



**PROYECTO
FALSIFICACIONES EN PRENDAS DE LUJO
HYPERLEDGER FABRIC**

<https://github.com/DavidDGWeb/ProjectHyperledgerFabric>



ESCUELA DE NEGOCIOS ESPECIALIZADA EN NUEVAS TECNOLOGÍAS

www.ebiseducation.com



DAVID GUTIÉRREZ CARBALLO

PROYECTO FIN DE POSTGRADO BLOCKCHAIN ENGINEERING REDES PERMISIONADAS

www.linkedin.com/in/david-gutierrez-carballo/

Índice

DISEÑO Y ANÁLISIS FUNCIONAL	3
HOJA DE CONTROL.....	3
INTRODUCCIÓN	4
Objetivos	4
Fuentes.....	4
Tecnologías aplicas	4
ESPECIFICACIONES Y CASOS DE USO DEL SISTEMA	5
Actores del sistema.....	5
FLUJO GENERAL DE PROCESO.....	5
INTERFACES Y PANTALLAS	5
ESPECIFICACIONES DE CASOS DE USO DEL SISTEMA.....	6
Listado Casos de uso del Sistema	6
Diagrama UML	6
Detalles Casos de uso del Sistema	7
ENTIDADES DEL SISTEMA	9
Diagrama E/R	10
Diagrama E/R - Chaincode.....	10
DEFINICIÓN DEL CHAINCODE	11
Definición	11
Métodos	11
INTEGRACIÓN CON OTROS SISTEMAS.....	11
DESPLIEGUE Y TEST DEL PROYECTO	12
Alcance del proyecto	12
Instalación de componentes necesarios	12
Instalación de Hyperledger Fabric.....	14
Desplegamos la red de Hyperledger Fabric	15
Creamos un proyecto Gradle para la gestión del chaincode	16
Configuración de Gradle	17
Creamos el chaincode con las funciones que vamos a invocar.....	19
Primeros test, deployamos e invocamos el chaincode a través de sus funciones	20
Inspeccionamos el CouchDB.....	22
Log de la red – Hyperledger Explorer.....	23
Log de la red – Prometheus y Grafana	26
Creación de una Api-Rest para Backend y Frontend con Swagger.....	27
Test frontal a través de Swagger	30

DISEÑO Y ANÁLISIS FUNCIONAL

HOJA DE CONTROL

Proyecto	BLOCKCHAIN PERMISIONADAS – FALSIFICACIONES EN PRENDAS DE LUJO		
Entregable	Documento diseño, análisis funcional, despliegue y test		
Autor	David Gutiérrez Carballo		
Versión/Edición	V.1-2024	Fecha Versión	21/01/2024
Aprobado por		Fecha Aprobación	
		Nº Total de Páginas	33

REGISTRO DE CAMBIOS

Versión	Motivo del Cambio	Responsable del Cambio	Fecha
001	Versión inicial	DAVID GUTIÉRREZ	21/01/2024

INTRODUCCIÓN

Este documento detalla las especificaciones funcionales, despliegue y test para el proyecto desarrollado en blockchain permissionada Hyperledger Fabric, como solución a la falsificación de prendas de lujo.

Objetivos

El objetivo de este proyecto es desarrollar una red Hyperledger Fabric que registre en la blockchain, las prendas fabricadas de marcas de lujo para evitar la venta de falsificaciones. A su mismo los puntos de venta de las marcas al vender las prendas, pondrán asignar al propietario de la misma, que mediante un código qr dará la veracidad de la prenda y de su propiedad.

El chaincode también está diseñado para desde un punto de venta hacer la transferencia de la prenda a otro propietario, así como saber la lista de propietarios que ha tenido la prenda.

FUENTES Y TECNOLOGÍAS

Fuentes

- Repositorio GitHub: <https://github.com/DavidDGWeb/ProjectHyperledgerFabric>
- Documento análisis funcional, despliegue y test
- Video de presentación del proyecto

Tecnologías aplicas

- Blockchain permissionada Hyperledger Fabric para el despliegue de la red con un orderer y dos peers, certificados CA, un canal y base de datos couchDB
- Diseño de chaincode en JAVA con Spring Boot, como framework de desarrollo
- Api-Rest para Backend y Frontend con Swagger
- Para monitorear y testear la red se ha utilizado Prometheus, Grafana e Hyperledger Explorer

ESPECIFICACIONES Y CASOS DE USO DEL SISTEMA

Actores del sistema

1	ORDERER
Descripción	<i>Será quien recibe las transacciones de los nodos peers, construye los bloques a través del consenso, el cual envía a todos los peers que pertenezcan al canal, que validan el bloque y lo incorporan a la blockchain</i>
Comentarios	

2	FABRICANTES
Descripción	Son nodos peers quién registran la fabricación de la prenda en la blockchain
Comentarios	

3	PUNTOS DE VENTA
Descripción	Son nodos peers quién asignan la propiedad de la prenda al comprador
Comentarios	

FLUJO GENERAL DE PROCESO

- El servicio orderer será el encargado de mantener la red
- Habrán nodos peers que actúan como fábricas para registrar la fabricación de la prenda y dar la veracidad de la misma
- Habrán nodos peers que actuarán como puntos de venta para registrar la propiedad del comprador.

INTERFACES Y PANTALLAS

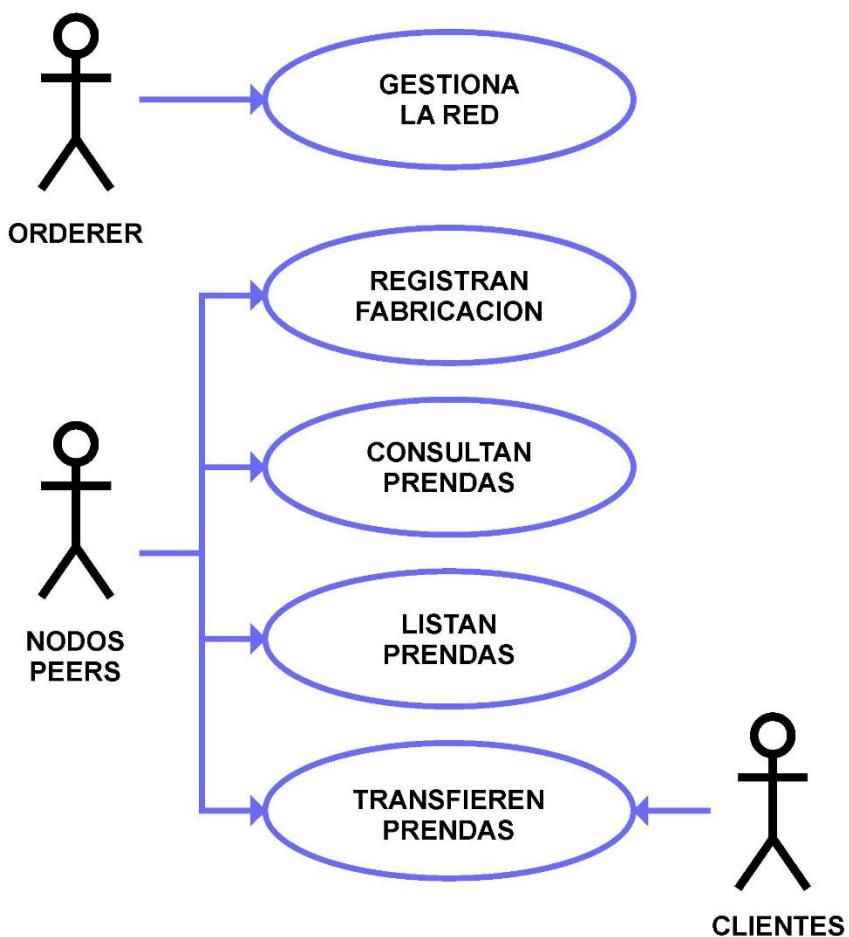
- La consola para el despliegue de la red
- Plataforma web con acceso a la BBDD CouchDB
- Plataforma web para sacar métricas de la red Prometheus, Grafana e Hyperledger Explorer
- Plataforma web frontal con Swagger para los test

ESPECIFICACIONES DE CASOS DE USO DEL SISTEMA

Listado Casos de uso del Sistema

Código	Descripción	Prioridad	Dependencia
CU1	Crear prenda desde las fábricas		
CU2	Consultar prenda		
CU3	Listar prendas		
CU4	Transferir prenda		
CU5	Borrar prenda		

