

Trabajo Práctico

[TA046] Ingeniería de Software I
Curso 2
Segundo cuatrimestre de 2024

Nombre	Padrón
Converso, Lara Daniela	107632
Galindez, María	105741
De San Vicente, Hernan Martin	108800
Cuevas, Juan Francisco	107963
Ahumada, Lorenzo Sebastian	106780
Fernandez, Milton Tomas	108749

Índice

1. Introducción	2
2. Arquitectura	2
2.1. Vista Lógica	2
2.1.1. Componentes clave	2
2.1.2. Diagrama de clases UML	3
2.2. Vista de Desarrollo	3
2.2.1. Componentes clave	4
2.2.2. Organización de Carpetas	4
2.2.3. Herramientas utilizadas	4
2.3. Vista de Procesos	5
2.3.1. Ejemplos de flujos	5
2.3.2. Diagramas de secuencia	6
2.4. Vista Física	7
2.4.1. Componentes clave	7
2.4.2. Diagrama de despliegue	8
2.4.3. Despliegue	8
2.5. Escenarios	8
2.5.1. Diagrama de casos de uso	9
2.5.2. Detalles adicionales	9
3. ¿Cómo correr el proyecto?	9
3.1. Set up repo en local	9
3.1.1. Levantar con docker el proyecto	9
3.1.2. Levantar en local el proyecto	9

1. Introducción

El presente informe reúne la documentación de la solución del trabajo práctico de la materia Ingeniería de Software I. El mismo consiste en desarrollar una sistema que permita realizar pedidos de productos. Nuestro objetivo es aplicar los conceptos explicados a lo largo de la materia, utilizando el lenguaje de tipado estático Java.

2. Arquitectura

2.1. Vista Lógica

Esta vista se enfoca en el diseño funcional del sistema, mostrando las entidades principales y cómo interactúan entre ellas.

2.1.1. Componentes clave

- **Usuario:** Permite registrarse, autenticarse, recuperar su contraseña, hacer pedidos y ver/-cancelar pedidos.
- **Administrador:** Crea productos, gestiona los stocks y los pedidos confirmados.
- **Pedido:** Contiene productos y características, reglas dinámicas (peso, cantidad), estados (CONFIRMADO, PROCESO, ENVIADO, CANCELADO). Los estados se representan como un **enum**.
- **Producto:** Tiene atributos dinámicos (color, talla, etc.) y permite extensibilidad en los atributos.

