Transformación Digital de "Perfulandia SPA"

Integrantes: Brayan Ahumada Rojas

Cristian Ormazabal

Sección: 011D

Profesor: Eduardo Baeza

Índice

[**1. Diagrama de arquitectura de microservicios 3**](#_z06snt4j5q09)

[1.1. Diagrama Usuario 3](#_2gsms42lcxdi)

[1.2. Diagrama Inventario 3](#_yl3e2ah6mnp2)

[1.3. Diagrama Envío 4](#_nwyo12oyfd6q)

[1.4. Diagrama de Despliegue 4](#_s3uyueac2lwk)

[**2. Plan de pruebas 5**](#_iq8u63zcsksz)

[2.1. Pruebas unitarias 5](#_3t07aetn14rg)

[2.1.1. Herramientas y frameworks utilizados (JUnit, Mockito, etc.) 5](#_e9vnz0283lcq)

[2.1.2. Justificación de su elección 5](#_aayin0fqi6oq)

[2.1.3. Imágenes de la codificación de las pruebas 6](#_khh07ce1c7au)

[2.2. Pruebas de integración 9](#_6ktlyi58ubr)

[2.2.1. Frameworks y herramientas utilizados (SpringBootTest, MockMvc, etc.) 9](#_djixrriqmrci)

[2.2.2. Justificación de su elección 9](#_jakaktd1ezok)

[2.2.3. Imágenes de la codificación de las pruebas 10](#_h4b2odhy2cpj)

[● EnvioRestControllerTest.java 10](#_x377gkz86jho)

[**3. Ejecución de pruebas 13**](#_gyi50jfc2wv5)

[3.1. Evidencias de ejecución 13](#_71lg58jpzt8i)

[3.1.1. Imágenes de resultados exitosos o fallidos 13](#_yb9ectyoj4lb)

[3.1.2. Cobertura alcanzada 14](#_2ysoajpip63v)

[3.1.3. Documentación por Swagger 14](#_ubiisznzmr4n)

[**4. Uso de git y Github 15**](#_bskb6pq1d18r)

[4.1. Comandos utilizados 15](#_a9qj8zv9klpd)

[4.2. Estructura del repositorio 18](#_o54e056yrzlu)

[4.2.1. Descripción de carpetas Exp1, Exp2Exp3 18](#_ana89uuus2mj)

[4.2.2. Rama utilizada 19](#_pa932cfjyx61)

[**5. Conclusión 19**](#_yznzle7zfvjp)

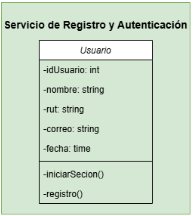
[5.1. Reflexión general del trabajo realizado 19](#_ce412dxm9zec)

[5.2. Principales aprendizajes 19](#_4vastkemwa10)

# 

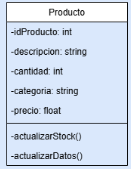
## **Diagrama de arquitectura de microservicios**

### Diagrama Usuario

* + 1. **Diagrama de Clases y Casos de Uso**

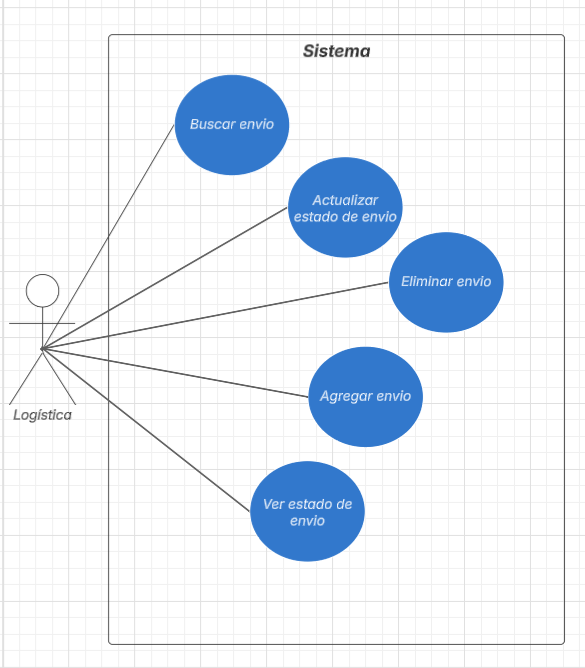
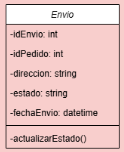
### Diagrama Inventario