Diagrama UML



Detalles Casos de uso del Sistema

CU1	<i>Crear prendas desde las fábricas</i>
Descripción	Alta nueva prenda
Pantalla	Crear prenda
Rol	Nodo peers autorizado por el orderer para esta función
Flujo	<ul style="list-style-type: none"> - <i>Desde el frontal la fábrica con el rol de peers, una vez fabricada la prenda podrá registrarla en la red. Para ello se requiere:</i> <ul style="list-style-type: none"> - <i>id (nº que identifique la prenda)</i> - <i>Marca</i> - <i>Categoría</i> - <i>País de fabricación</i> - <i>Fecha de fabricación</i> - <i>Propietario</i> - <i>Código qr (identifica la prenda)</i>
Flujo alternativo	
Requisito técnico	

CU2	<i>Consultar prenda</i>
Descripción	Consultar la prenda
Pantalla	Ver prenda
Rol	Nodo peers autorizado por el orderer para esta función
Flujo	<ul style="list-style-type: none"> - <i>Desde el frontal la fábrica o un punto de venta a través del id, puede consultar la fabricación, el propietario o lista de propietarios que ha tenido la prenda</i>
Flujo alternativo	
Requisito técnico	

CU3	<i>Listar prendas</i>
Descripción	Listar la prenda
Pantalla	Listar prenda
Rol	Nodo peers autorizado por el orderer para esta función
Flujo	<ul style="list-style-type: none"> - <i>Desde el frontal la fábrica o un punto de venta pueden listar todas las prendas</i>
Flujo alternativo	
Requisito técnico	

CU4	<i>Transferir prenda</i>
Descripción	Transferir prenda
Pantalla	Transferir prenda
Rol	Nodo peers autorizado por el orderer para esta función
Flujo	- <i>Desde el frontal el punto de venta puede transferir la prenda a otro propietario</i>
Flujo alternativo	
Requisito técnico	

CU5	<i>Borrar prenda</i>
Descripción	Borrar prenda
Pantalla	Borrar prenda
Rol	Nodo peers autorizado por el orderer para esta función
Flujo	- <i>Desde el frontal se puede borrar la prenda. Se borrará del World State pero no del ledger por la inmutabilidad que tiene la blockchain</i>
Flujo alternativo	
Requisito técnico	

ENTIDADES DEL SISTEMA

1	<i>ORDERER</i>
Descripción	<i>ORDERER el encargado de mantener la red a través de los permisos y del conceso establecido</i>
Atributos	<ul style="list-style-type: none"> - <i>Para este ejemplo desplegamos dos nodos peers</i> - <i>Tiene certificación CA</i> - <i>Tiene un canal de comunicación mychannel</i> - <i>Tiene desplegada una base de datos CouchDB, para de manera eficiente tener una vista más rápida</i>

2	<i>NODO PEER</i>
Descripción	<i>El NODO PEER tendrá como función ser una fábrica o un punto de venta</i>
Atributos	<ul style="list-style-type: none"> - <i>Valida transacciones en la red</i> - <i>Ejecuta el chaincode</i> - <i>Participa en el consenso</i> - <i>Mantiene el ledger para mantener la integridad y coherencia de la red BC</i>

3	<i>CHAINCODE (Contrato inteligente)</i>
Descripción	<i>El CHAINCODE está diseñado para ejecutar o consultar las transacciones en la red</i>
Atributos	<ul style="list-style-type: none"> - <i>registrarPrenda</i> - <i>imprimirPrenda</i> - <i>listarPrendas</i> - <i>transferenciaPrenda</i> - <i>borrarPrenda</i>

Diagrama E/R

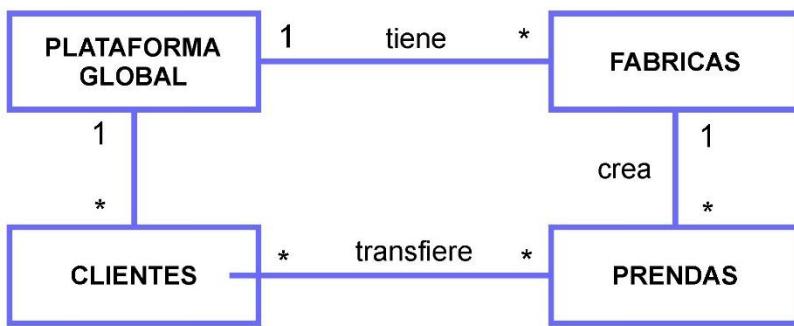
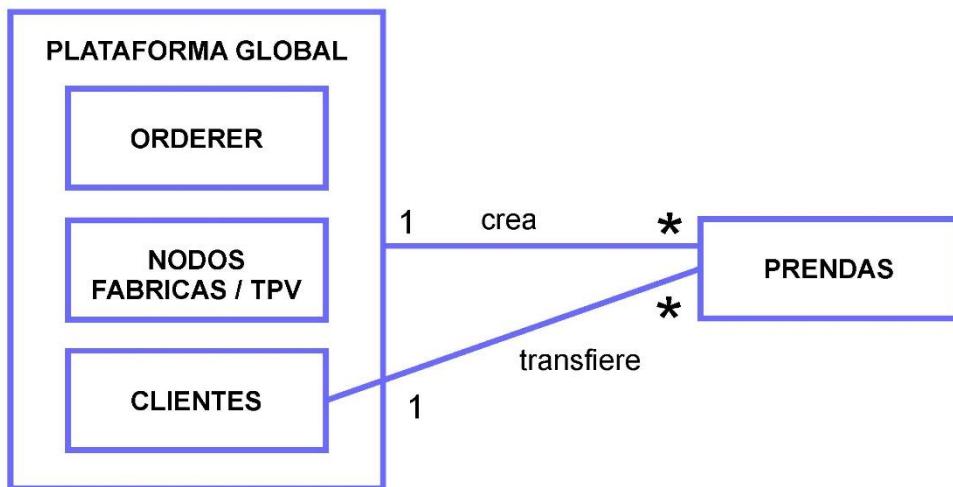


Diagrama E/R- Chaincode



DEFINICIÓN DEL CHAINCODE

La red está gobernada por un chaincode desarrollado en Java, que gestiona la plataforma.

Definición

CHAINCODE		
<u>CAMPO</u>	<u>TIPO</u>	<u>COMENTARIO</u>
id	Entero, autonumérico	Identifica la prenda
marca	String	Registro de fichajes
categoría	String	Categoría de la prenda (bolso, reloj, camisa, etc.)
paisFabricación	String	País donde se fabrica la prenda
fechaFabricacion	String	Fecha cuando se fabrica la prenda
propietario	String	El propietario actual de la prenda
qr	String	Código QR que identifica la prenda

Métodos

- registrarPrenda -> método para que la fábrica pueda registrar la prenda a su fabricación
- imprimirPrenda -> a través del id puedes imprimir una prenda
- listarPrendas -> listar todas las prendas
- transferenciaPrenda -> transferir la prenda al propietario
- borrarPrenda -> para borrar la prenda

INTEGRACIÓN CON OTROS SISTEMAS

- Hyperledger Explorer, Prometheus y Grafanas para ver las métricas y testear la red

DESPLIEGUE Y TEST DEL PROYECTO

Alcance del proyecto

- Red de test de Hyperledger Fabric con un orderer, dos peer, certificados Ca, CouchDB y un canal
- Herramientas de diagnóstico de la red, Prometheus y Grafana
- Framework Gradle para generar el proyecto Java y diseñar el chaincode
- El diseño del chaincode sería:
 - Para este ejemplo cada Peer será una fábrica asociada a la marca y podrá registrar en la red la fabricación de la prenda
 - Se podrá consultar en la red la prenda (lo que dará la veracidad de que no sea una falsificación, porque la red certifica su fabricación)
 - Transferencia de la prenda para asignar la propiedad al comprador (los peer podrán asignar o transferir la prenda a un comprador)
- Creación de API-REST para conectar al frontal a través de swagger, para hacer los test
- IntelliJ IDEA como editor de código para el despliegue

Instalación de componentes necesarios

Comprobamos versión y cuenta git

```
ev-k8s@evk8s-VirtualBox:~$ git --version
git version 2.34.1
ev-k8s@evk8s-VirtualBox:~$ git config --list
user.name=DavidDGWeb
user.email=info@creativoslan.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=git@github.com:DavidDGWeb-bsm-labs.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
ev-k8s@evk8s-VirtualBox:~$
```

