

ArdKalic

CFGS DAW

CURSO 2020/2021

ADRIAN MATEOS CEREZO, MARIO REBOLLO GARCIA

COLEGIO CALASANZ SALAMANCA



Contenido

1. Introducción.....	3
2. Descripción	3
3. Tecnologías escogidas y justificación.....	6
4. Diseño de la aplicación.....	7
4.1. Diagramas y definición de casos de uso	7
4.2. Diagramas de clases.....	21
4.3. Modelo de dominio.....	26
5. Arquitectura de la aplicación	27
5.1. Estructura del proyecto.....	27
5.2. Librerías externas utilizadas.....	28
6. Manual de despliegue	29



1. Introducción

Nuestra idea para el proyecto consistirá en un e-commerce de hardware para PCs domésticos con diversas funcionalidades de cara al usuario que quiera ser nuestro cliente.

El nombre de nuestro proyecto será “ArdKalic”

En nuestra página web el usuario tendrá acceso a un amplio catálogo de componentes de PC para que ellos mismos se puedan montar su PC en casa o ampliar las características del que ya tenían.

Cada usuario se podrá registrar en la aplicación y comprar los productos que desee.

2. Descripción

Descripción de la aplicación

Usuario:

Lo primero que se encontrará el usuario al entrar en nuestro sitio web será la página de entrada en la que se encontrará un resumen de los productos de nuestra tienda, algunas recomendaciones y una sugerencia a registrarse como nuevo usuario.

Usuario no registrado: Solo podrá mirar el catálogo y precios de los productos que se encuentran en esta tienda online y hacer la compra de producto que le interese. Además, también podrá contactar con nuestro equipo de soporte para cualquier duda si nos proporciona su email.

Usuario registrado: Tendrá acceso completo a la web y podrá comprar cualquier producto que se encuentre en stock. Además, tendrá su propio perfil donde podrá consultar su información personal y la que nos quiera dar a nosotros como empresa. También puede eliminar su cuenta cuando lo desee.

(A parte también tendrá los mismos permisos que el usuario no registrado)



La página también dispondrá de un buscador para que el usuario pueda encontrar un producto específico rápidamente.

En la parte superior de la página se situará un menú que ayudará al usuario a navegar por la página sin ningún tipo de problema. Será claro y fácil de usar.

Cuando se accede al apartado de un producto, este tendrá una descripción y los detalles sobre el precio y la cantidad de stock del mismo.

Habrà un apartado para contactar con nosotros y otro que enseña quienes somos y una pequeña descripción de nuestra empresa.

En el pie de la página se podrán encontrar nuestras redes sociales.

Administradores:

Nosotros como administradores nos encargaremos de añadir o quitar stock y además de contactar con mensajes personalizados a clientes que específicos.

Cuando un administrador inicia sesión en la página se le mostrará y dará acceso a más apartados que un usuario normal no puede ver ni acceder. Estos apartados tienen se utilizan para que los administradores cumplan las funciones mencionadas anteriormente.

Sistema:

El sistema de la página se encargará de todas las gestiones internas de la aplicación, así como:

- Validar usuarios, compras, etc.
- Generar nuevos perfiles.

Interfaz:

Nuestra web tendrá un esquema de interfaz igual o muy similar en todas las páginas de la misma.



El menú superior de la página tendrá unos botones y un desplegable que se activaran cuando se haga clic encima de ellos. A través de este menú (el cual estará disponible en todas las partes de la web) el usuario podrá navegar por toda nuestra tienda sin ningún tipo de problema.

Las imágenes de la web contendrán vínculos hacia los productos que se muestren.

En los apartados de los productos habrá un sistema de filtros para que el usuario pueda encontrar un producto a su medida (precios, marcas, ...). el usuario podrá usar los filtros proporcionados por la página para encontrar su producto según su criterio.

Para seleccionar un objeto a comprar habrá un botón de añadir al carrito, que acumulará el producto en una lista de la compra. En esta lista habrá un botón para finalizar la compra que hará la compra efectuada.

Estarán disponibles siempre los botones de perfil de usuario, carrito de compra y ayuda al cliente.

Opcional: habrá un buscador en la parte superior de la página para que el usuario encuentre lo que quiere con mayor facilidad.



3. Tecnologías escogidas y justificación

- **Spring:**

Spring MVC es un framework para el desarrollo de aplicaciones web bajo el patrón modelo-vista-controlador en la plataforma Java EE.

Hemos decidido usar Spring porque lo bueno que tiene Spring es su modularidad. Puedes combinar Spring con cualquier librería o framework, no estás atado a usar frameworks relacionados con Spring.

- **Angular:**

Para la parte de Angular hemos escogido el framework de Angular debido a que durante nuestro desarrollo en la practicas de empresa es el framework que hemos utilizado y por ello ya estamos más familiarizados.

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página (SPA).

Con la librería Routing de Angular controlaremos de una forma óptima y eficiente la navegación por las vistas.

- **MySQL:**

Es el gestor de base de datos relacional que utilizaremos en nuestro proyecto, porque durante nuestra formación del grado de formación profesional es el que hemo utilizado y mayor conocimiento tenemos.

4. Diseño de la aplicación

4.1. Diagramas y definición de casos de uso

Nombre: Identificación de usuario

Id: PM01

Actores: Cliente registrado, Administrador

Descripción: El usuario introduce sus datos para autenticarse en la plataforma

Precondiciones: Está registrado el usuario

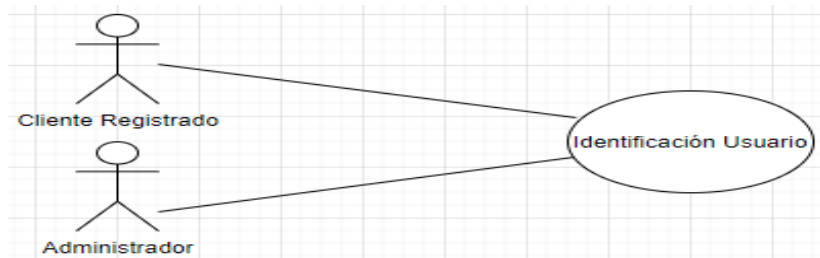
Pasos:

1. El sistema pedirá los datos del usuario por pantalla
2. El usuario rellena los campos que se piden.
3. El sistema comprueba que los datos existen y se validan correctamente.
4. El usuario queda identificado con rol en la página.