* + 1. Diagrama de Clases y Casos de Uso

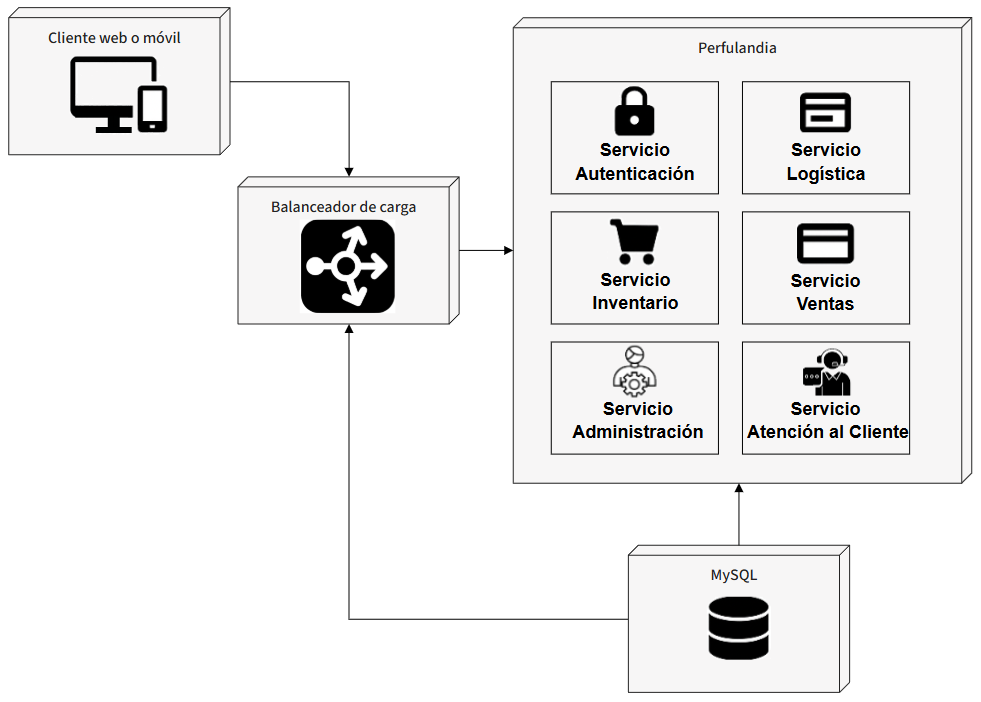


### Diagrama Envío

* + 1. Diagrama de Clases y Casos de Uso



### Diagrama de Despliegue

****

## **Plan de pruebas**

### **Pruebas unitarias**

#### Herramientas y frameworks utilizados (JUnit, Mockito, etc.)

Las herramientas utilizadas para trabajar en este proyecto son:

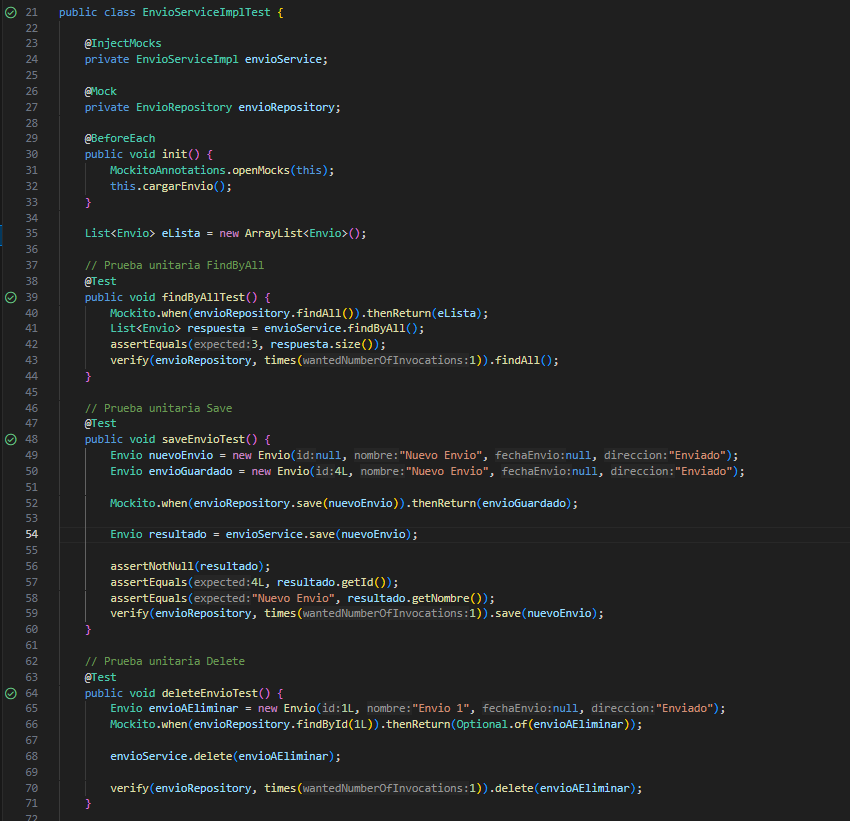
* JUnit para la ejecución de las pruebas unitarias
* Mockito para la simulación de las dependencias,facilitando la creación de objetos simulados

#### Justificación de su elección

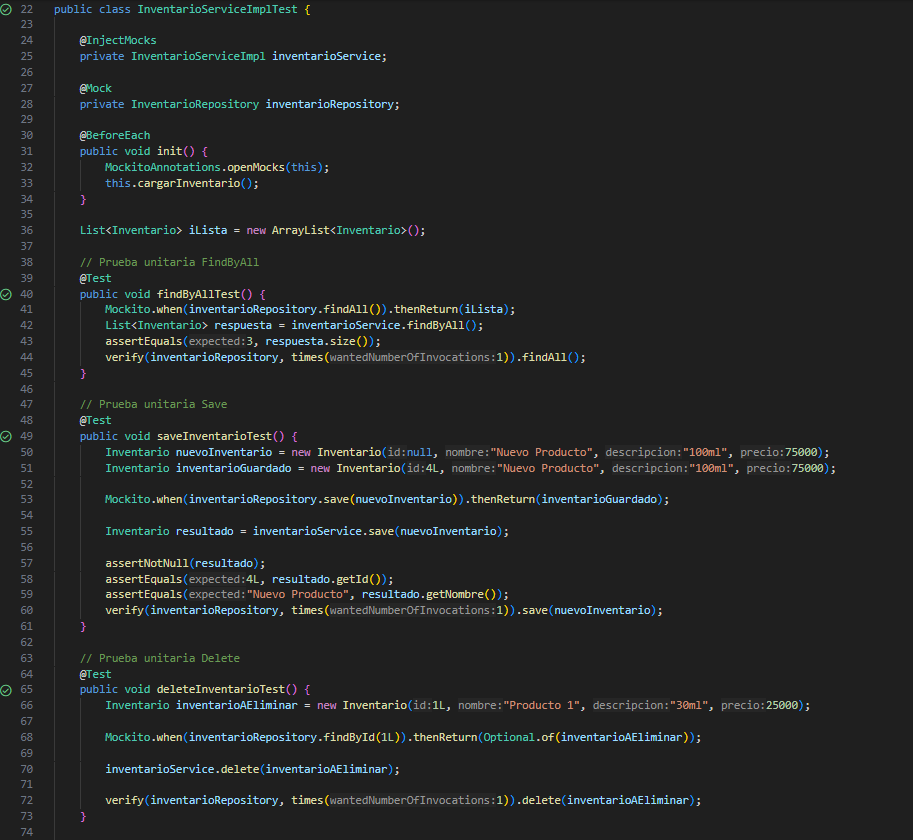
Estas herramientas fueron seleccionadas por alta eficiencia y facilidad de integración en el desarrollo de software con Java y Springboot