Comprobamos versión Docker y Docker compose

```
ev-k8s@evk8s-VirtualBox:~$ docker version
Client: Docker Engine - Community
  Version:           24.0.7
  API version:      1.43
  Go version:       go1.20.10
  Git commit:       afdd53b
  Built:            Thu Oct 26 09:07:41 2023
  OS/Arch:          linux/amd64
  Context:          default

Server: Docker Engine - Community
  Engine:
    Version:           24.0.7
    API version:      1.43 (minimum version 1.12)
    Go version:       go1.20.10
    Git commit:       311b9ff
    Built:            Thu Oct 26 09:07:41 2023
    OS/Arch:          linux/amd64
    Experimental:    false
  containerd:
    Version:           1.6.24
    GitCommit:         61f9fd88f79f081d64d6fa3bb1a0dc71ec870523
  runc:
    Version:           1.1.9
    GitCommit:         v1.1.9-0-gccaecfc
  docker-init:
    Version:           0.19.0
    GitCommit:         de40ad0
ev-k8s@evk8s-VirtualBox:~$
```

```
ev-k8s@evk8s-VirtualBox:~$ docker-compose version
docker-compose version 1.29.2, build unknown
docker-py version: 5.0.3
CPython version: 3.10.12
OpenSSL version: OpenSSL 3.0.2 15 Mar 2022
ev-k8s@evk8s-VirtualBox:~$
```

```
ev-k8s@evk8s-VirtualBox:~$ sudo apt-get install curl
[sudo] contraseña para ev-k8s:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
curl ya está en su versión más reciente (7.81.0-1ubuntu1.15).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 139 no actualizados.
ev-k8s@evk8s-VirtualBox:~$ sudo apt-get install jq
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
jq ya está en su versión más reciente (1.6-2.1ubuntu3).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 139 no actualizados.
ev-k8s@evk8s-VirtualBox:~$ sudo apt-get install golang-go
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
golang-go ya está en su versión más reciente (2:1.18~0ubuntu2).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 139 no actualizados.
```

Instalamos Java

```
sudo apt-get install openjdk-11-jdk
```

Instalamos IntelliJ IDEA en Ubuntu como editor de código para gestionar el proyecto Gradle

```
sudo snap find "intellij"
sudo snap install intellij-idea-community --classic
```

Instalación de Hyperledger Fabric

Descargamos los binarios y ejemplos de Docker:

```
curl -sSL0  
https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh
```

Instalamos Hyperledger Fabric

```
sudo ./install-fabric.sh docker samples binary
```

Damos permisos sudo a la carpeta fabric-samples

```
sudo chmod -R 777 * fabric-samples
```

Desplegamos la red de Hyperledger Fabric

Paramos la red y eliminamos cualquier red y contenedores, que puedan estar desplegados.

```
cd fabric-samples/test-network  
.network.sh down
```

```
docker stop $(docker ps -a -q)  
docker rm $(docker ps -a -q)  
docker volume prune  
docker network prune
```

Desplegamos la red con un orderer, dos peers, certificación Ca, un canal mychannel y con BBDD Couchdb.

```
cd fabric-samples/test-network  
.network.sh up createChannel -ca -s couchdb  
docker ps -a
```

Exportando las variables de entorno, comprobamos que el canal se haya creado correctamente
`peer channel list`

Comprobamos que la red esté iniciada y corriendo.

```
watch docker ps
```

Una vez tengamos el chaincode alojado en la carpeta test-network para proyecto Gradle podemos desplegarlo ejecutando lo siguiente. También sirve para si hacemos modificaciones en el chaincode, deployarlo nuevamente sin necesidad de desplegar nuevamente la red.

```
./network.sh deployCC -ccn chaincode -ccp ../chaincode/ -ccl java
```

*Si analizamos el log del deploy, lo primero que hace es compilar el código Java ejecutando el installDist de gradlew, después lo empaqueta, lo instala en la Org1 y Org2, los aprueba y los comitea.

Para parar la red, lo podemos hacer ejecutando.

```
sudo ./network.sh down
```

Creamos un proyecto Gradle para la gestión del chaincode

<https://start.spring.io/>

The screenshot shows the Spring Initializr interface with the following settings:

- Project:** Gradle - Groovy (selected)
- Language:** Java (selected)
- Spring Boot:** 3.1.8 (SNAPSHOT) (selected), 3.2.2 (SNAPSHOT), 3.2.1
- Project Metadata:**
 - Group: com.example
 - Artifact: chaincode
 - Name: chaincode
 - Description: Proyecto DGWeb - EBIS
 - Package name: com.example.chaincode
- Packaging:** Jar (selected)
- Java:** 21 (selected), 17

El archivo que genera Gradle lo descomprimimos y lo alojamos dentro de la carpeta fabric-samples. Este proyecto contendrá los chaincode y las librerías necesarias para conectar con Hyperledger Fabric para el despliegue del chaincode.

```
ev-k8s@evk8s-VirtualBox:~/fabric-samples$ ls -l
total 540
drwxrwxrwx  3 root  root    4096 ene 18 08:49 asset-transfer-abac
drwxrwxrwx 17 root  root    4096 ene 18 08:49 asset-transfer-basic
drwxrwxrwx  8 root  root    4096 ene 18 08:49 asset-transfer-events
drwxrwxrwx  6 root  root    4096 ene 18 08:49 asset-transfer-ledger-queries
drwxrwxrwx  7 root  root    4096 ene 18 08:49 asset-transfer-private-data
drwxrwxrwx  5 root  root    4096 ene 18 08:49 asset-transfer-sbe
drwxrwxrwx  5 root  root    4096 ene 18 08:49 asset-transfer-secured-agreement
drwxrwxrwx  5 root  root    4096 ene 18 08:49 auction-dutch
drwxrwxrwx  4 root  root    4096 ene 18 08:49 auction-simple
drwxrwxrwx  2 1001 docker  4096 ago 29 20:25 bin
drwxrwxrwx  3 1001 docker  4096 ago  2 15:59 builders
drwxr-xr-x  7 ev-k8s ev-k8s  4096 oct 29 02:34 chaincode
drwxrwxrwx  1 root  root   398453 ene 18 08:49 CHANGELOG.md
```

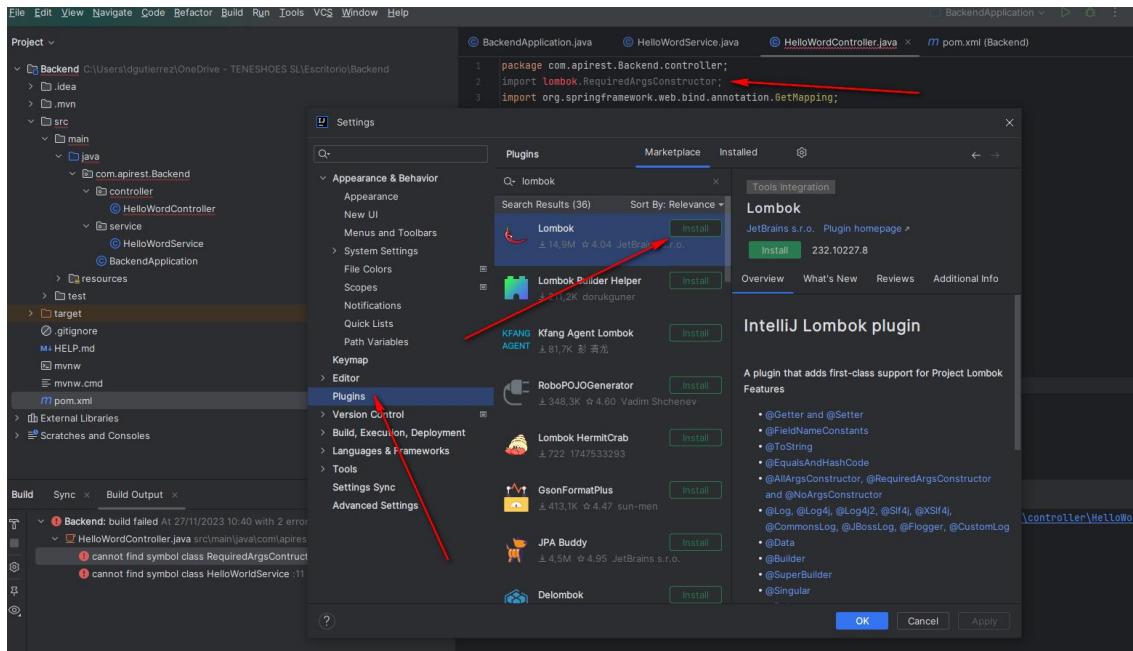
```
ev-k8s@evk8s-VirtualBox:~/fabric-samples$ cd chaincode/
ev-k8s@evk8s-VirtualBox:~/fabric-samples/chaincode$ ls -l
total 40
drwxr-xr-x  9 ev-k8s ev-k8s 4096 oct 29 03:08 build
-rw-r--r--  1 ev-k8s ev-k8s   791 oct 29 02:34 build.gradle
drwxr-xr-x  3 ev-k8s ev-k8s 4096 oct 28 17:11 gradle
-rw-r--r--  1 ev-k8s ev-k8s  8639 oct 28 17:11 gradlew
-rw-r--r--  1 ev-k8s ev-k8s  2868 oct 28 17:11 gradlew.bat
-rw-r--r--  1 ev-k8s ev-k8s   576 oct 28 17:11 HELP.md
-rw-r--r--  1 ev-k8s ev-k8s   31 oct 28 17:11 settings.gradle
drwxr-xr-x  4 ev-k8s ev-k8s 4096 oct 28 17:11 src
ev-k8s@evk8s-VirtualBox:~/fabric-samples/chaincode$
```