Postcondiciones: El sistema mostrará la correcta autenticación del usuario.

Alternativa:

1. No se validan los datos introducidos.
2. El sistema mostrará una alerta del error ocurrido y te dará dos opciones; registrarte como nuevo usuario o volver a introducir los datos por si el usuario los introdujo mal la vez anterior.





Nombre: Búsqueda de productos

Id: PM02

Actores: Cliente anónimo, Cliente registrado, Administrador

Descripción: El usuario busca productos en base a sus criterios o filtros proporcionados.

Precondiciones: Hay productos en la web (en la base de datos).

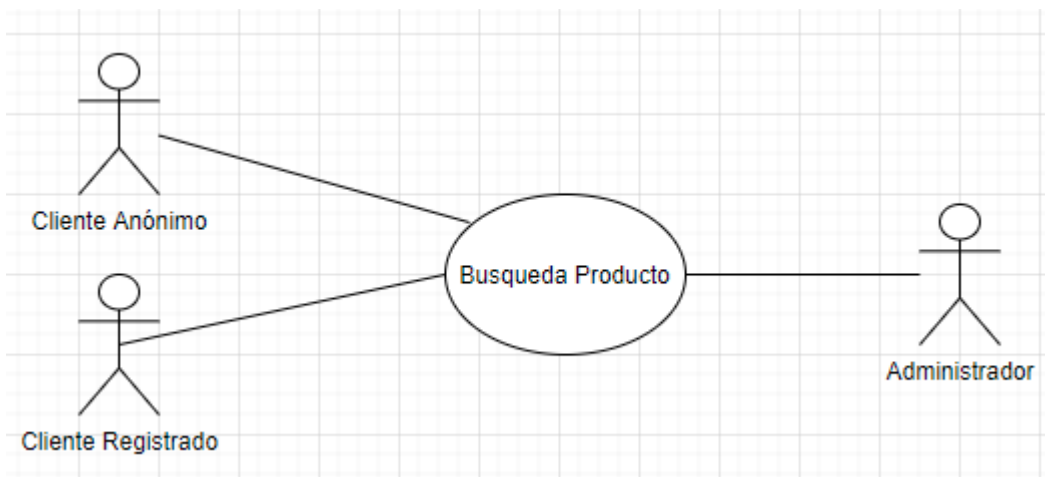
Pasos:

1. El sistema muestra en la página filtros de búsqueda junto a ofertas.
2. El usuario seleccionará los filtros según su criterio.
3. El proceso finaliza.

Postcondiciones: El sistema devuelve los productos con características similares a las seleccionadas por el usuario.

Alternativa:

1. No hay productos con las características seleccionadas.
2. Saldrá un mensaje indicando que no existen productos.





Nombre: Registrar usuario

Id: PM03

Actores: Cliente anónimo

Descripción: El usuario desea quedar registrado en la web.

Precondiciones: No está registrado.

Pasos:

1. El usuario selecciona la opción de registrarse.
2. El sistema pide datos de registro al usuario.
3. El usuario introduce sus datos.
4. El sistema guarda esos datos en la base de datos.

Postcondiciones: Se devuelve un mensaje de éxito en el registro.

Alternativa:

1. El usuario ya estaba registrado



Nombre: Añadir producto a la cesta

Id: PM04

Actores: Cliente registrado, Cliente anónimo

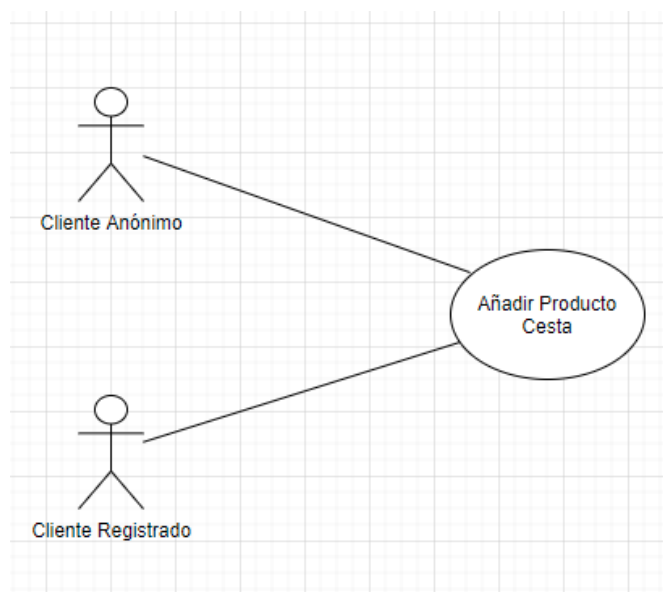
Descripción: El cliente selecciona un producto para meterlo en su cesta de compra.

Precondiciones: Hay productos seleccionables.

Pasos:

1. El cliente hace clic en el producto que le interesa.
2. El sistema añade el producto seleccionado a la cesta correspondiente al usuario.

Postcondiciones: El sistema muestra el contenido de la cesta.



Nombre: Eliminar de la cesta

Id: PM05

Actores: Cliente registrado, Cliente anónimo

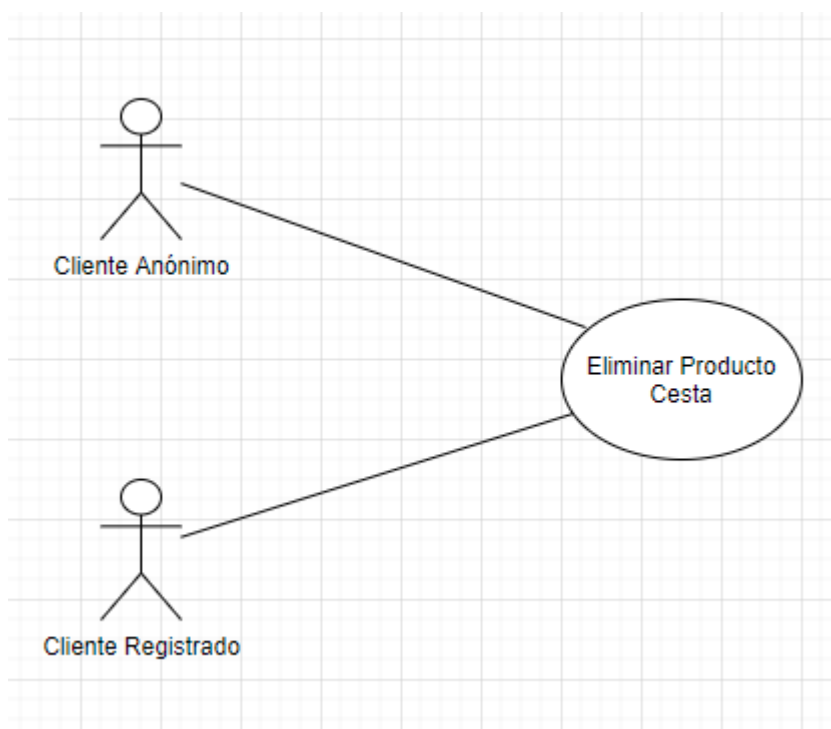
Descripción: El cliente saca un producto de su cesta.