#### Imágenes de la codificación de las pruebas

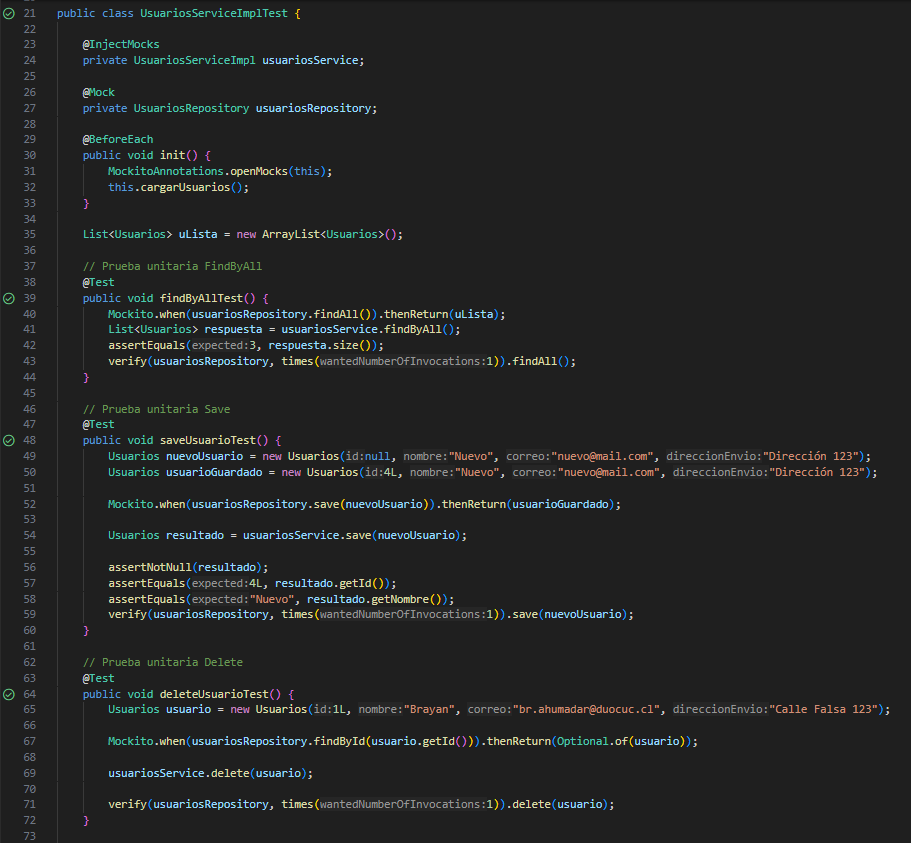
* EnvioServiceImplTest.java



* InventarioServiceImplTest.java



* UsuarioServiceImplTest.java



### **Pruebas de integración**

#### Frameworks y herramientas utilizados (SpringBootTest, MockMvc, etc.)

Las herramientas utilizadas para trabajar en este proyecto son:

* SpringBootTest: Su función principal es cargar el contexto completo de la aplicación Spring, permitiendo que las pruebas se ejecuten con la misma configuración que tendrían en un entorno de producción
* MockMVC: Es una herramienta fundamental en Spring Boot 3 para realizar pruebas exhaustivas de controladores REST y verificar el intercambio de datos en formato JSON

#### Justificación de su elección

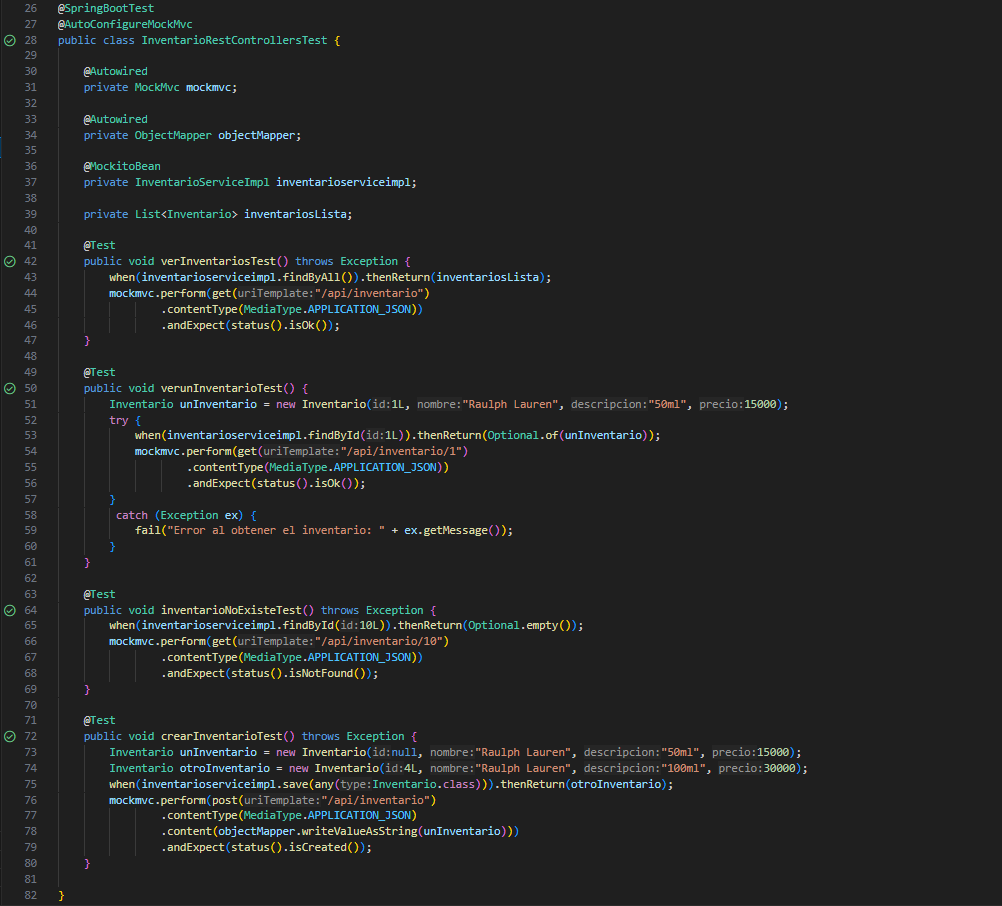
Fueron seleccionados por que permiten la integración de múltiples componentes del sistema simulando llamadas HTTP a los controladores