Configuración de Gradle

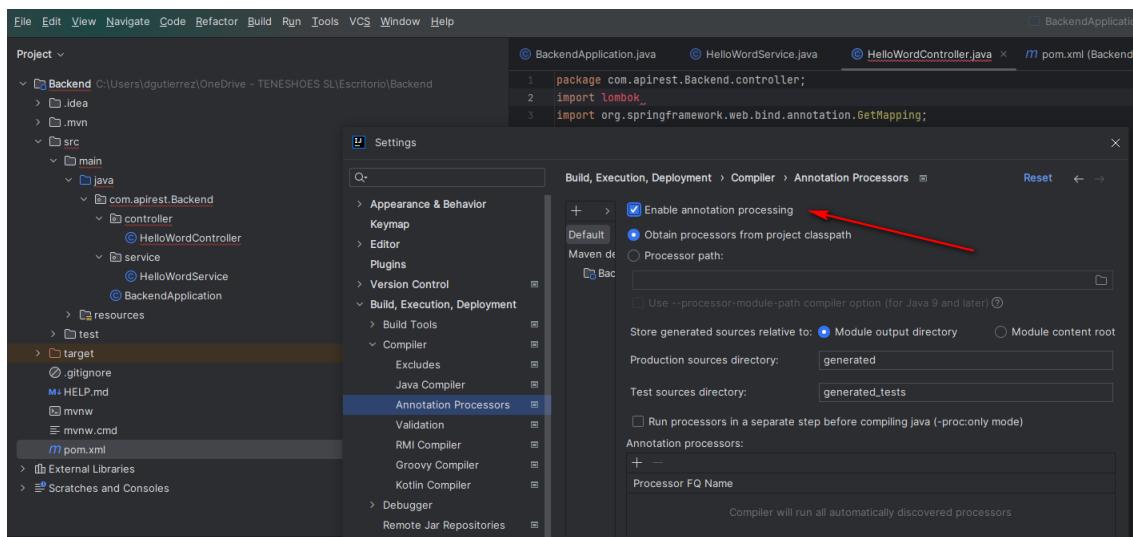
Configuramos el IntelliJ IDEA – añadir capturas

- Instalar JDK
- Añadir pluggins y librerías en Gradle
- Hacemos setting de Gandle

Añadimos el plugin de Lombok para facilitar las anotaciones del código

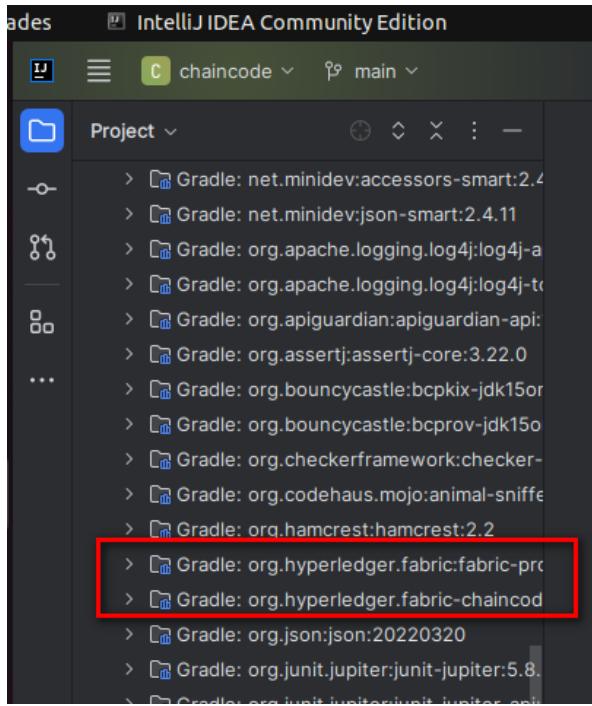


Activamos el proceso de anotaciones



Revisamos la versión Java del compilador

Comprobamos que, en las librerías, tenemos instaladas las de hyperledger fabric



Para instalar la tarea installDist ejecutamos

```
chmod +x gradlew  
.gradlew installDist
```

```
ev-k8s@evk8s-VirtualBox:~/fabric-samples/chaincode$ ls -l  
total 40  
drwxr-xr-x 9 ev-k8s ev-k8s 4096 oct 29 03:08 build  
-rw-r--r-- 1 ev-k8s ev-k8s 791 oct 29 02:34 build.gradle  
drwxr-xr-x 3 ev-k8s ev-k8s 4096 oct 28 17:11 gradle  
-rwxr-xr-x 1 ev-k8s ev-k8s 8639 oct 28 17:11 gradlew  
-rw-r--r-- 1 ev-k8s ev-k8s 2868 oct 28 17:11 gradlew.bat  
-rw-r--r-- 1 ev-k8s ev-k8s 576 oct 28 17:11 HELP.md  
-rw-r--r-- 1 ev-k8s ev-k8s 31 oct 28 17:11 settings.gradle  
drwxr-xr-x 4 ev-k8s ev-k8s 4096 oct 28 17:11 src  
ev-k8s@evk8s-VirtualBox:~/fabric-samples/chaincode$ chmod +x gradlew  
ev-k8s@evk8s-VirtualBox:~/fabric-samples/chaincode$ ./gradlew installDist  
Starting a Gradle Daemon, 1 stopped Daemon could not be reused, use --status for details  
<-----> 0% INITIALIZING [24s]  
> Evaluating settings
```

Podemos ver todas las tareas creadas ejecutando

```
./gradlew task
```

```
ev-k8s@evk8s-VirtualBox:~/fabric-samples/chaincode$ ./gradlew task  
> Task :tasks  
  
-----  
Tasks runnable from root project 'chaincode'  
-----  
  
Application tasks  
-----  
bootRun - Runs this project as a Spring Boot application.  
run - Runs this project as a JVM application
```

Creamos el chaincode con las funciones que vamos a invocar

```
J AssetPrenda.java X J PrendaTransfer.java
src > main > java > com > bsm > chaincode > J AssetPrenda.java
1 package com.bsm.chaincode;
2
3 import com.owlike.genson.annotation.JsonProperty;
4 import org.hyperledger.fabric.contract.annotation.DataType;
5 import org.hyperledger.fabric.contract.annotation.Property;
6
7 import java.util.List;
8 import java.util.Objects;
9
10 @DataType()
11 public final class AssetPrenda {
12
13     @Property()
14     private final String id;
15
16     @Property()
17     private final String marca;
18
19     @Property()
20     private final String categoria;
21
22     @Property()
23     private final String paisFabricacion;
24
25     @Property()
26     private final String propietario;
27
28     @Property()
29     private final String qr;
30
31     @Property()
32     private final List<String> propietarios;
33
34
35     public AssetPrenda(@JsonProperty("id") final String id,
36                         @JsonProperty("marca") final String marca,
37                         @JsonProperty("categoria") final String categoria,
38                         @JsonProperty("paisFabricacion") final String paisFabricacion,
39                         @JsonProperty("propietario") final String propietario,
40                         @JsonProperty("qr") final String qr,
41                         @JsonProperty("propietarios") final List<String> propietarios) {
```

```
J AssetPrenda.java X J PrendaTransfer.java
src > main > java > com > bsm > chaincode > J PrendaTransfer.java
17
18     @Contract(
19         name = "PrendaTransfer",
20         info = @Info(
21             title = "PrendaTransfer contract",
22             description = "Chaincode para el registro de fabricación de prendas de lujo",
23             version = "0.0.1"))
24     @Default
25     public final class PrendaTransfer implements ContractInterface {
26
27         private final Genson genson = new Genson();
28
29         private enum PrendaTransferErrors {
30             Prenda_NOT_FOUND,
31             Prenda_ALREADY_EXISTS
32         }
33
34         @Transaction(intent = Transaction.TYPE.SUBMIT)
35         > public AssetPrenda registrarPrenda(final Context ctx, final String id, final String marca, final String categoria, final String paisFabricacion, final String propietario, final String qr, final List<String> propietarios) {
36             ...
37         }
38
39         @Transaction(intent = Transaction.TYPE.EVALUATE)
40         > public AssetPrenda imprimirPrenda(final Context ctx, final String id) {
41             ...
42         }
43
44         @Transaction(intent = Transaction.TYPE.SUBMIT)
45         > public void borrarPrenda(final Context ctx, final String id) {
46             ...
47         }
48
49         @Transaction(intent = Transaction.TYPE.SUBMIT)
50         > public String transferenciaPrenda(final Context ctx, final String id, final String newOwner) {
51             ...
52         }
53
54         @Transaction(intent = Transaction.TYPE.EVALUATE)
55         > public String listarPrendas(final Context ctx) {
56             ...
57         }
58     }
```