2.1.2. Diagrama de clases UML

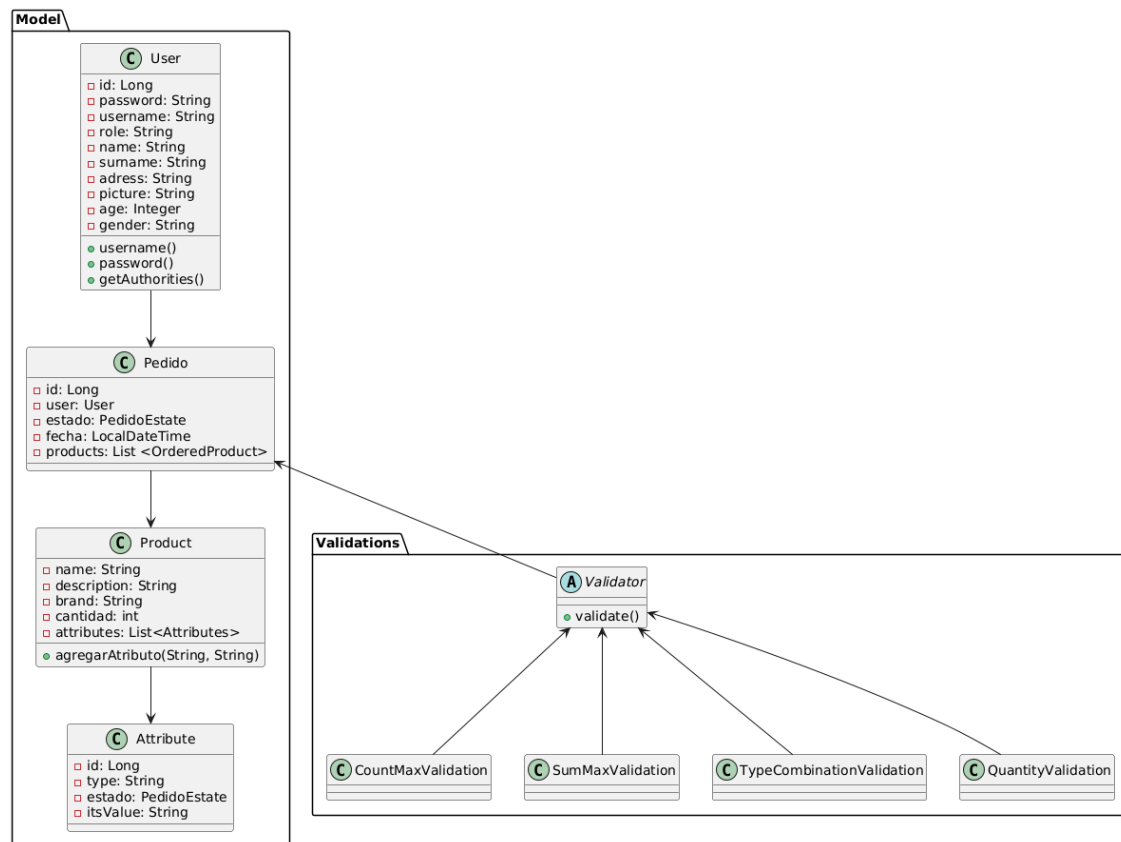


Figura 1: Diagrama de clases general.

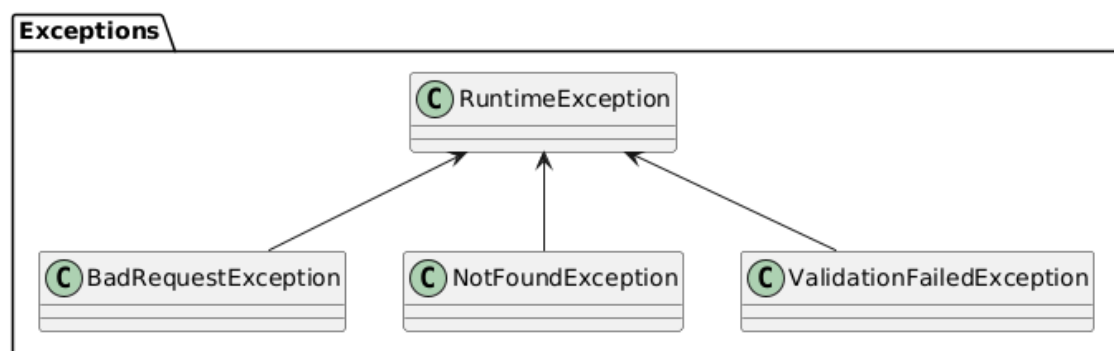


Figura 2: Diagrama de clases de las Excepciones.

2.2. Vista de Desarrollo

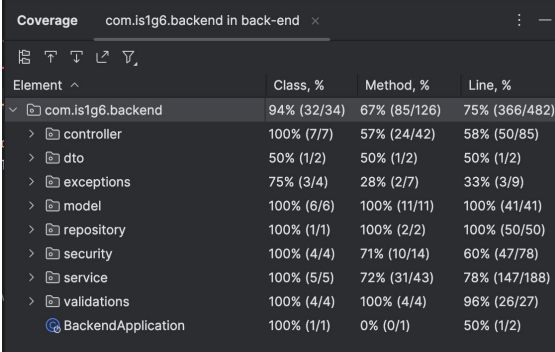
Esta vista muestra la estructura del software, centrándose en cómo los componentes se organizan en el código.

2.2.1. Componentes clave

- **Backend:** Implementado en Java. Contiene controladores, servicios, repositorios y modelos. Utiliza JWT para autenticación.
- **Front-end básico:** Interfaz simple para que los usuarios realicen pedidos y los administradores gestionen productos.
- **API REST:** Expone endpoints para manejar las operaciones sobre pedidos, productos y usuarios.
- **Persistencia:** Base de datos para almacenar usuarios, productos y pedidos.