#### Imágenes de la codificación de las pruebas

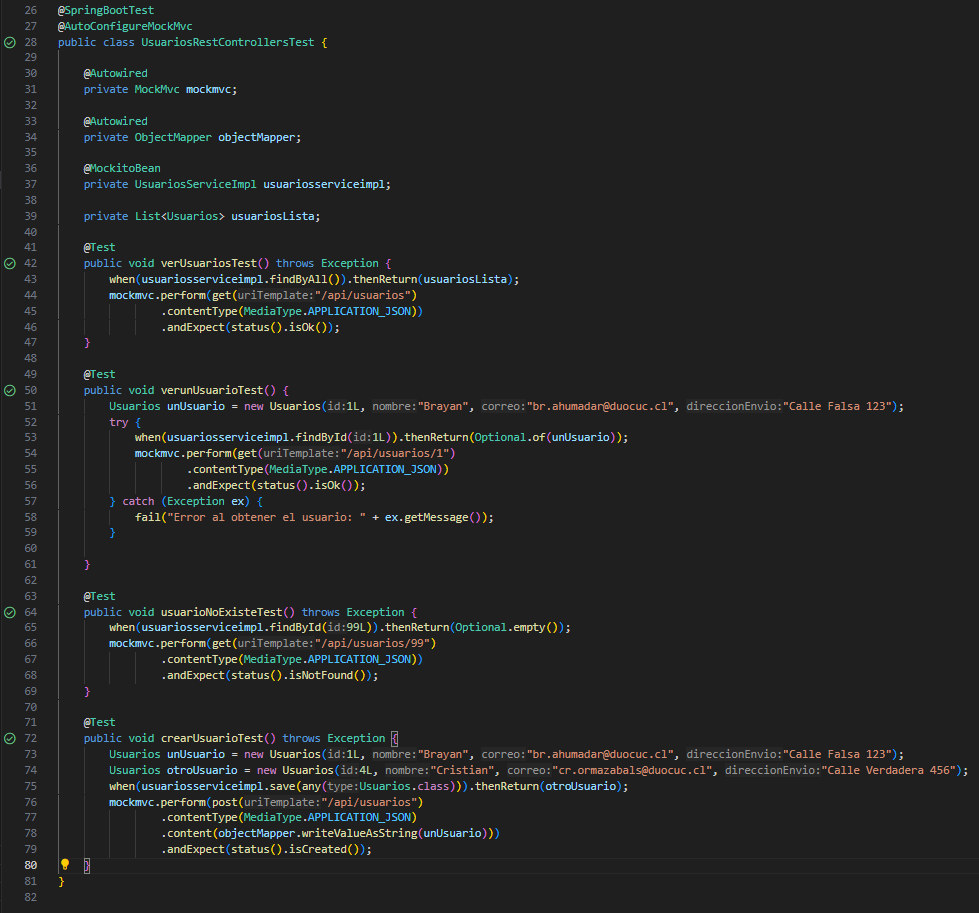
#### EnvioRestControllerTest.java



* InventarioRestControllerTest.java



* UsuarioRestControllerTest.java



## **Ejecución de pruebas**

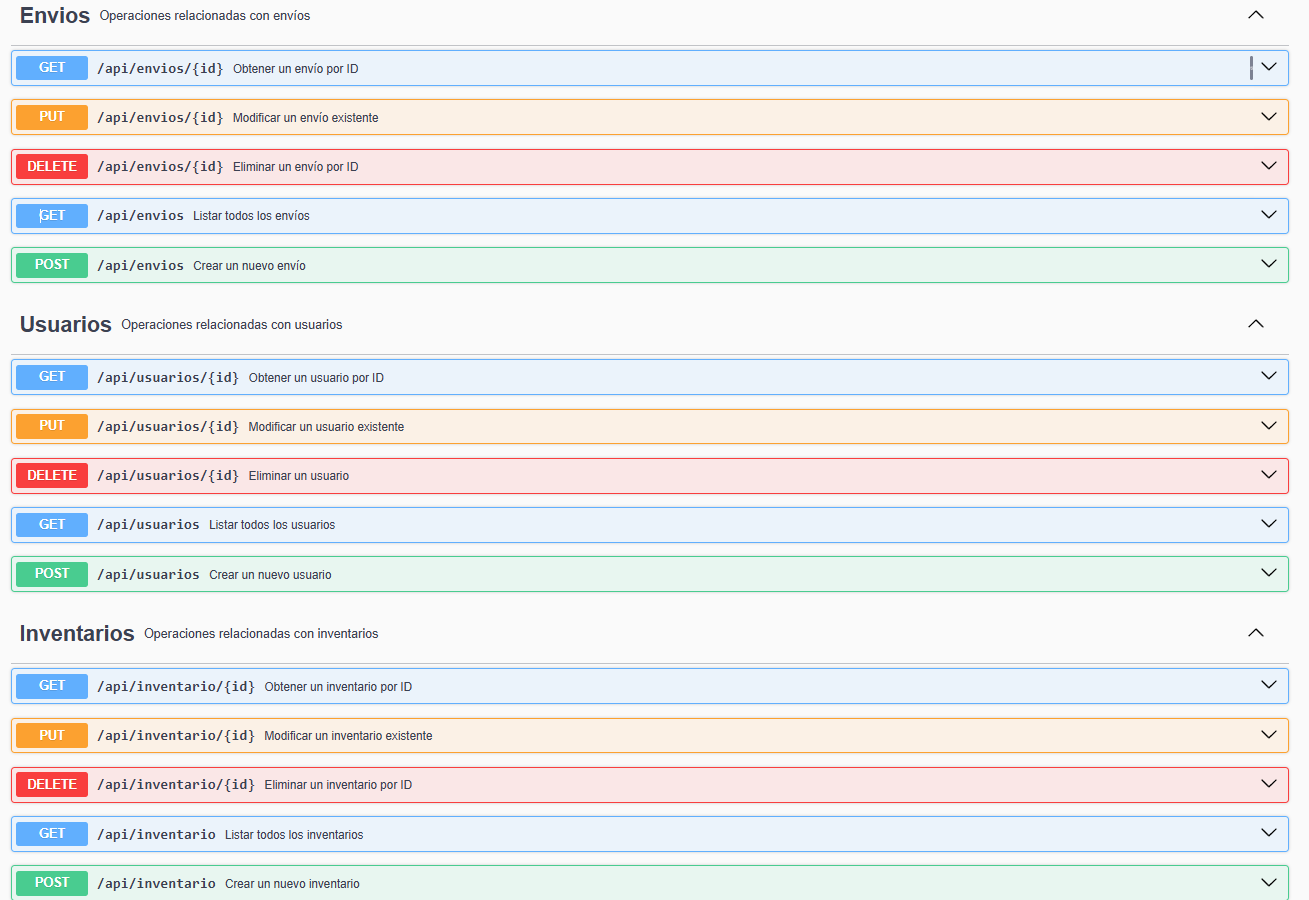