Precondiciones: Hay productos en la cesta.

Pasos:

1. El cliente hace clic en el botón de quitar producto de la cesta.
2. El sistema elimina el producto seleccionado de la cesta del usuario.

Postcondiciones: El sistema muestra el contenido de la cesta.





Nombre: Ver cesta

Id: PM06

Actores: Cliente registrado, Cliente anónimo

Descripción: El cliente abre la cesta de compra para revisar los productos que quiere comprar.

Precondiciones: Hay productos en la cesta.

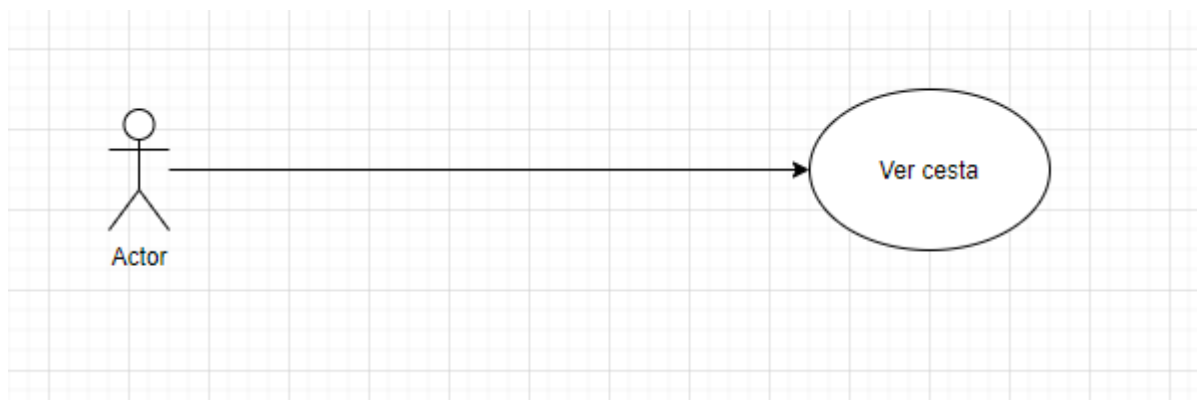
Pasos:

1. El cliente selecciona la opción cesta.
2. Se mostrará un cuadro con los productos añadidos.

Postcondiciones: El sistema muestra el contenido de la cesta.

Alternativa:

1. No hay productos en la cesta
2. El sistema avisa de que no hay productos en la cesta





Nombre: Compra

Id: PM07

Actores: Cliente registrado, Cliente anónimo

Descripción: El cliente realiza un nuevo pedido con los productos que tenga en su cesta.

Precondiciones: Hay productos a la cesta.

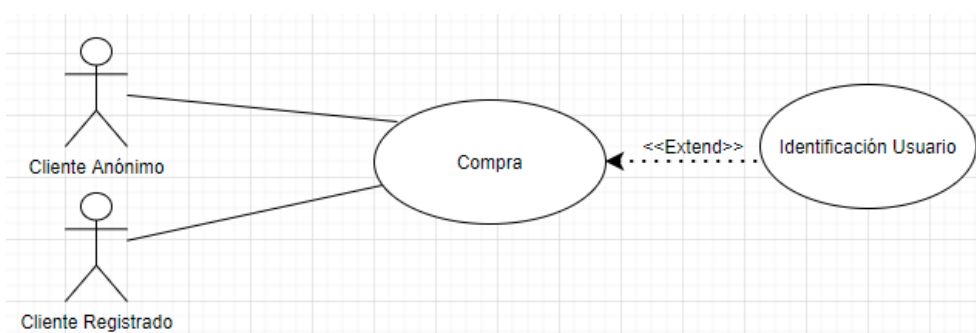
Pasos:

1. El cliente accede a su cesta de la compra y selecciona finalizar compra.
2. El sistema pide confirmación de la cesta.
3. El cliente confirma.
4. El sistema avisara de que la compra ha tenido éxito.

Postcondiciones: El sistema mostrará la correcta finalización del pedido.

Alternativa:

1. La cesta está vacía.
2. El sistema informa al cliente de que no hay productos seleccionados para su compra.





Nombre: Alta producto

Id: PM08

Actores: Administrador

Descripción: Se quiere ofrecer un nuevo producto en la tienda.

Precondiciones: El usuario tiene permisos de administrador.

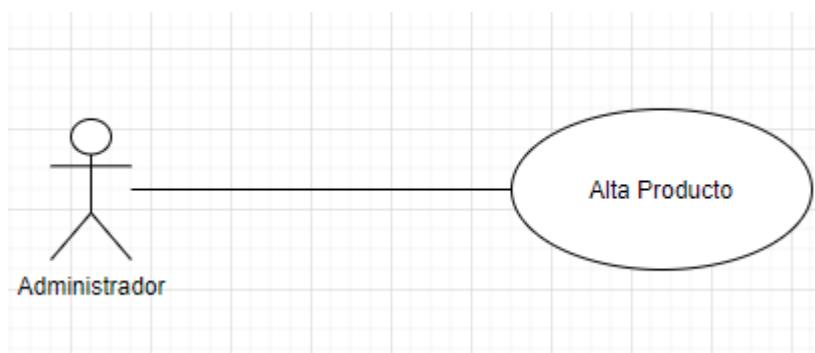
Pasos:

1. El administrador pulsara la opción de añadir nuevo producto.
2. Se cargará una vista donde se pedirán los datos del producto que se quiera introducir como nuevo producto.
3. El administrador rellena los datos del producto.
4. Se registra el producto con sus datos en la base de datos.