***IMPORTANTE:** la función borrarPrenda, del Ledger no se puede borrar, pero si del world state

Primeros test, deployamos e invocamos el chaincode a través de sus funciones

Deployamos el chaincode en la red, ejecutamos

```
./network.sh deployCC -ccn chaincode -ccp ./chaincode/ -ccl java
```

Exportamos las variables de entorno

```
export PATH=${PWD}/../bin:${PWD}:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export ORDERER_CA=${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
export CORE_PEER_TLS_ROOTCERT_FILE_ORG1=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_TLS_ROOTCERT_FILE_ORG2=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
```

Nos ponemos como la **Org1**

```
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

Registramos prenda ID 1, ID 2, ID 3

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c '{"Args":["registrarPrenda", "1", "Gucci", "Camisa", "Fabricado en la India", "2024-01-15", "DAVID GUTIERREZ", "152151XS5ESA21"]}'
```



```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c '{"Args":["registrarPrenda", "2", "Dior", "Camisa", "Fabricado en China", "2023-10-05", "PEPE PEREZ", "99W515SFESDFSD5"]}'
```



```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c '{"Args":["registrarPrenda", "3", "Loewe", "Bolso", "Fabricado en Madrid", "2023-09-25", "MARIA DIAZ", "D223DF234WAEDF"]}'
```

Imprimimos prenda

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C
mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles
$CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --
tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c
'{"Args":["imprimirPrenda", "1"]}'
```

Nos ponemos como la Org2

```
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051
```

Listamos las prendas

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C
mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles
$CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --
tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c
'{"Args":["listarPrendas"]}'
```

Borramos una prenda

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C
mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles
$CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --
tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c
'{"Args":["borrarPrenda", "2"]}'
```

Transferimos la prenda del ID 2

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C
mychannel -n chaincode --peerAddresses localhost:7051 --tlsRootCertFiles
$CORE_PEER_TLS_ROOTCERT_FILE_ORG1 --peerAddresses localhost:9051 --
tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE_ORG2 -c
'{"Args":["transferenciaPrenda", "2", "JUAN CARLOS"]}'
```

Inspeccionamos el CouchDB

http://localhost:5984/_utils/#login

User: **admin**

Pass: **adminpw**

The screenshot shows the Fauxton interface with the title "Project Fauxton - Databases". On the left is a sidebar with icons for databases, replication, users, and system. The main area lists databases with columns: Name, Size, # of Docs, and Partitions. The databases listed are: _replicator, _users, fabric_internal, mychannel_, mychannel.lifecycle, mychannel.lifecycle\$h_implicit_org_\$org1\$m\$\$p, mychannel.lifecycle\$h_implicit_org_\$org2\$m\$\$p, mychannel.lifecycle\$p_implicit_org_\$org1\$m\$\$p, mychannel_chaincode, and mychannel_lscc. All databases have 0 bytes size and 0 docs.

Registraremos una prenda y vemos como el CouchDB la almacena correctamente

The screenshot shows two Fauxton windows. The left window is titled "Project Fauxton - _all_dbs" and lists the same databases as the previous screenshot. The right window is titled "Project Fauxton - database: mychannel_chaincode" and shows a single document with the ID "1". The document details are:

```
1 {
2   "_id": "1",
3   "_rev": "1-7cf721519755ee750a1d33e4bb613919",
4   "alimentacion": "GUCCI",
5   "denominacionOrigen": "Fabricado en la India",
6   "id": "1",
7   "intermediarios": [
8     "2024-01-18"
9   ],
10  "propietario": "2024-01-18",
11  "raza": "Canta Gucci",
12  "valor": "QR 152151X55ES21",
13  "-version": "CgMBBgA=",
14 }
```

Log de la red – Hyperledger Explorer

Si no tenemos estos componentes instalados, los instalamos.

```
sudo dpkg --configure -a  
sudo apt-get install curl git docker.io docker-compose nodejs npm  
python2
```

Creamos una carpeta en la raíz **explorer** y damos permisos

```
sudo mkdir explorer  
sudo chmod -R 777 * explorer/
```

Dentro de la carpeta explorer descargamos Hyperledger Explorer

```
wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/examples/net1/config.json  
wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/examples/net1/connection-profile/test-network.json -P connection-profile  
wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/docker-compose.yaml
```

Copiamos los crypto artefactos de la red de test a la carpeta de Explorer. **Importante:** cada vez que se levanta la red, hay que copiar los artefactos de la misma.

```
sudo cp -r ~/fabric-samples/test-network/organizations/ ~/explorer/
```

```
ev-k8s@evk8s-VirtualBox:~/explorer$ ls -l  
total 16  
-rw-rw-r-- 1 ev-k8s ev-k8s 161 ene 18 18:41 config.json  
drwxrwxr-x 2 ev-k8s ev-k8s 4096 ene 18 18:41 connection-profile  
-rw-rw-r-- 1 ev-k8s ev-k8s 1582 ene 18 18:46 docker-compose.yaml  
drwxr-xr-x 7 root root 4096 ene 18 18:42 organizations  
ev-k8s@evk8s-VirtualBox:~/explorer$ ls -l organizations/  
total 32  
-rwxr-xr-x 1 root root 1655 ene 18 18:42 ccp-generate.sh  
-rwxr-xr-x 1 root root 1276 ene 18 18:42 ccp-template.json  
-rwxr-xr-x 1 root root 765 ene 18 18:42 ccp-template.yaml  
drwxr-xr-x 2 root root 4096 ene 18 18:42 cfssl  
drwxr-xr-x 2 root root 4096 ene 18 18:42 cryptogen  
drwxr-xr-x 5 root root 4096 ene 18 18:42 fabric-ca  
drwxr-xr-x 3 root root 4096 ene 18 18:42 ordererOrganizations  
drwxr-xr-x 4 root root 4096 ene 18 18:42 peerOrganizations  
ev-k8s@evk8s-VirtualBox:~/explorer$
```

Editamos el fichero de Docker compose, para ello utilizamos VIM, si no lo tenemos instalado, ejecutamos: `sudo apt install vim`

```
sudo vim docker-compose.yaml
```

Si nos fijamos tenemos dos contenedores más, uno con la BBDD del Explorer y otro con el Explorer.

SSH en el navegador

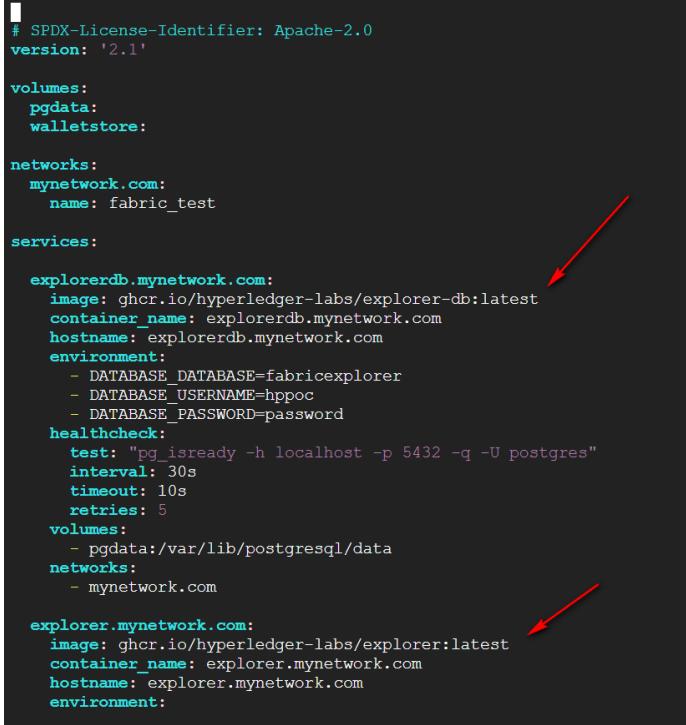
```
# SPDX-License-Identifier: Apache-2.0
version: '2.1'

volumes:
  pgdata:
  walletstore:

networks:
  mynetwork.com:
    name: fabric_test

services:
  explorerdb.mynetwork.com:
    image: ghcr.io/hyperledger-labs/explorer-db:latest
    container_name: explorerdb.mynetwork.com
    hostname: explorerdb.mynetwork.com
    environment:
      - DATABASE_DATABASE=fabricexplorer
      - DATABASE_USERNAME=hppoc
      - DATABASE_PASSWORD=password
    healthcheck:
      test: "pg_isready -h localhost -p 5432 -q -U postgres"
      interval: 30s
      timeout: 10s
      retries: 5
    volumes:
      - pgdata:/var/lib/postgresql/data
    networks:
      - mynetwork.com

  explorer.mynetwork.com:
    image: ghcr.io/hyperledger-labs/explorer:latest
    container_name: explorer.mynetwork.com
    hostname: explorer.mynetwork.com
    environment:
```