### **Evidencias de ejecución**

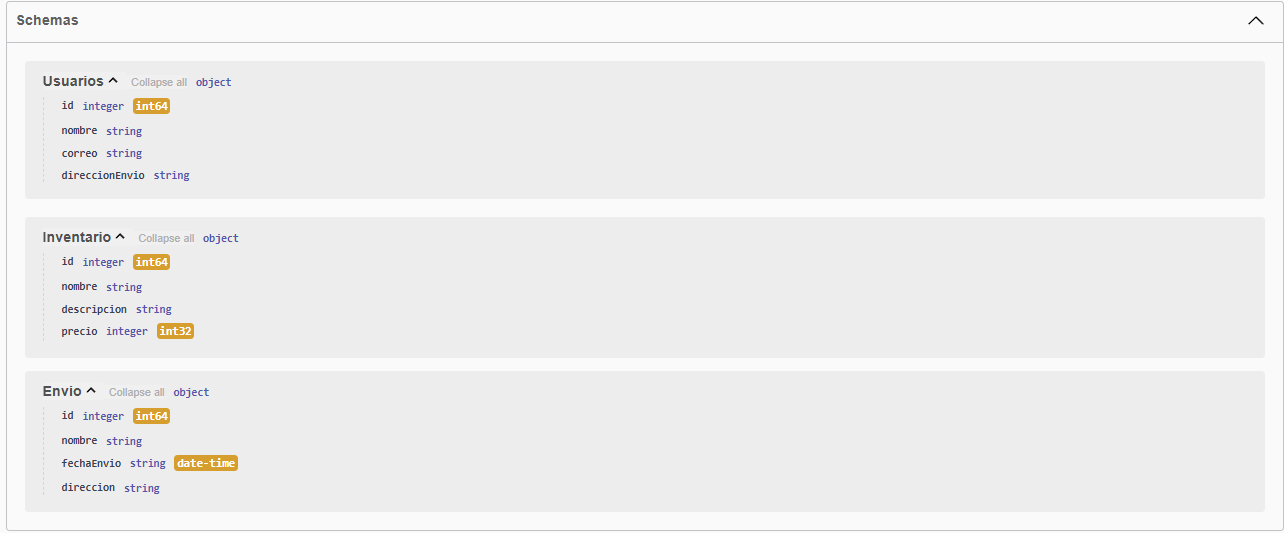
#### Imágenes de resultados exitosos o fallidos

#### Cobertura alcanzada

La cobertura realizada y alcanzada de estas pruebas unitarias y de integración alcanzan el 100% de éxito.