Postcondiciones: El producto aparece disponible para su compra en la página web.

Alternativa:

1. El producto ya existe.





Nombre: Baja producto

Id: PM09

Actores: Administrador

Descripción: Se quiere quitar un producto de la tienda para que ya que no se venda de cara al público.

Precondiciones: Existe el producto que se quiere eliminar.

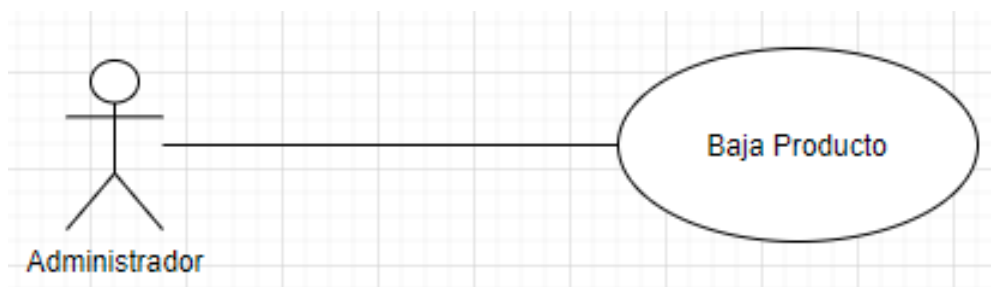
Pasos:

1. El administrador selecciona la opción de eliminar producto que aparece en cada producto.
2. El sistema devuelve una confirmación de eliminación.
3. El administrador elije si confirmar la eliminación o cancelarla.
4. En caso de que se confirme el sistema elimina todos los datos de ese producto de la base de datos y desaparece de la tienda.

Postcondiciones: El sistema mostrará que se ha eliminado correctamente.

Alternativa:

1. El administrador cancela la eliminación.
2. El producto que se quiere eliminar no existe desde un principio.





Nombre: Modificar producto

Id: MP10

Actores: Administrador

Descripción: Se desea cambiar los datos de un producto de cara al público.

Precondiciones: Existen productos.

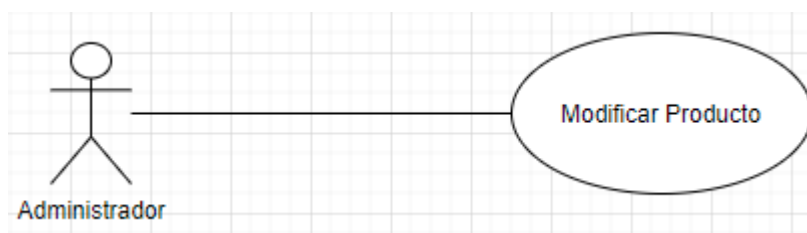
Pasos:

1. El administrador selecciona la opción de editar producto.
2. Se muestra una vista con campos editables del producto.
3. El administrador modifica lo que desea y guarda los cambios.
4. El sistema actualiza la base de datos con los nuevos datos.

Postcondiciones: Se mostrará el producto actualizado en la tienda correctamente.

Alternativa:

1. No existe el producto.





Nombre: Logout

Id: MP11

Actores: Administrador, Cliente registrado

Descripción: El usuario desea desconectarse de la página.

Precondiciones: El usuario esta iniciado sesión en la página.

Pasos:

1. El usuario selecciona la opción de cerrar sesión.

Postcondiciones: El sistema redirige al usuario a la pantalla de identificación.

Alternativa:

1. El usuario no estaba iniciado sesión desde el principio.





Nombre: Ver datos cuenta

Id: MP12

Actores: Administrador, Cliente registrado

Descripción: El usuario quiere ver los datos de su cuenta.

Precondiciones: El usuario esta iniciado sesión en la página.

Pasos:

1. El usuario selecciona la opción "Mi cuenta".
2. El sistema devuelve los datos de la cuenta del usuario

Postcondiciones: El sistema mostrará una vista con los datos personales del usuario.

Alternativa:

1. El usuario no estaba iniciado sesión desde el principio.



Nombre: Modificar datos cuenta

Id: MP13

Actores: Administrador, Cliente registrado

Descripción: El usuario quiere modificar los datos de su cuenta.

Precondiciones: El usuario esta iniciado sesión en la página.

Pasos:

1. El usuario selecciona la opción “Mi cuenta”.
2. Una vez en la vista con sus datos, el usuario seleccionará la opción “Modificar datos”.
3. La vista será editable en ese momento para que el usuario cambie lo que desee.
4. Para guardar los cambios el usuario deberá seleccionar la opción “Hecho”

Postcondiciones: Al acabar la modificación el sistema devolverá la vista de los datos de la cuenta.

Alternativa:

1. El usuario no estaba iniciado sesión desde el principio o no está registrado.



Nombre: Eliminar cuenta

Id: MP14

Actores: Administrador, Cliente registrado

Descripción: El usuario desea deshacerse de su cuenta en nuestra aplicación web y desvincularse de ella.

Precondiciones: El usuario esta iniciado sesión en la página.

Pasos:

1. El usuario selecciona la opción “Mi cuenta”.
2. Selecciona la opción darse de baja.
3. El sistema mostrará un mensaje de seguridad para confirmar la acción.
4. Cuando se confirma la acción todos los datos de la cuenta de usuario serán eliminados

Postcondiciones: El sistema mostrará la vista principal de la página.