Reemplazamos lo que hay en el apartado de volúmenes por esto:

- ./config.json:/opt/explorer/app/platform/fabric/config.json
- ./connection-profile:/opt/explorer/app/platform/fabric/connection-profile
- ./organizations:/tmp/crypto
- walletstore:/opt/explorer/wallet

Para editar el fichero VIM:

- i = modo editar
- ctrl+c :wq! = guardar
- dd = eliminar línea

Arrancamos Hyperledger Explorer

```
sudo docker-compose up -d
```

Iniciamos Hyperledger Explorer:

<http://localhost:8080/#/login>

```
"id": "exploreradmin",
"password": "exploreradminpw"
```

Project Fauxton - database Hyperledger Explorer +

localhost:8080/#

HYPERLEDGER EXPLORER

DASHBOARD NETWORK BLOCKS TRANSACTIONS CHAINCODES CHANNELS mychannel

Search by Txn Hash / Block
Txn Hash

BLOCKS 22 **TRANSACTIONS** 22 **NODES** 2 **CHAINCODES** 1

Peer Name	Status
peer0.org1.example.com:7051	●
peer0.org2.example.com:9051	●
orderer.example.com:7050	●

BLOCKS / HOUR BLOCKS / MIN TX / HOUR TX / MIN

Chaincode Details

Info:

- root: {} 6 keys
- license: {} 1 key
- description: "Chaincode para el registro de fabricación de prendas de lujo"
- termsOfService: ""
- title: "PrendaTransfer contract"
- version: "0.0.1"
- contact: {} 0 keys

Contracts:

- root: {} 2 keys
- org.hyperledger.fabric: {} 3 keys
- PrendaTransfer: {} 3 keys

Components:

- root: {} 1 key
- schemas: {} 1 key

NETWORK DASHBOARD BLOCKS TRANSACTIONS CHAINCODES CHANNELS mychannel

Peer Name	Request Url	Peer Type	MSPID	High	Low	Unsigned
peer0.org1.example.co...	peer0.org1.example.co...	PEER	Org1MSP	0	22	true
peer0.org2.example.co...	peer0.org2.example.co...	PEER	Org2MSP	0	22	true
orderer.example.com:7...	orderer.example.com:7...	ORDERER	OrdererMSP	-	-	-

Para parar el Explorer ejecutamos:

```
sudo docker-compose down -v
```

Log de la red – Prometheus y Grafana

A continuación, dentro la carpeta **fabric-samples/test-network/prometheus-grafana** hay que modificar el valor de la versión de Docker del **docker-compose.yaml** que está dentro de la carpeta a la 3.7

```
vim docker-compose.yaml
```

```
version: '3.7'

volumes:
  prometheus_data: {}
  grafana_storage: {}

services:
  prometheus:
    image: prom/prometheus:v2.32.1
    container_name: prometheus
    volumes:
      - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
      - prometheus_data:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/usr/share/prometheus/console_libraries'
      - '--web.console.templates=/usr/share/prometheus/consoles'
    ports:
      - "9090:9090"

  grafana:
    image: grafana/grafana:8.3.4
```

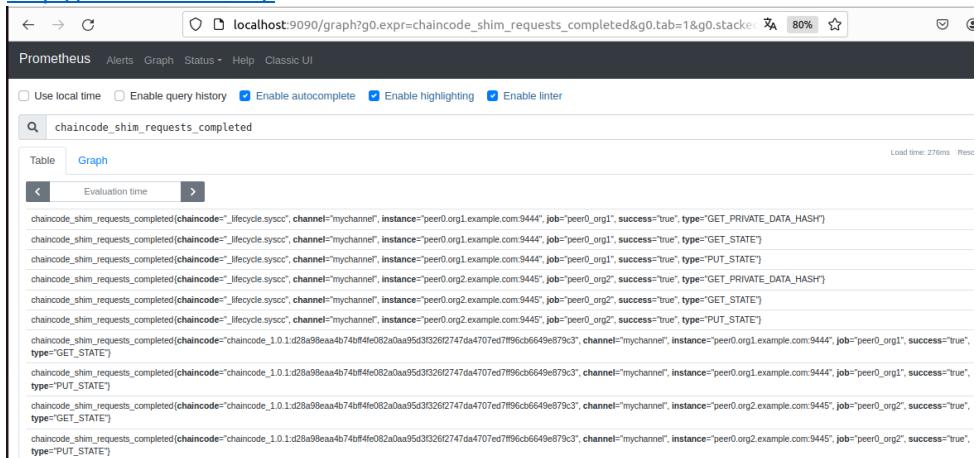
*Para editar el archivo pulsamos i y para guardar Ctrl+C y escribimos :wq

A continuación, ejecutamos el siguiente comando para levantar las herramientas.

```
sudo docker-compose up -d
```

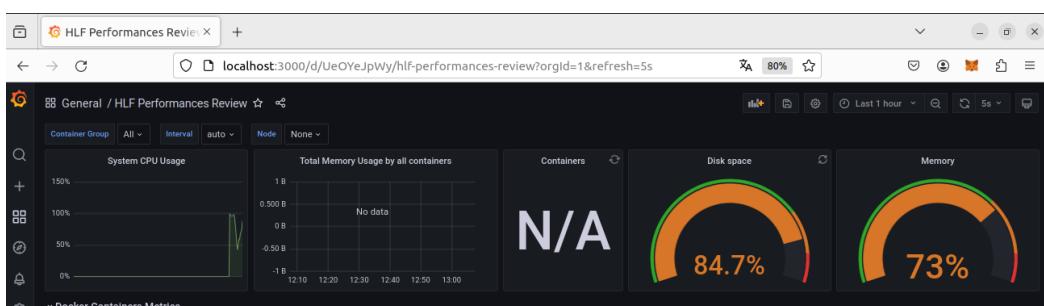
Abrimos en el navegador Prometheus para ver los logs de la red.

<http://localhost:9090/>



Abrimos los logs con Grafana: <http://localhost:3000/login/>

User: **admin** – Pass: **admin** – New Pass: **admin1234**



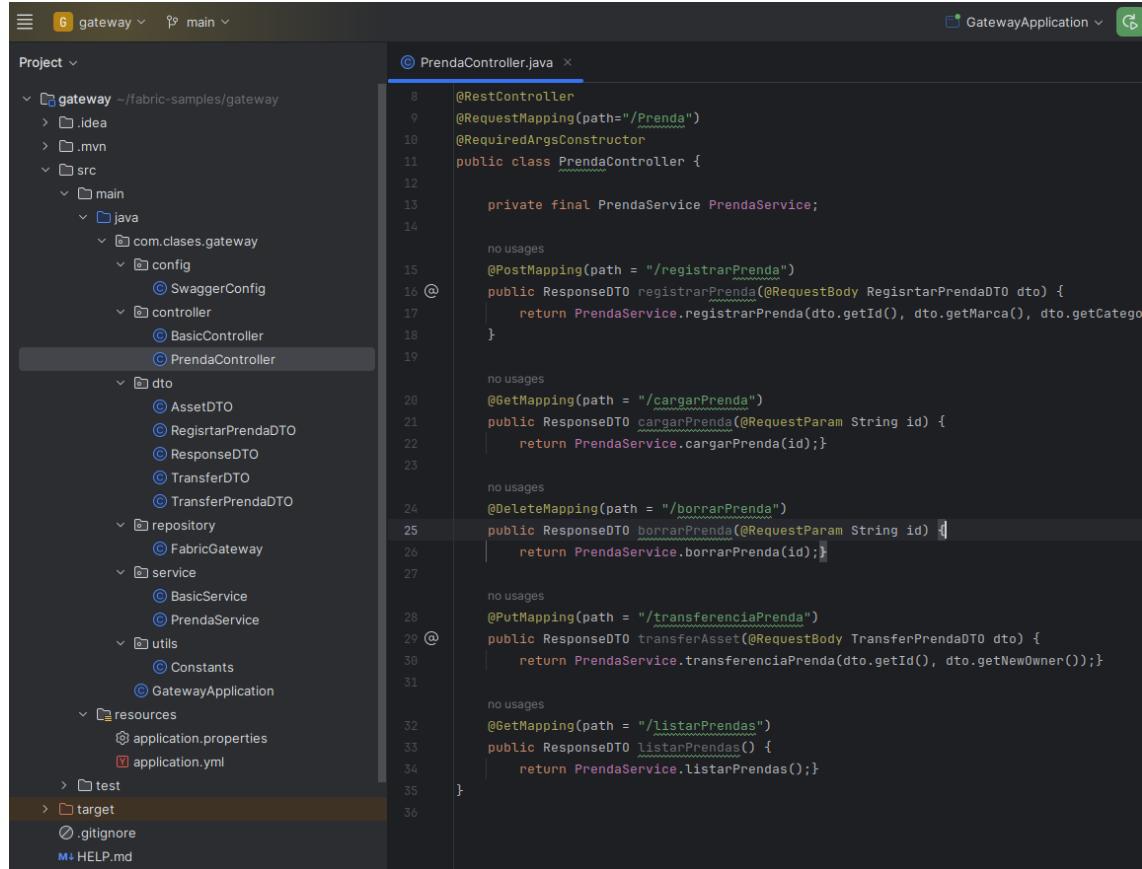
Creación de una Api-Rest para Backend y Frontend con Swagger