#### Documentación por Swagger

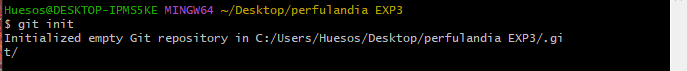




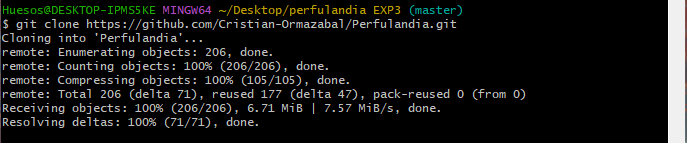
## **Uso de git y Github**

### **Comandos utilizados**

* Git init: es utilizado para inicializar un nuevo repositorio Git en una carpeta local



* Git clone: Es utilizado para clonar un repositorio remoto de Git de forma local



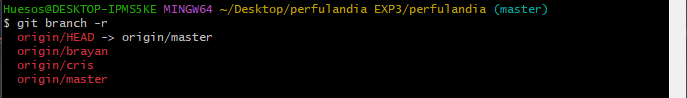
* cd “Perfulandia”: Es utilizado para navegar de un directorio a otro



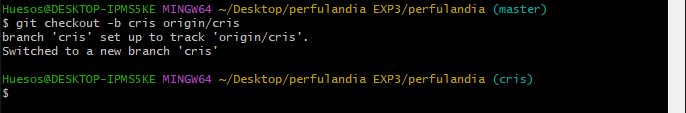
* Git branch: Muestra todas las ramas locales disponible en el repositorio



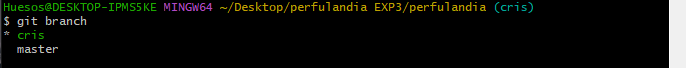
* Git branch -r : Realiza un listado de todas las ramas disponibles en el repositorio



* Git checkout -b : Crea una nueva rama de manera local copiando una rama remota del repositorio y te cambia a ella



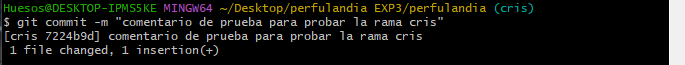
* Git branch: Muestra todas las ramas locales disponible en el repositorio



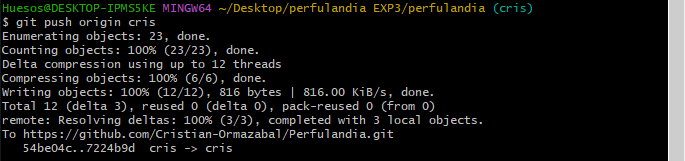
* Git add . : Permite agregar todos los archivos locales a una area de preparacion



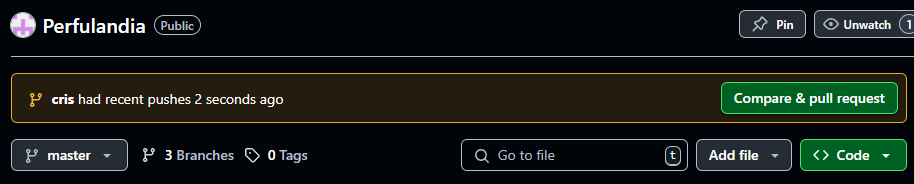
* Git commit -m: Permite guardar los cambios locales empaquetados con un comentario descriptivo



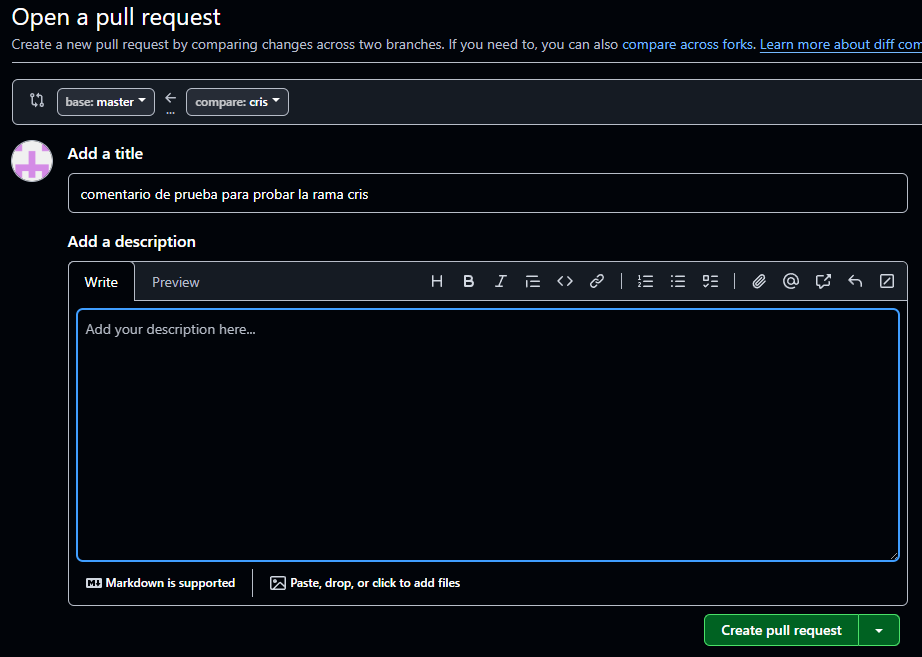
* Git push origin cris: Se envía el commit local a la rama del repositorio con el nombre de cris en Github