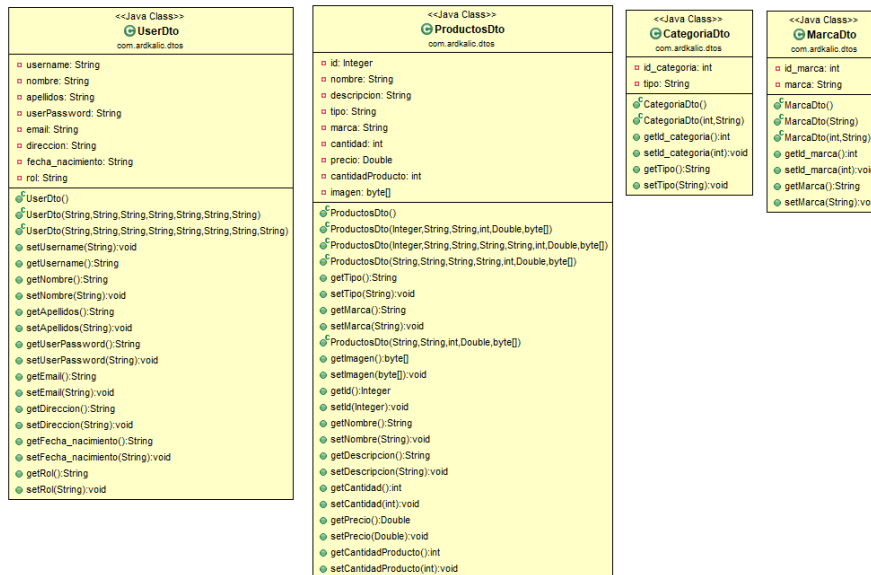
Alternativa:

1. El usuario no dispone de una cuenta registrada.

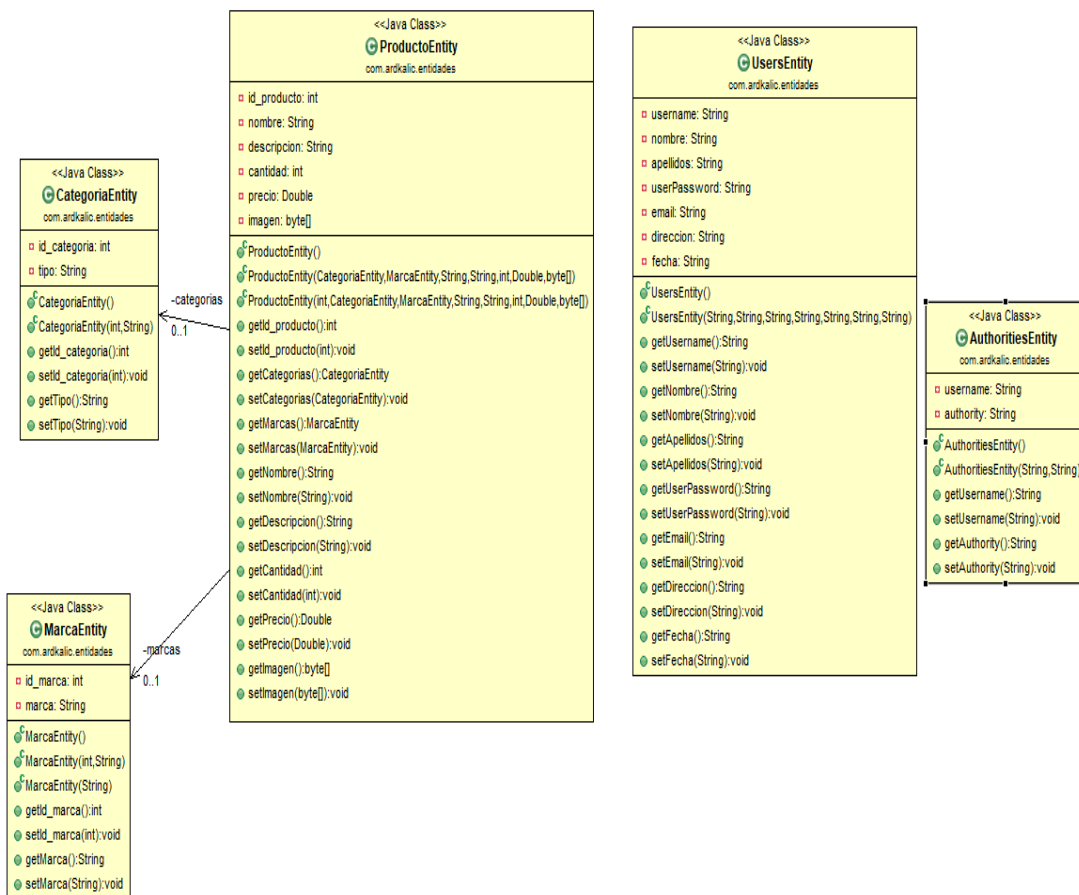


4.2. Diagramas de clases

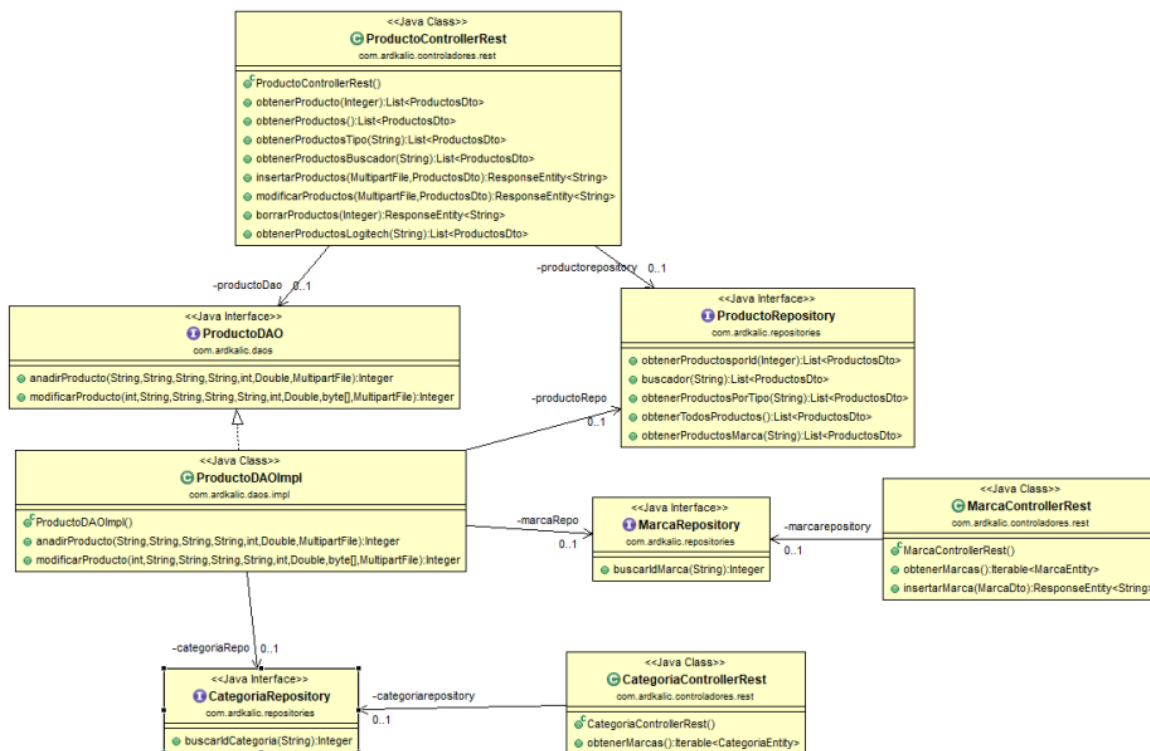
- En primer lugar, en la imagen que se mostrara a continuación son los diagramas de clase de los DTO.



