




Order ID: 2  
Nombres: PABLO PICAPIEDRA  
Email: pablo@picapiedara.com

Estado: CREATED  
Apellidos: PABLO PICAPIEDRA  
Celular: 3121212121



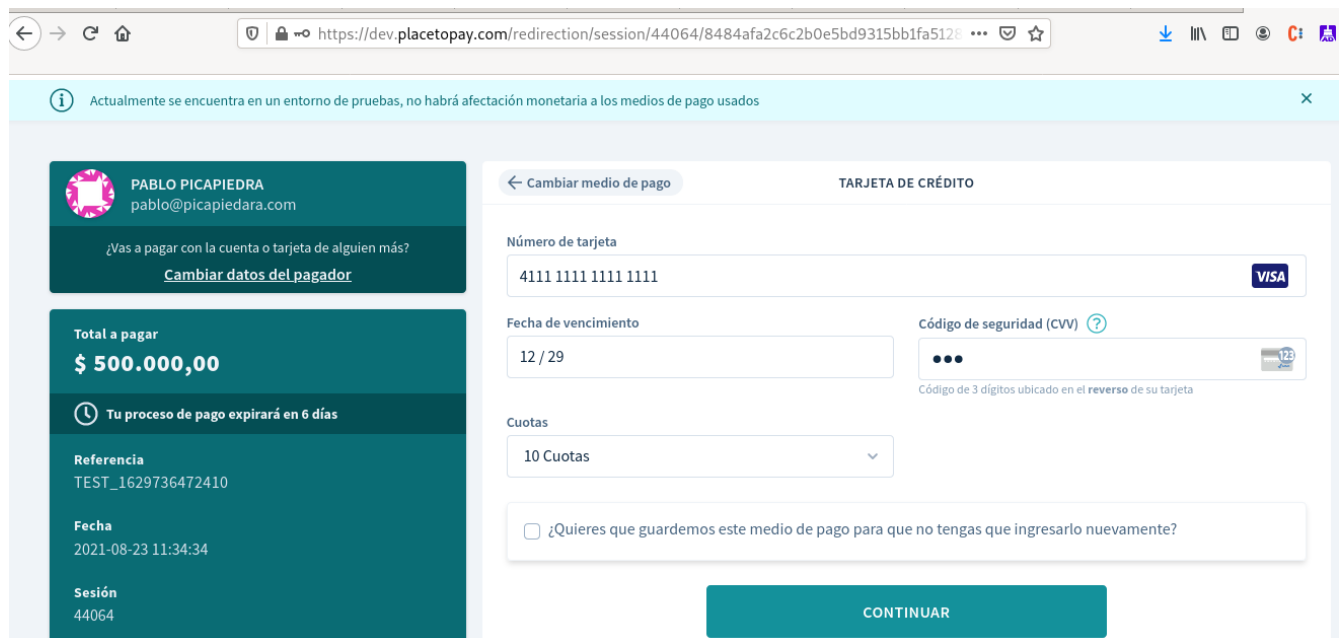
Llanta  
GitiPremium  
H1  
Llanta para uso  
en carretera  
\$200.000 COP




Rin Dark Gun  
Metal  
Machined  
Face  
RIN 19x8.5  
5/120 ET +35 B  
72.56  
\$300.000 COP

Pagar

Al oprimir “Pagar” se hace el llamado a la pasarela de Pago



Actualmente se encuentra en un entorno de pruebas, no habrá afectación monetaria a los medios de pago usados



PABLO PICAPIEDRA  
pablo@picapiedara.com

¿Vas a pagar con la cuenta o tarjeta de alguien más?  
[Cambiar datos del pagador](#)

**Total a pagar**  
**\$ 500.000,00**


**Tu proceso de pago expirará en 6 días**

**Referencia**  
TEST\_1629736472410



**Fecha**  
2021-08-23 11:34:34

**Sesión**  
44064

[← Cambiar medio de pago](#) **TARJETA DE CRÉDITO**

Número de tarjeta  
4111 1111 1111 1111 

Fecha de vencimiento  
12 / 29

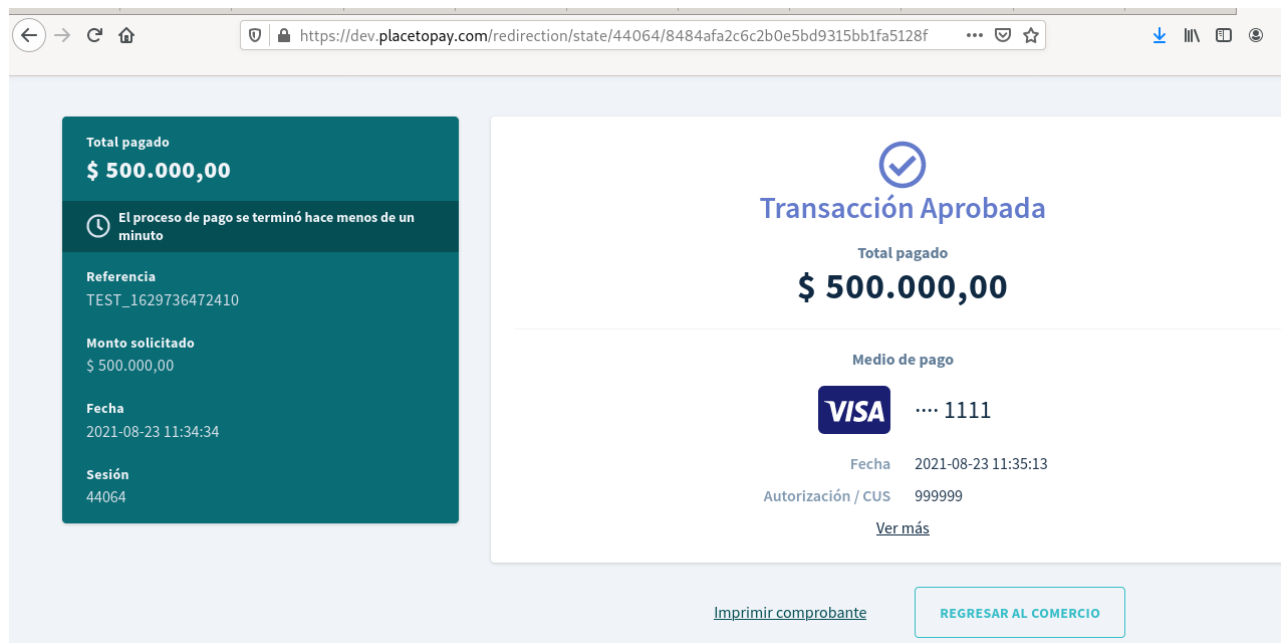
Código de seguridad (CVV)   
...   
Código de 3 dígitos ubicado en el reverso de su tarjeta

Cuotas  
10 Cuotas

☐ ¿Quieres que guardemos este medio de pago para que no tengas que ingresarlo nuevamente?

**CONTINUAR**





Al oprimir el botón “Regresar al comercio” se devuelve al Detalle de la Orden con la información del pago: Si el pago es satisfactorio se deshabilita el botón “Pagar” de lo contrario se habilita para posteriormente ejecutar el pago.