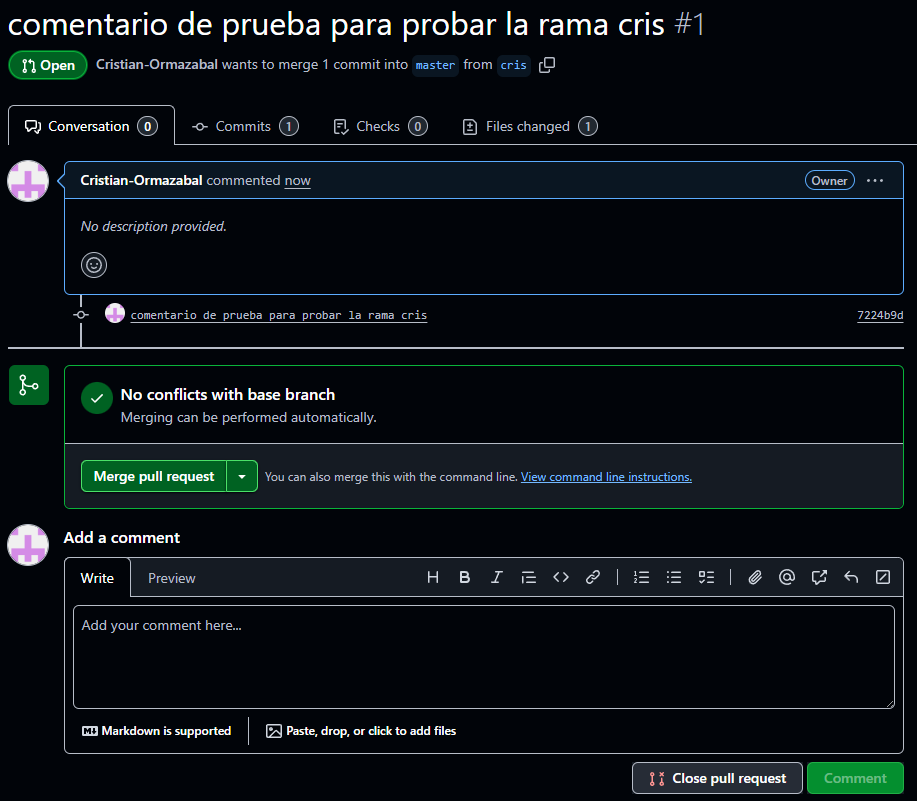


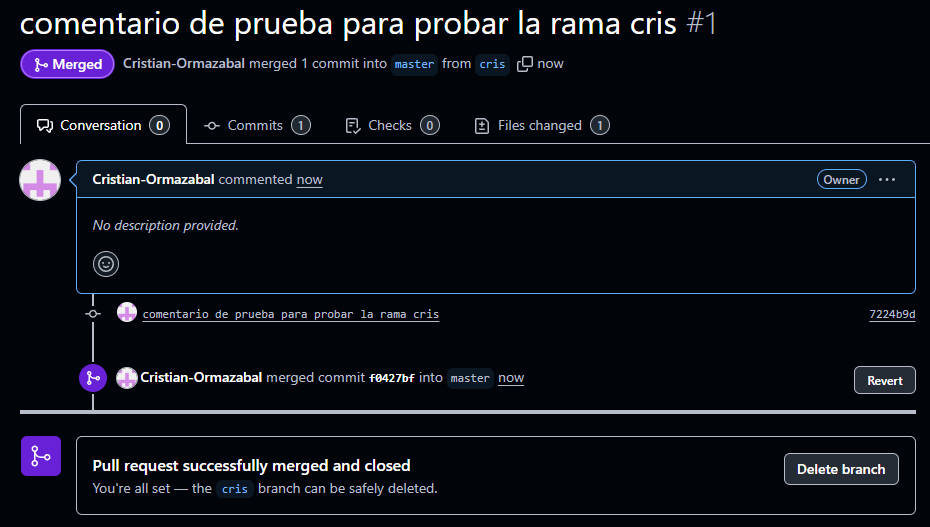
* Aquí aparece el push realizado por la rama cris