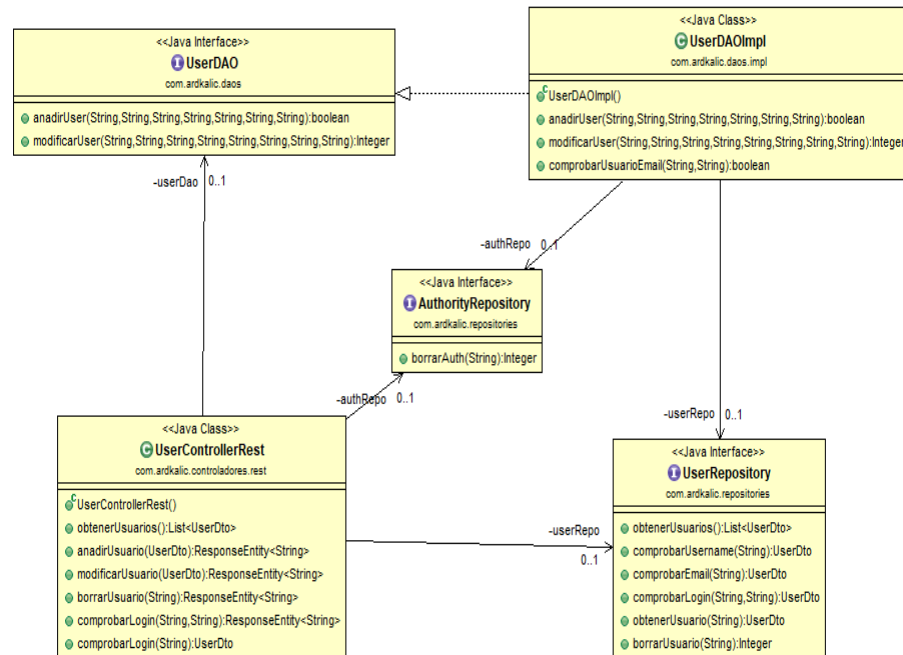
- Estas son las entidades presentes de mi aplicación



- A continuación, los diagramas de clases de todo lo relacionado con los productos

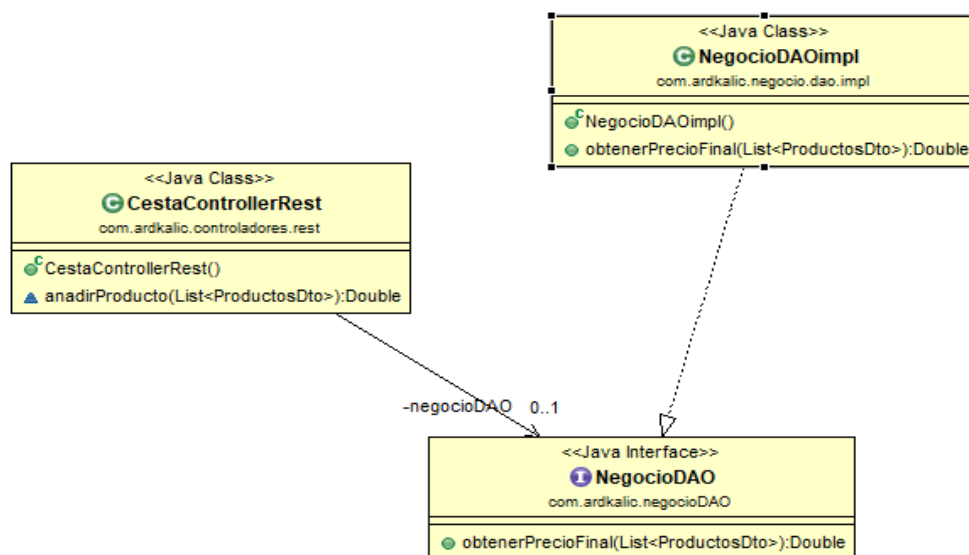


- En la siguiente imagen son los diagramas de clases relacionado con los usuarios