En la carpeta **fabric-samples/gateway** se incluye un proyecto que comunica con Hyperledger mediante Api-Rest y expone los métodos del chaincode en microservicios a través del frontal con Swagger, que es un módulo de Spring Web que permite crear una interfaz web para comunicarnos con el chaincode de Hyperledger Fabric.

Utilizaremos las herramientas Maven y Gradle para desarrollar Hyperledger Fabric Cliente API for Java. Para ello inicializamos un nuevo proyecto con Start Spring de Maven con lenguaje Java.

<https://start.spring.io/>

Creamos el controler con las funciones que vamos a invocar del chaincode



The screenshot shows a Java-based development environment with the following details:

- Project View:** Shows the project structure under "gateway". Key components include:
 - src/main/java/com.clases.gateway/controller:** Contains the **PrendaController**.
 - src/main/java/com.clases.gateway/dto:** Contains **AssetDTO**, **RegistrarPrendaDTO**, **ResponseDTO**, **TransferDTO**, and **TransferPrendaDTO**.
 - src/main/java/com.clases.gateway/repository:** Contains **FabricGateway**.
 - src/main/java/com.clases.gateway/service:** Contains **BasicService** and **PrendaService**.
 - src/main/java/com.clases.gateway/utils:** Contains **Constants**.
 - src/main/java/com.clases.gateway/GatewayApplication:** The main application class.
- Code Editor:** Displays the **PrendaController.java** file with the following code:

```
8  @RestController
9  @RequestMapping(path = "/Prenda")
10 @RequiredArgsConstructor
11 public class PrendaController {
12
13     private final PrendaService PrendaService;
14
15     no usages
16     @PostMapping(path = "/registrarPrenda")
17     public ResponseDTO registrarPrenda(@RequestBody RegistrarPrendaDTO dto) {
18         return PrendaService.registrarPrenda(dto.getId(), dto.getMarca(), dto.getCatego
19     }
20
21     no usages
22     @GetMapping(path = "/cargarPrenda")
23     public ResponseDTO cargarPrenda(@RequestParam String id) {
24         return PrendaService.cargarPrenda(id);
25     }
26
27     no usages
28     @DeleteMapping(path = "/borrarPrenda")
29     public ResponseDTO borrarPrenda(@RequestParam String id) {
30         return PrendaService.borrarPrenda(id);
31     }
32
33     no usages
34     @PutMapping(path = "/transferenciaPrenda")
35     public ResponseDTO transferAsset(@RequestBody TransferPrendaDTO dto) {
36         return PrendaService.transferenciaPrenda(dto.getId(), dto.getNewOwner());
37     }
38
39     no usages
40     @GetMapping(path = "/listarPrendas")
41     public ResponseDTO listarPrendas() {
42         return PrendaService.listarPrendas();
43     }
44 }
```

Creamos los servicios del controller y será el encargado de conectar a través de la variable Gateway con Hyperledger Fabric a través del canal

```

1 package com.clases.gateway.service;
2
3 import ...
4
5 2 usages
6 @Service
7 @RequiredArgsConstructor
8 public class PrendaService {
9
10     public final FabricGateway fabricGateway;
11
12     private final Gson gson = new GsonBuilder().setPrettyPrinting().create();
13
14     Usage
15     public ResponseDTO registrarPrenda(final String id, final String marca, final String categoria, final String paisFabricacion) {
16
17         ResponseDTO response = new ResponseDTO();
18
19         try (var gateway = fabricGateway.createConnection().connect()) {
20             var network = gateway.getNetwork(Constants.CHANNEL_NAME);
21
22                 // Get the smart contract from the network.
23                 Contract contract = network.getContract(Constants.CHAINCODE_PRENDA_NAME);
24                 byte[] result = contract.submitTransaction("registrarPrenda", id, marca, categoria, paisFabricacion, fechaFabricacion);
25
26                 response.setCode("0");
27                 response.setData(prettyJson(result));
28             } catch (Exception e) {
29                 response.setCode("1");
30                 response.setData(e.getMessage());
31             }
32
33         return response;
34     }
35
36     ...
37
38 }
39
40
41
42
43
44
45

```

Creamos los DTO que nos ayudará en el frontal a la hora de crear

```

1 package com.clases.gateway.dto;
2
3 import io.swagger.annotations.ApiModelProperty;
4 import lombok.Data;
5
6 1 usage
7 @Data
8 public class RegistrarPrendaDTO {
9
10     @ApiModelProperty(value = "Prenda ID", example = "1")
11     String id;
12
13     @ApiModelProperty(value = "Marca", example = "Gucci")
14     String marca;
15
16     @ApiModelProperty(value = "Categoria", example = "Bolso")
17     String categoria;
18
19     @ApiModelProperty(value = "Pais de fabricacion", example = "Italia")
20     String paisFabricacion;
21
22     @ApiModelProperty(value = "Fecha de fabricacion", example = "2024-01-19")
23     String fechaFabricacion;
24
25     @ApiModelProperty(value = "Propietario", example = "Pepe")
26     String propietario;
27
28     @ApiModelProperty(value = "Qr", example = "5w15s8w51599wa")
29     String qr;
30 }
31

```

En el archivo Constants.java definimos los parámetros necesarios para comunicar con Hyperledger Fabric. Importante: definir bien la ruta de comunicación

```

Project ~ /fabric-samples/gateway
> .idea
> .mvn
src
  main
    java
      com.clases.gateway
        config
          SwaggerConfig
        controller
          BasicController
          PrendaController
        dto
          AssetDTO
          RegistrarPrendaDTO
          ResponseDTO
          TransferDTO
          TransferPrendaDTO
        repository
          FabricGateway
        service
          BasicService
          PrendaService
        utils
          Constants
          GatewayApplication
      resources
        application.properties
        application.yml
    test
  target

```

```

Constants.java x
3 > import ...
5
34 usages
6 public final class Constants {
7
8     no usages
9     private Constants(){}
10
11     1 usage
12     public static final String MSP_ID_ORG1 = "Org1MSP";
13     12 usages
14     public static final String CHANNEL_NAME = "mychannel";
15     7 usages
16     public static final String CHAINCODE_BASIC_NAME = "basic";
17     5 usages
18     public static final String CHAINCODE_PRENDA_NAME = "chaincode";
19
20     // Path to crypto materials.
21     3 usages
22     public static final Path CRYPTO_PATH_ORG1 = Paths.get(first: "../test-network/organizations/peerOrganizations/org1.example.com/certs/chaincode.cryptostore");
23     // Path to user certificate.
24     1 usage
25     public static final Path CERT_PATH_ORG1 = CRYPTO_PATH_ORG1.resolve(Paths.get(first: "users/User1@org1.example.com/msp/signcert"));
26     // Path to user private key directory.
27     1 usage
28     public static final Path KEY_DIR_PATH_ORG1 = CRYPTO_PATH_ORG1.resolve(Paths.get(first: "users/User1@org1.example.com/msp/keystore"));
29     // Path to peer tls certificate.
30     1 usage
31     public static final Path TLS_CERT_PATH_ORG1 = CRYPTO_PATH_ORG1.resolve(Paths.get(first: "peers/peer0.org1.example.com/tls/ca.crt"));
32
33     // Gateway peer end point.
34     1 usage
35     public static final String PEER_ENDPOINT_ORG1 = "localhost:7051";
36
37     1 usage