2.2.2. Organización de Carpetas

- El **Front-end** se encuentra en la carpeta **resources**, donde está organizado por templates, imágenes y funciones.
- El **Backend** sigue la estructura básica de Spring Boot (**model**, **controller**, **repository**, y **service**) y se encuentra en la carpeta **backend**. Dentro del backend tenemos:
 - **Controladores:** Manejan las solicitudes HTTP entrantes y coordinan las respuestas.
 - **Servicios:** Contienen la lógica de negocio principal.
 - **Repositorios:** Gestionan la interacción con la base de datos.
 - **Modelos:** Representan las entidades y sus atributos.
 - **Validaciones:** Contiene las distintas reglas y validaciones de los pedidos.
 - **dto:** Aquí están los DTOs de los distintos objetos.
 - **Security:** Seteo de JWT.
 - **Exceptions:** gestión de errores del sistema mediante clases específicas.
- Los tests del backend están ubicados en la carpeta **backend/src/test** e incluyen pruebas unitarias desarrolladas con JUnit y Mockito.
 - La cobertura de estos es de un 75 %:



Element	Class, %	Method, %	Line, %
com.is1g6.backend	94% (32/34)	67% (85/126)	75% (366/482)
controller	100% (7/7)	57% (24/42)	58% (50/85)
dto	50% (1/2)	50% (1/2)	50% (1/2)
exceptions	75% (3/4)	28% (2/7)	33% (3/9)
model	100% (6/6)	100% (11/11)	100% (41/41)
repository	100% (1/1)	100% (2/2)	100% (50/50)
security	100% (4/4)	71% (10/14)	60% (47/78)
service	100% (5/5)	72% (31/43)	78% (147/188)
validations	100% (4/4)	100% (4/4)	96% (26/27)
BackendApplication	100% (1/1)	0% (0/1)	50% (1/2)

Figura 3: Cobertura de los tests unitarios.

2.2.3. Herramientas utilizadas

- **Gestión de código:** GitLab.
- **Entorno de contenedores:** Docker y Docker Compose.
- **Pruebas:** Unit Testing con JUnit.

- **Documentación API:** Postman.

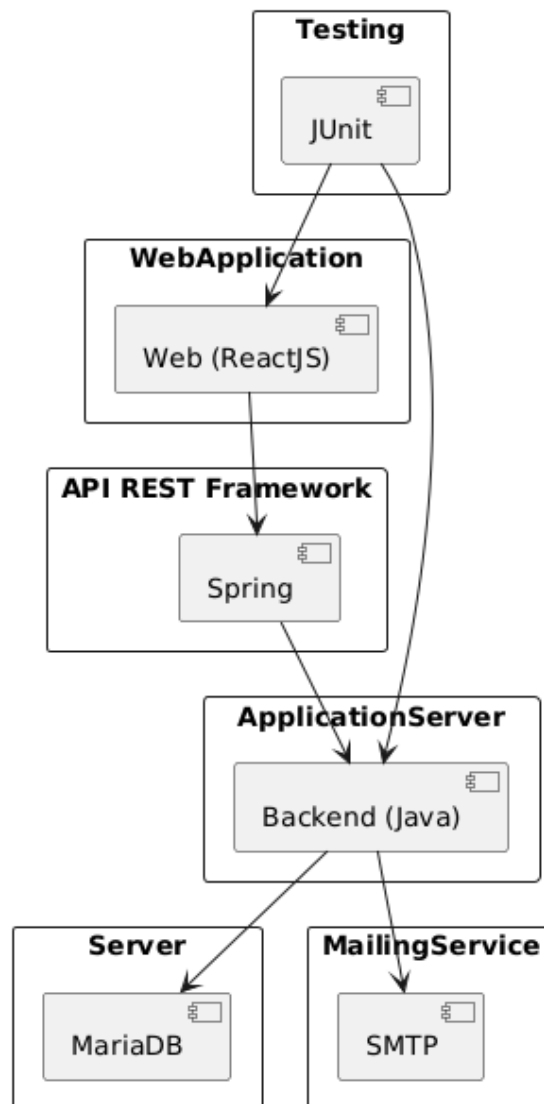


Figura 4: Diagrama de componentes general.

2.3. Vista de Procesos

Describe el comportamiento dinámico del sistema en tiempo de ejecución, centrándose en la concurrencia y sincronización de procesos.

2.3.1. Ejemplos de flujos

- **Registro y autenticación:** El usuario se registra y autentica usando JWT.
- **Pedido:** El usuario elige productos, se validan las reglas dinámicas (e.g., peso total no excede 10kg), se confirma el pedido, se actualiza el stock y se notifica al usuario por email.
- **Administración:** El administrador gestiona productos y estados de pedidos.
- **Cancelación:** Si el pedido no está en PROCESO, el usuario lo cancela y el stock se restaura.