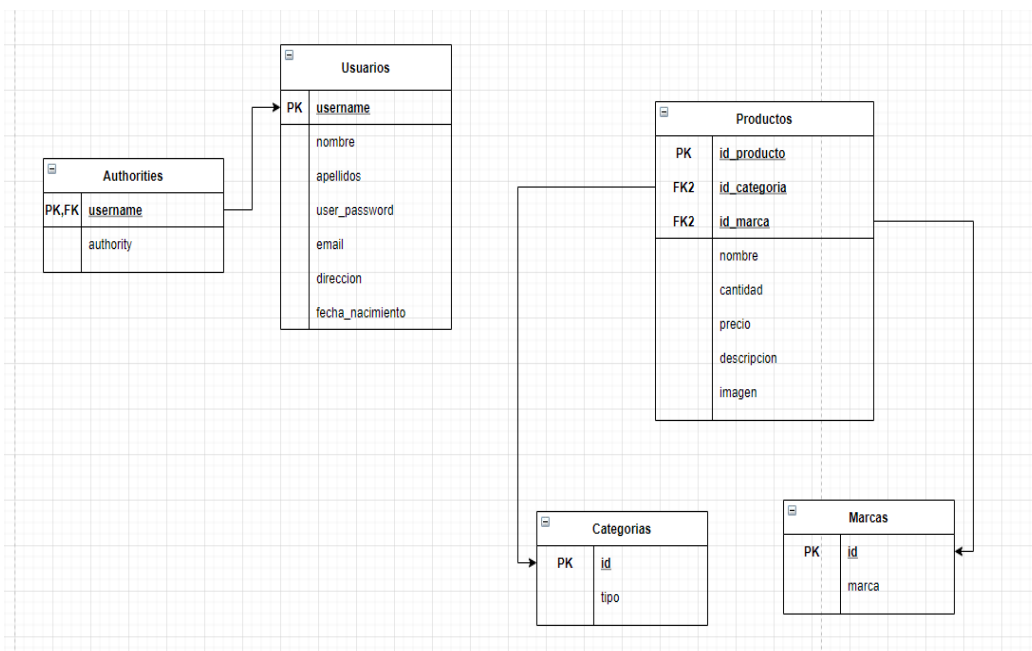


- Seguidamente mostramos los diagramas de clase relacionados con el controlador cesta





4.3. Modelo de dominio





5. Arquitectura de la aplicación

5.1. Estructura del proyecto

Usaremos un modelo vista-controlador donde se destacan la de presentación, negocio y datos. La arquitectura que hemos elegido será REST, ya que nos parece la forma más óptima para la creación de la aplicación.

Para la parte back usaremos Spring, que constará de los siguientes paquetes:

- Configuración
- Controladores: clases que trataran los eventos que se producen en la interfaz gráfica(vista).
- Dtos: Clase que almacenara datos.
- Entities: clases asociadas a una tabla de la BBDD.
- Repositories: clases que se encargan de gestionar todas las operaciones de persistencia contra una tabla de la base de datos.
- Negocio: será la parte que se encargará de la capa de negocio en la cual se harán lo cálculos necesario de la aplicación.
- Daos: es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos.

La parte front será una aplicación de Angular que nos ayudará a relacionar nuestra capa de presentación con nuestra capa de negocio y constará de:

- Assets: carpeta donde se almacenarán imágenes y otro tipo de archivos que puedan ser utilizados.
- app: Carpeta donde se encuentran todos los componentes de la app.
- app-routing.module.ts: archivo que configura el enrutamiento de todos los componentes de la aplicación.
- index.html: documento base donde todos los eventos de la app se verán por pantalla.



Por último, para la base de datos modelo relacional hemos usado el gestor de base de datos MySQL

5.2. Librerías externas utilizadas

Usaremos:

- Java.util.*
- Bootstrap
- Ngx-toastr
- Ngx-pagination
- Angular material
- Spring-boot-starter-data-jpa
- Spring-boot-starter-data-web
- Spring-boot-devtools
- Mysql-connector-java
- Spring-boot-starter-test
- Tomcat-embed-jasper
- Spring-boot-maven-plugin
- Maven-surefire-plugin
- Jstl









6. Manual de despliegue

En este apartado llevaremos a cabo el despliegue de la aplicación con Docker para ello se deberán seguir los siguientes pasos:

En primer lugar, se deberá tener instalado en el equipo el software de Docker para poder llevar a cabo las instrucciones y comandos que se describirán a continuación:

Para crear la imagen de la BBDD el archivo Dockerfile tendrá que estar colocado como se puede observar en la imagen.

 .git	18/05/2021 23:12	Carpeta de archivos	
 ProyectoFinal	18/05/2021 21:24	Carpeta de archivos	
 ProyectoFinalAngular	17/05/2021 15:39	Carpeta de archivos	
 scriptSQL	18/05/2021 21:07	Carpeta de archivos	
 Dockerfile	18/05/2021 21:13	Archivo	1 KB
 README.txt	18/05/2021 21:01	Documento de te...	2 KB

Y deberá contener los siguientes comandos que se muestran en la imagen.

```
#Dockerfile para crear una imagen basada en mysql
FROM mysql

#Se copian los scripts desde la carpeta en la que estén al directorio dentro del contenedor
#En este ejemplo no se va a mapear mediante volúmenes ni a bindear la bbdd.
COPY ./scriptSQL/ /docker-entrypoint-initdb.d/

#docker build -t mariorg/bbdd . es lo que hago para generar la imagen
#docker run -d --network ardalic --name mi_mysql -e MYSQL_ROOT_PASSWORD=mario -e MYSQL_DATABASE=ardalic -e MYSQL_PASSWORD=mario -p 3307:3306 mariorg/bbdd y esto para arrancar el contenedor
```

Como se puede apreciar en la imagen para evitar conflictos con el collate de la base de datos recomendamos descargar la imagen de la última versión de MySQL.

Para crear la imagen de la base de datos usaremos el siguiente comando: `docker build --tag nombreImagen .` (el punto hace referencia a la ruta en la que estamos).

Una vez creada la imagen, se deberá crear una red interna de docker para ello se utilizará el comando: `docker network create nombreNetwork .`



Cuando se haya creado la red se procederá a ejecutar el comando:














```
docker run -d --network nombreNetwork --name nombreContendor -e  
MYSQL_ROOT_PASSWORD=password
```

```
-e MYSQL_DATABASE=nombreBaseDatos -e MYSQL_PASSWORD=password -p 3307:3306  
nombreImagen.
```

La contraseña deberá coincidir con la que se haya definido en el application.properties en el apartado spring.datasource.password de nuestra parte back de la aplicación de Spring.

Ahora, una vez creada la imagen y haber arrancado el contenedor correspondiente a la imagen de la base de datos continuaremos con la explicación para crear imagen y ejecutar el contenedor de la parte back de la aplicación.