```

Comprobamos que tenemos todas las dependencias exigidas instaladas

```

Project ~ gateway ~ main
src
  main
    java
      com.clases.gateway
        config
          SwaggerConfig
        controller
          BasicController
          PrendaController
        dto
          AssetDTO
          RegistrarPrendaDTO
          ResponseDTO
          TransferDTO
          TransferPrendaDTO
        repository
          FabricGateway
        service
          BasicService
          PrendaService
        utils
          Constants
          GatewayApplication
      resources
        application.properties
        application.yml
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml

```

```

pom.xml (gateway) x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.7.18-SNAPSHOT</version>
9     <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>com.clases</groupId>
12    <artifactId>gateway</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>gateway</name>
15    <description>Demo project for Spring Boot</description>
16    <properties>
17      <java.version>11</java.version>
18    </properties>
19    <dependencies>
20      <dependency>
21        <groupId>org.springframework.boot</groupId>
22        <artifactId>spring-boot-starter</artifactId>
23      </dependency>
24
25      <dependency>
26        <groupId>org.springframework.boot</groupId>
27        <artifactId>spring-boot-starter-test</artifactId>
28        <scope>test</scope>
29      </dependency>
30
31      <dependency>
32        <groupId>org.springframework.boot</groupId>
33        <artifactId>spring-boot-starter-web</artifactId>
34      </dependency>
35
36    </dependencies>
37  </project>

```

Test frontal a través de Swagger

Corremos la aplicación para que compile, despliegue servicios y nos abra el frontal con Swagger a través del puerto 8080

The screenshot shows a Java IDE interface with the following details:

- Project View:** Shows the project structure under "src/main/java/com.classes.gateway". The "PrendaController" class is selected.
- Code Editor:** Displays the code for the PrendaController.java file, which contains annotations for various HTTP methods and their corresponding service implementations.
- Terminal:** Shows the command-line output of the application's startup process. It includes logs from Spring Boot, Tomcat, and the application itself, indicating the successful initialization of the application at port 8080.

Cargamos Swagger a través del navegados y hacemos los test unitarios con el smartcontract
<http://localhost:8080/swagger-ui/#>

The screenshot shows the Swagger UI interface for the application. Key elements include:

- Header:** Shows the URL "localhost:8080/swagger-ui/#".
- Swagger Logo:** The "Swagger" logo with the text "Supported by SMARTBEAR".
- Api Documentation 1.0:** The main title of the API documentation.
- Base URL:** "localhost:8080/" and "http://localhost:8080/v2/api-docs".
- Basic Controller:** A section for the "basic-controller".
- Prenda Controller:** A section for the "prenda-controller".
- Operations:** A list of API operations grouped by controller:
 - DELETE /Prenda/borrarPrenda:** "borrarPrenda"
 - GET /Prenda/cargarPrenda:** "cargarPrenda"
 - GET /Prenda/listarPrendas:** "listarPrendas"
 - POST /Prenda/registrarPrenda:** "registrarPrenda"
 - PUT /Prenda/transferenciaPrenda:** "transferAsset"

Registramos una nueva prenda y le asignamos el propietario

POST /Prenda/registrarPrenda registrarPrenda

Parameters

Name	Description
dto * required object (body)	dto Edit Value Model

```
{
  "categoria": "Bolso",
  "fechaFabricacion": "2024-01-19",
  "id": 4,
  "marca": "ARMANI",
  "paisFabricacion": "ITALIA",
  "propietario": "CARMEN",
  "qr": "82w95s9dfg954"
}
```

Cancel

Parameter content type
application/json

Responses

Curl

```
curl -X POST "http://localhost:8080/Prenda/registrarPrenda" -H "accept: */*" -H "Content-Type: application/json" -d '{"id": 4, "marca": "ARMANI", "paisFabricacion": "ITALIA", "propietario": "CARMEN", "qr": "82w95s9dfg954"}'
```

Request URL
http://localhost:8080/Prenda/registrarPrenda

Server response

Code	Details
200	Response body

```
{
  "code": "0",
  "data": "\n  \"id\": \"4\",\\n  \"propietario\": \"CARMEN\"\n\""
}
```

Comprobamos también los datos a través de couchDB

← → ⌂ localhost:5984/_utils/#data

mychannel_chaincode ➤ 4

Save Changes Cancel

```

1 + []
2   "_id": "4",
3   "_rev": "1-07457def87950e10fef12c442667169d",
4   "categoria": "Bolso",
5   "fechaFabricacion": "2024-01-19",
6   "id": "4",
7   "marca": "ARMANI",
8   "paisFabricacion": "ITALIA",
9   "propietario": "CARMEN",
10  "propietarios": [
11    "CARMEN"
12  ],
13  "qr": "82w95s9dfg954",
14  "~version": "CgMBEQ4="
15  []

```

Probamos a transferir la prenda que acabamos de crear

PUT /Prenda/transferenciaPrenda transferAsset

Parameters

Name	Description
dto * required	dto
object (body)	Edit Value Model

```
{  
  "id": 4,  
  "newOwner": "MONICA"  
}
```

Responses

Curl

```
curl -X PUT "http://localhost:8080/Prenda/transferenciaPrenda" -H "accept: /*" -H "Content-Type: application/json" -d "{ \"id\": 4, \"newOwner\": \"MONICA\"}"
```

Request URL

```
http://localhost:8080/Prenda/transferenciaPrenda
```

Server response

Code Details

200 Response body

```
{  
  "code": "0",  
  "data": "Nuevo propietario MONICA"  
}
```

Comprobamos a través del couchDB los propietarios que ha tenido esta prenda

localhost:5984/_utils/#database/mychannel_chaincode/4

mychannel_chaincode > 4

Save Changes Cancel

```
1 v []
2   "_id": "4",
3   "_rev": "2-c6b3b2c64e9dfde2b4695d5116b81154",
4   "categoria": "Bolso",
5   "fechaFabricacion": "2024-01-19",
6   "id": "4",
7   "marca": "ARMANI",
8   "paisFabricacion": "ITALIA",
9   "proprietario": "MONICA",
10  "proprietarios": [
11    "CARMEN",
12    "MONICA"
13  ],
14  "qr": "82w95s9dfg954",
15  "-version": "CgMBEgA="
```

Testeamos cargar prenda

GET /Prenda/cargarPrenda cargarPrenda

Parameters

Name	Description
Id * required string (query)	id

4

Responses

Curl

```
curl -X GET "http://localhost:8080/Prenda/cargarPrenda?id=4" -H "accept: */*
```

Request URL

```
http://localhost:8080/Prenda/cargarPrenda?id=4
```

Server response

Code	Details
200	Response body

```
{
  "code": "0",
  "data": "{\n    \"id\": \"4\",\n    \"propietario\": \"MONICA\"\n  }"
}
```

Testeamos listar todas las prendas

GET /Prenda/listarPrendas listarPrendas

Parameters

No parameters

Responses

Curl

```
curl -X GET "http://localhost:8080/Prenda/listarPrendas" -H "accept: */*
```

Request URL

```
http://localhost:8080/Prenda/listarPrendas
```

Server response

Code	Details
200	Response body

```
{
  "code": "0",
  "data": "[{\n    \"categoria\": \"Camisa\", \n    \"fechaFabricacion\": \"2024-01-15\", \n    \"id\": \"1\", \n    \"marca\": \"Gucci\", \n    \"paisFabricacion\": \"Fabricado en la India\", \n    \"propietario\": \"JUAN GONZALEZ\", \n    \"propietarios\": [\n        \"DAVID GUTIERREZ\", \n        \"JUAN GONZALEZ\"\n    ], \n    \"qr\": \"15215XSE5A21\"}, {\n    \"categoria\": \"Bolso\", \n    \"fechaFabricacion\": \"2023-12-15\", \n    \"id\": \"2\", \n    \"marca\": \"Dior\", \n    \"paisFabricacion\": \"Maria\", \n    \"propietario\": \"Espana\", \n    \"propietarios\": [\n        \"Espana\", \n        \"qr\": \"8954bddd4w\"], \n    \"qr\": \"0923-09-01\", \n    \"id\": \"3\", \n    \"marca\": \"GUESS\", \n    \"paisFabricacion\": \"JORDANA\", \n    \"propietario\": \"SOFIA\", \n    \"propietarios\": [\n        \"ITALIA\", \n        \"SOFIA\", \n        \"qr\": \"9w6alj3685s\"], \n    \"qr\": \"2024-01-19\", \n    \"id\": \"4\", \n    \"marca\": \"ARMANI\", \n    \"paisFabricacion\": \"ITALIA\", \n    \"propietario\": \"MONICA\", \n    \"propietarios\": [\n        \"CARMEN\", \n        \"MONICA\"], \n    \"qr\": \"82w959d9f954\"}]"
}
```

Download