2.3.2. Diagramas de secuencia

Registro de Usuario

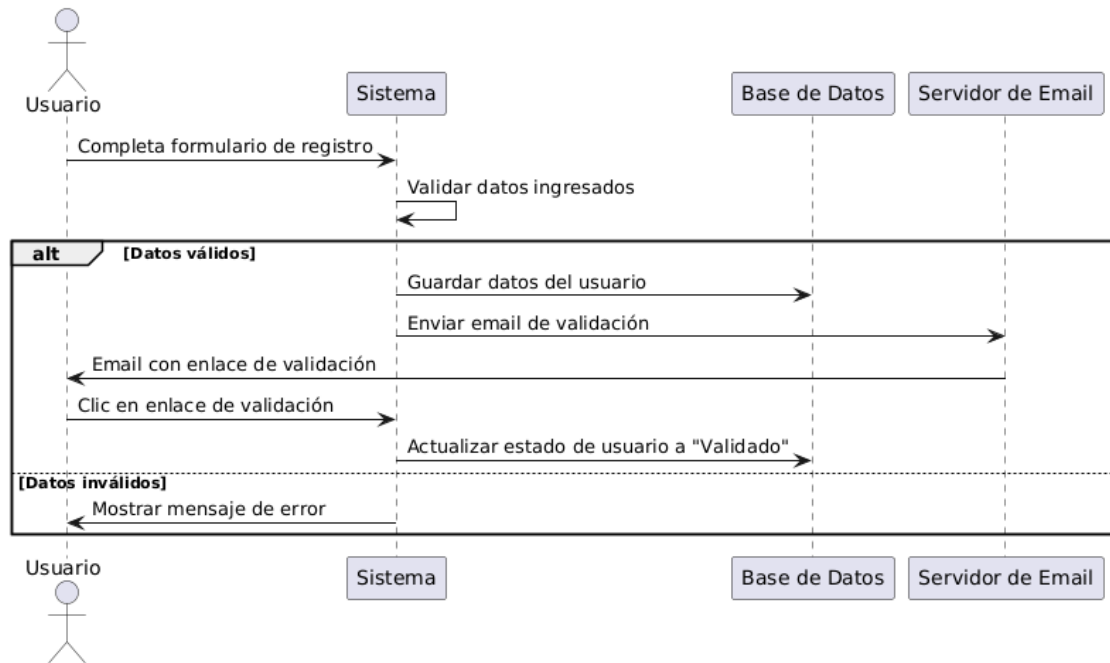


Figura 5: Diagrama de Secuencia n°1.

Realización de un Pedido

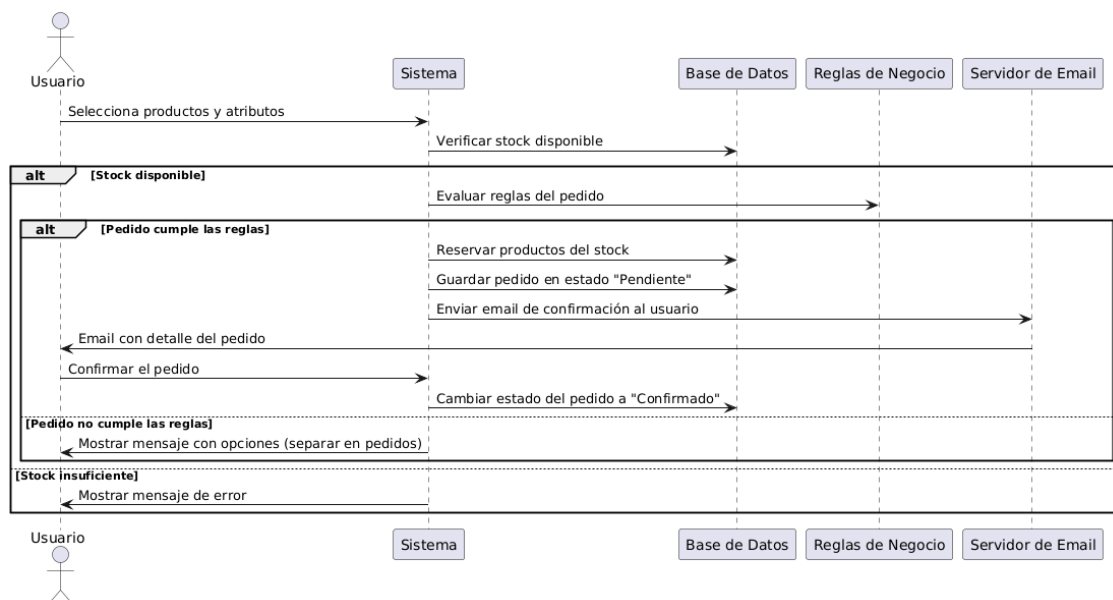


Figura 6: Diagrama de Secuencia n°2.