Como se mostrará en la imagen que se presenta a continuación el Dockerfile tendrá que tener la siguiente estructura de carpetas:

	.mvn	24/04/2021 8:44	Carpeta de archivos	
	.settings	16/05/2021 20:23	Carpeta de archivos	
	bin	11/05/2021 18:39	Carpeta de archivos	
	src	24/04/2021 8:44	Carpeta de archivos	
	target	18/05/2021 21:23	Carpeta de archivos	
	.classpath	17/05/2021 15:20	Archivo CLASSPATH	3 KB
	.gitignore	24/04/2021 8:44	Documento de te...	1 KB
	.project	17/05/2021 13:40	Archivo PROJECT	2 KB
	ardkalic.war	18/05/2021 21:23	Archivo WAR	48.010 KB
	Dockerfile	17/05/2021 13:36	Archivo	1 KB
	mvnw	24/04/2021 8:44	Archivo	11 KB
	mvnw.cmd	24/04/2021 8:44	Script de comand...	7 KB
	pom.xml	17/05/2021 14:13	Documento XML	3 KB



Y el contenido del Dockerfile será como se indica en la siguiente imagen:

```
FROM tomcat:9.0.24

EXPOSE 8080

RUN rm -rf /usr/local/tomcat/webapps/

#incluimos el war generado en la carpeta correspondiente del servidor tomcat
ADD ./ardkalic.war /usr/local/tomcat/webapps/

CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

#Un ejemplo para la sentencia de arranque:
#docker run --network nombreNetwork --name nombreContenedor -p 8080:8080 nombreImagen
```

Para generar la imagen antes de nada hay que generar el war de la aplicación y en el application.properties en el apartado spring.datasource.url deberá cambiar localhost por el nombre del contenedor de la base de datos en la imagen se muestra como es el application.properties sin el cambio realizado.

```
server.servlet.context-path:/ardkalic

spring.datasource.url:jdbc:mysql://localhost:3306/ardkalic
spring.datasource.username:root
spring.datasource.password:mario
spring.datasource.driver-class-name:com.mysql.cj.jdbc.Driver
spring.jpa.database-platform:org.hibernate.dialect.MySQL5Dialect
spring.jpa.generate-ddl:true
spring.jpa.hibernate.ddl-auto:update
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.use_sql_comments=true
spring.jpa.properties.hibernate.format_sql=true
```

Cuando hayamos hecho los cambios y generado el war ejecutaremos el comando docker build -t nombreImagen . (el punto hace referencia a la ruta en la que estamos) para generar la imagen.

Después de haber generado la imagen, a continuación, ejecutaremos el siguiente comando para crear el contenedor, hay que tener en cuenta que el nombre de la network debe ser el mismo que hemos utilizado para el contenedor de la base de datos:











```
docker run --network nombreNetwork --name nombreContenedor -p 8080:8080
nombreImagen
```



Con estos pasos ya habremos conseguido crear la imagen y arrancar el contenedor de nuestra parte back de la aplicación.

A continuación, crearemos la imagen y contenedor para la parte front de nuestra aplicación que se llevara a cabo de la siguiente manera:

Primeramente, deberemos colocar nuestro archivo Dockerfile siguiendo la estructura mostrada en la imagen.

	.vscode	17/05/2021 15:39	Carpeta de archivos	
	dist	18/05/2021 23:12	Carpeta de archivos	
	e2e	17/05/2021 15:39	Carpeta de archivos	
	node_modules	17/05/2021 15:39	Carpeta de archivos	
	src	18/05/2021 19:53	Carpeta de archivos	
	.browserslistrc	24/04/2021 8:44	Archivo BROWSER...	1 KB
	.dockerignore	16/05/2021 22:17	Archivo DOCKERI...	1 KB
	.editorconfig	24/04/2021 8:44	Archivo EDITORC...	1 KB
	.gitignore	24/04/2021 8:44	Documento de te...	1 KB
	angular.json	15/05/2021 17:33	Archivo JSON	4 KB
	Dockerfile	16/05/2021 22:23	Archivo	1 KB
	karma.conf.js	24/04/2021 8:44	Archivo JavaScript	2 KB
	package.json	15/05/2021 19:09	Archivo JSON	2 KB
	package-lock.json	15/05/2021 19:09	Archivo JSON	603 KB
	README.md	24/04/2021 8:44	Archivo MD	2 KB
	tsconfig.app.json	24/04/2021 8:44	Archivo JSON	1 KB
	tsconfig.json	24/04/2021 8:44	Archivo JSON	1 KB
	tsconfig.spec.json	24/04/2021 8:44	Archivo JSON	1 KB
	tslint.json	24/04/2021 8:44	Archivo JSON	4 KB



Para generar la imagen el contenido de nuestro Dockerfile tendrá los siguientes comandos que aparecen en la imagen.

```
FROM node:12-alpine as build-step

RUN mkdir -p /app

WORKDIR /app

COPY package.json /app

RUN npm install

COPY . /app

RUN npm run build --prod

#Segunda Etapa
FROM nginx:1.17.1-alpine
#Si estas utilizando otra aplicacion cambia PokeApp por el nombre de tu app
COPY --from=build-step /app/dist/ProyectoFinal /usr/share/nginx/html
```

Seguidamente, ejecutaremos el comando `docker build -t nombrelimagen .` con este comando se generará nuestra imagen de la parte front.

Cuando se haya generado la imagen con el comando `docker run -d -p 4200:80 --name nombreContenedor nombrelimagen` se creará nuestro contenedor, se podrá poner cualquier puerto mientras sea puerto:80.

Resumiendo, para ver la aplicación en funcionamiento en cualquier equipo se deberán ejecutar los comandos de arranque de los contenedores con el nombre de la imagen que se presentan a continuación:

En primer lugar, hacer `docker network create ardkalic`

Imágenes Mario:

Front: `docker run -d -p 4200:80 --name ardkalic_front mariorg/front`

BBDD: `docker run -d --network ardkalic --name mi_mysql`

`-e MYSQL_ROOT_PASSWORD=mario -e MYSQL_DATABASE=ardkalic`

`-e MYSQL_PASSWORD=mario -p 3307:3306 mariorg/bbdd`



Back: `docker run --network ardkalic --name ardkalic_back -p 8080:8080 mariorg/back`

Nota: Recomendamos ejecutar en último lugar el contenedor que corresponde a back para ver así lo logs y ver su correcto funcionamiento o también puedes usar la opción `-d` y ejecutar los contenedores en el orden que quieras.

Imágenes Adrián:

Front: `docker run -d -p 4200:80 --name ardkalic_front giddychutoy/front`

Back: `docker run --network ardkalic --name ardkalic_back -p 8080:8080 giddychutoy /back`

BBDD: `docker run -d --network ardkalic --name mi_mysql`

`-e MYSQL_ROOT_PASSWORD=mario -e MYSQL_DATABASE=ardkalic`

`-e MYSQL_PASSWORD=mario -p 3307:3306 giddychutoy /bbdd`