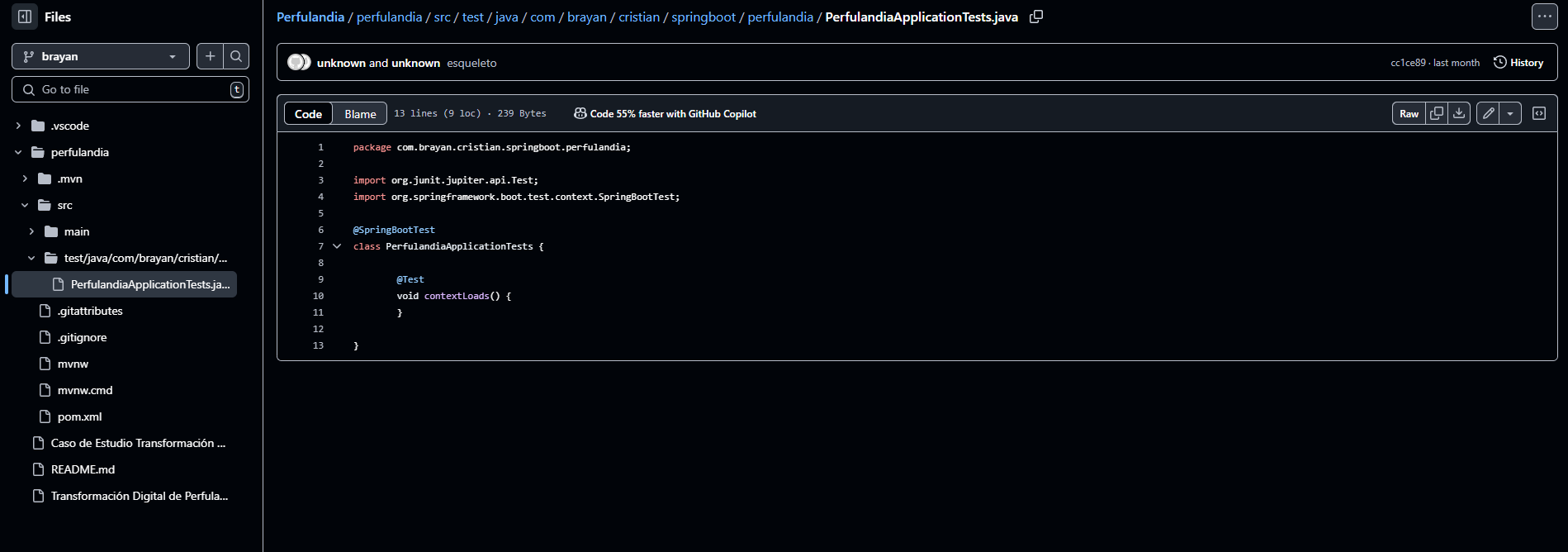
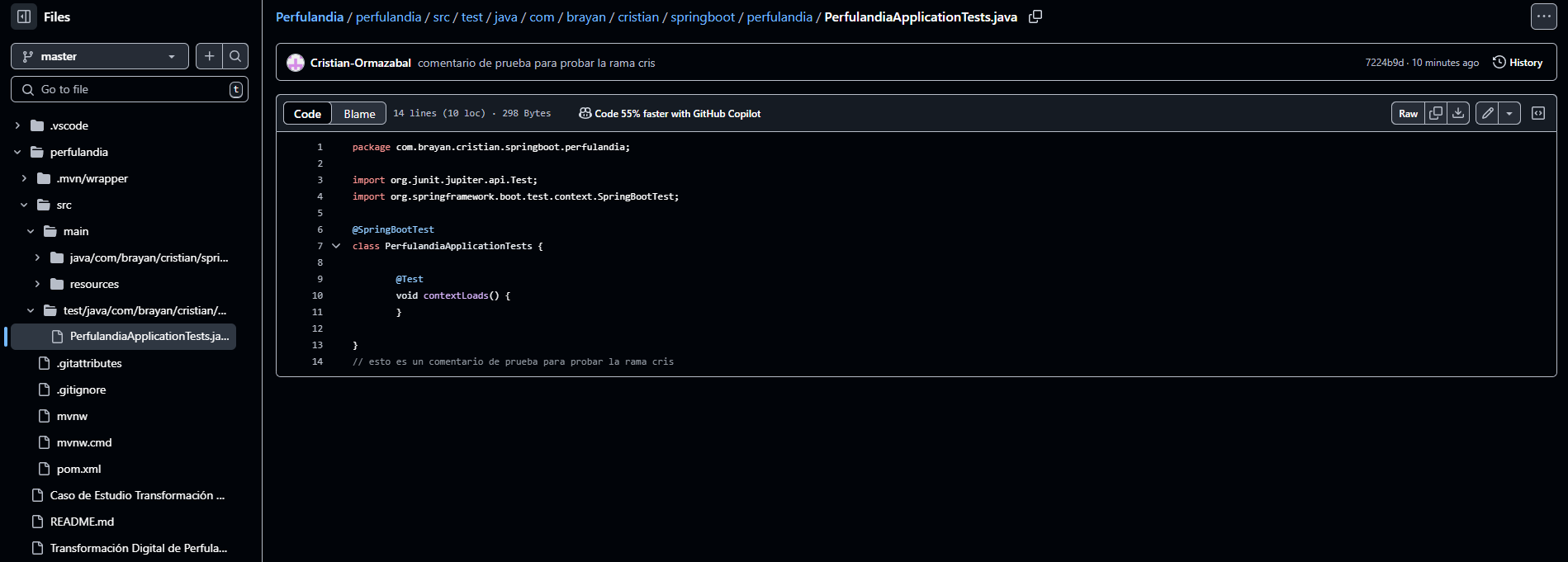
* Desde aquí esta el proceso a realizar que permite una revision del codigo antes de fusionar con la rama principal, ayudando a mantener la integridad del proyecto y favoreciendo el trabajo en paralelo mediante las ramas







* Comparativa entre la rama principal y la rama brayan, donde se muestra que el cambio solo afecta a una rama y no a todas



### **Estructura del repositorio**

#### Descripción de carpetas Exp1, Exp2Exp3

La estructura del repositorio para esta entrega se realizó con carpetas separadas por experiencias, donde se evidencia el proceso desde el inicio mostrando el cambio y evolución de este caso de estudios. Dentro del repositorio se encuentra

* Experiencia 1: Contiene una presentación y un informe inicial de los requisitos funcionales y no funcionales del cliente, además de presentar una propuesta de microservicios para su negocio que le permite una mejora continua a su sistema.
* Experiencia 2: Contiene el primer esqueleto del software realizado en java para el cliente con los microservicios de usuarios, inventario y ventas. Se incluye una presentación y un informe evidenciando los diagramas de microservicios, la estructura correspondiente al proyecto y ejecución de Postman.
* Experiencia 3: Contiene el software realizado en java para el cliente con los microservicios de usuarios, inventario y ventas. Además, sus respectivas pruebas unitarias, pruebas de integración y documentación de las pruebas. Asimismo se incluye una presentación y un informe con evidencia sobre las respectivas pruebas del software e información sobre las branch ocupadas en esta tercera entrega del repositorio

#### Rama utilizada

En este repositorio se encuentran tres branch, una principal (master) y dos de desarrollo (brayan y cris). Esta estructura tiene como finalidad facilitar el trabajo independiente de desarrolladores y a su vez puedan trabajar de manera colaborativa entre ellos. Además, se permite subir a la master solo los cambios revisados y autorizados por el team leader evitando subir errores a la branch maestra por accidente.

## **Conclusión**

### **Reflexión general del trabajo realizado**

El desarrollo de este trabajo permitió desarrollar habilidades y conocimiento prácticos en programación sobre el uso de la arquitectura de microservicios, explorando el uso de diversas herramientas que ayudaron a lograr el desarrollo de este ciclo de vida del software.

### **Principales aprendizajes**

Entre los principales objetivos que se adquirieron están:

* La aplicación de microservicios, permitiendo dividir el sistema en módulos más escalables y sostenibles.
* El uso básico y correcto de diversas herramientas de desarrollo, como frameworks, dependencias o entornos de pruebas.
* El conocimiento sobre el trabajo colaborativo que Github permite para desarrollar de manera independiente pero a su vez en conjunto con otros desarrolladores
* El uso eficiente de herramientas de testing, que permite asegurar la calidad y funcionalidad del sistema a través de diversas pruebas