Gestión de pedidos por el administrador

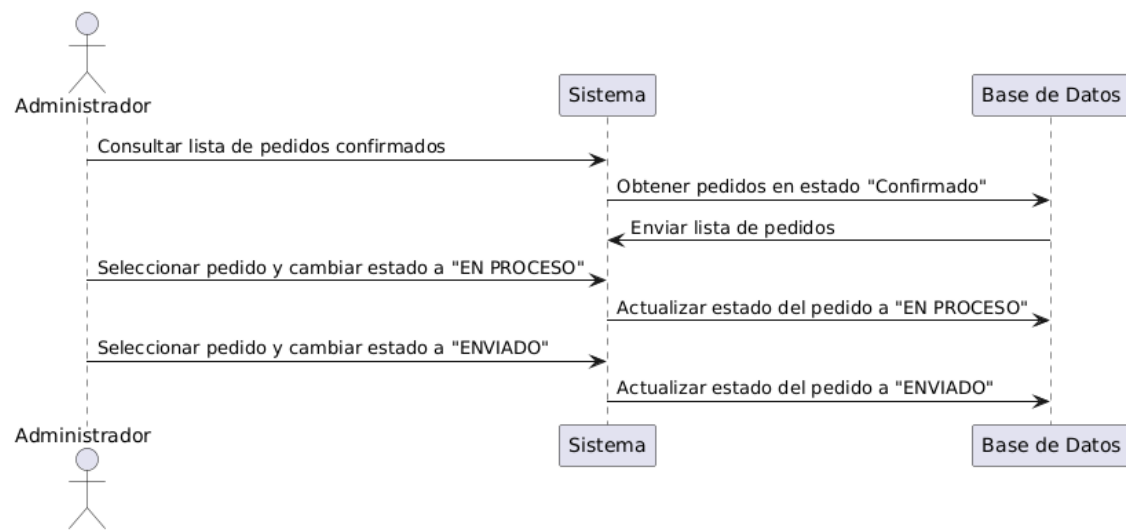


Figura 7: Diagrama de Secuencia n°3.

2.4. Vista Física

Describe el entorno de hardware y cómo el software se distribuye en los distintos nodos.

2.4.1. Componentes clave

- **Servidor de aplicaciones:** Ejecuta la API REST (Dockerizado).
- **Base de datos:** Almacenamiento de datos (Dockerizado).
- **Clientes:** Usuarios finales acceden al sistema vía navegador web o dispositivos móviles.

2.4.2. Diagrama de despliegue

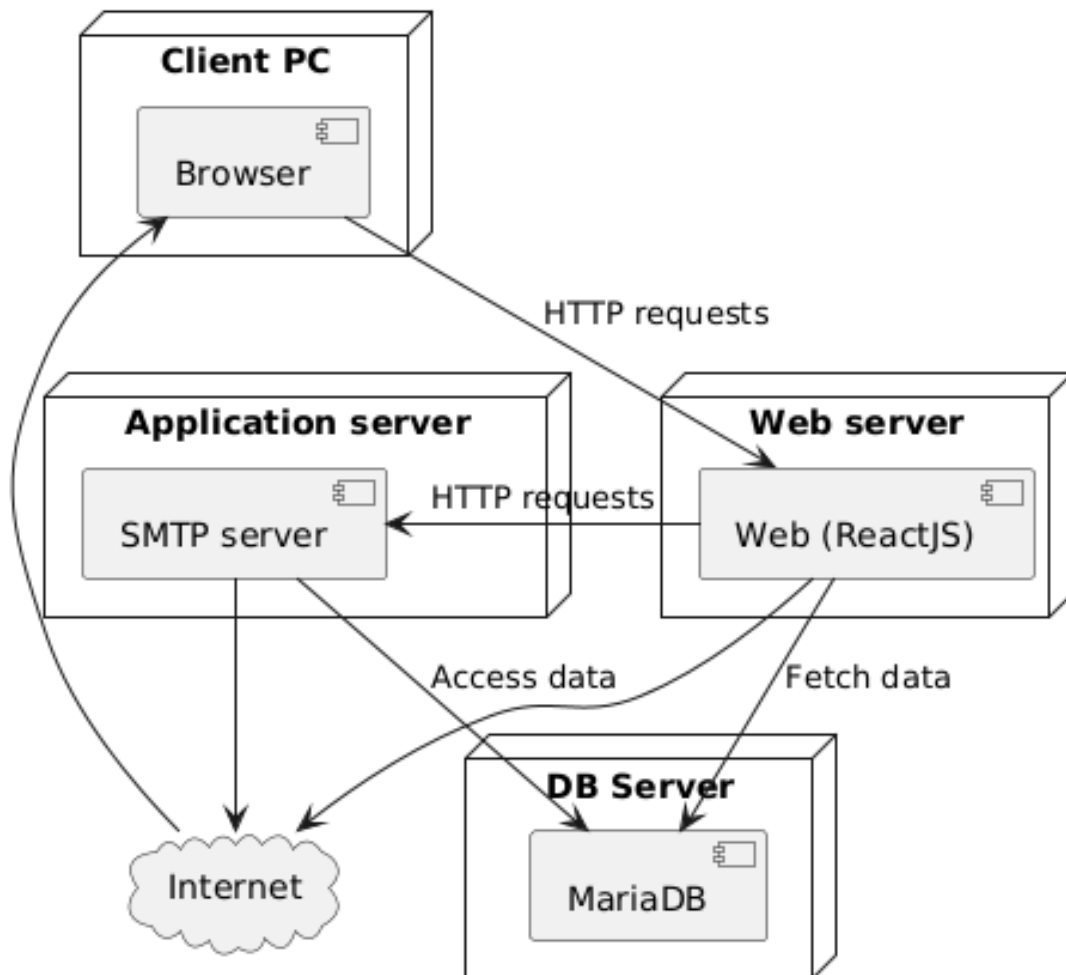


Figura 8: Diagrama de Despliegue.

2.4.3. Despliegue

La aplicación se ejecuta en un entorno de contenedores (Docker Compose) que orquesta la API, la base de datos y los servicios de autenticación.

2.5. Escenarios

Los siguientes casos de uso ilustran las funcionalidades del sistema.

2.5.1. Diagrama de casos de uso

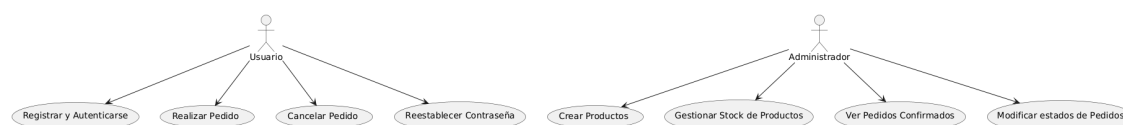


Figura 9: Diagrama de Casos de Uso.

2.5.2. Detalles adicionales

- **Registrar y Autenticarse:** El usuario puede crear una cuenta nueva, iniciar sesión con credenciales y recuperar su contraseña mediante email.
- **Realizar Pedido:** Los usuarios seleccionan productos, se validan las reglas de negocio, se confirma el pedido y reciben una notificación.
- **Cancelar Pedido:** Los usuarios cancelan pedidos no procesados, actualizando automáticamente el inventario.
- **Gestión Administrativa:** Los administradores crean productos, gestionan inventarios y supervisan pedidos confirmados.

3. ¿Cómo correr el proyecto?

3.1. Set up repo en local

Para clonar el repositorio en local, debemos correr la linea

```
git clone https://gitlab.com/Laraconverso/is1_grupo6.git
```

3.1.1. Levantar con docker el proyecto

Advertencia: en el caso de que sea la primera vez corriendo el proyecto chequear que en el archivo

```
.../resources/application.properties
```

la configuración de la base de datos debe estar en “create”. Este se debe ver algo así:

```
spring.application.name=back-end
spring.jpa.hibernate.ddl-auto=create
```

Una vez hecho eso, se debe correr el siguiente comando.

```
docker-compose up --build
```

En caso de querer tirar abajo todos los containers se debe correr este comando:

```
docker-compose down
```

Y en caso de querer frenar un único container, se corre el siguiente comando reemplazando <container> por el nombre del container.

```
docker compose stop <container>
```

3.1.2. Levantar en local el proyecto

Simplemente se corre el proyecto haciendo el run necesario.