

Eco Market

Integrantes: Miguel Laguela

Rodrigo Valdes

Matias Torres

Asignatura: Full Stack 1

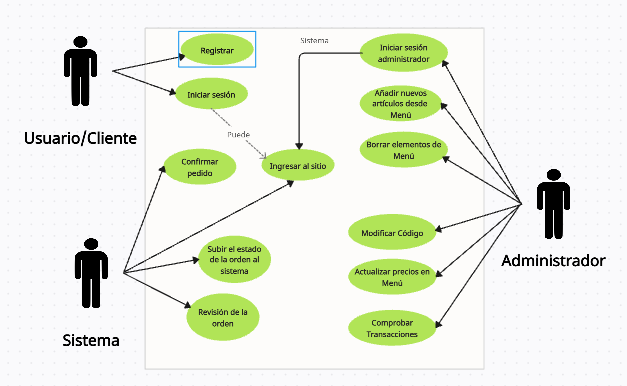
Fecha entrega: 06/04/2025

**Índice**

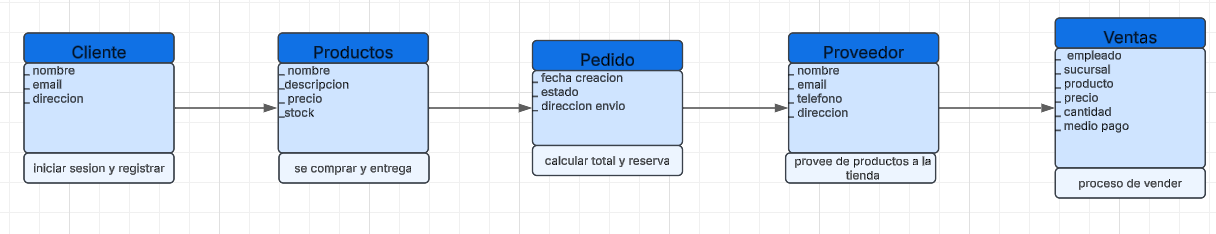
1. **Diagrama de Arquitectura ……………………...….. page 3**
2. **Plan de Pruebas …………………………….. page 4**
3. **Ejecución de Pruebas …………………………….. page 9**
4. **GitHub………………………………………...…….…..page 10**
5. **Conclusion………………………………………….…..page 11**

**1. Diagrama de microservicio:**

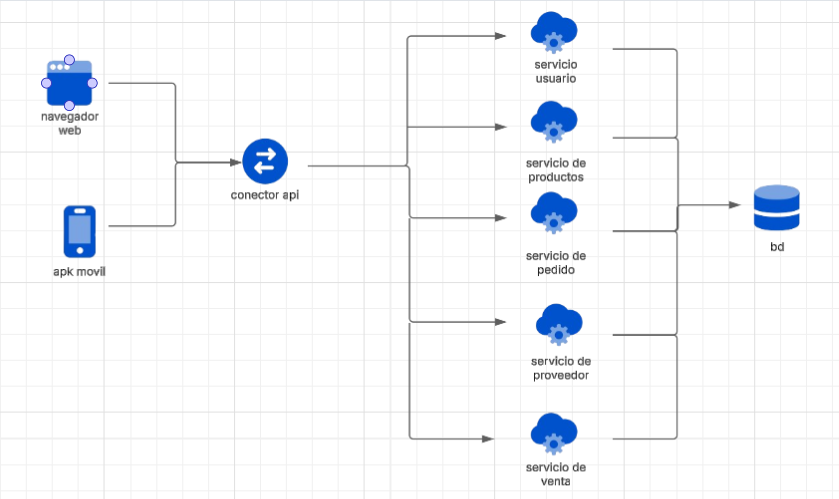
* Diagrama de Casos de Uso



* Diagrama de clases



* Diagrama de Despliegue

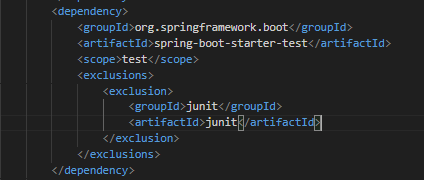


**2. Plan de Pruebas:**

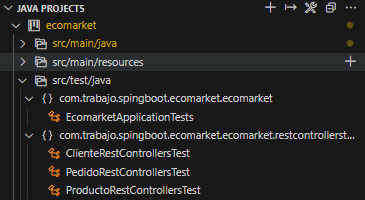
Los framework ocupados son: Junit version 5, Spring Boot Test, Mockito, AutoConfigureMockMvc y mas adelante encontrarán una explicación de cada framework

Documentación Pruebas Integración:

dentro del pom agregamos la dependencia junit el cual es version 5

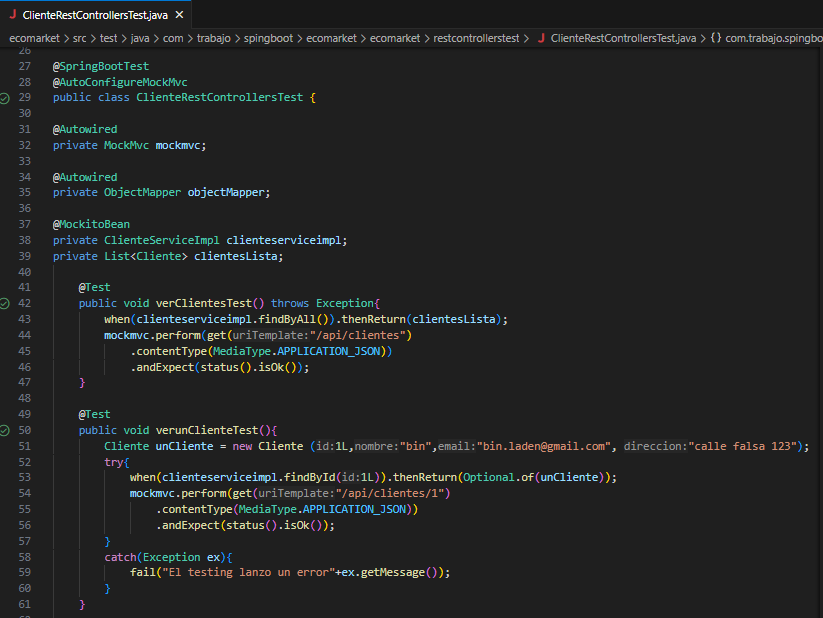


Aquí dentro de la ruta src/test/java se crea un package dentro del package ecomarket. Aca creamos el package con el sufijo restcontrollerstest que sería el paquete donde contendremos nuestras clases testing

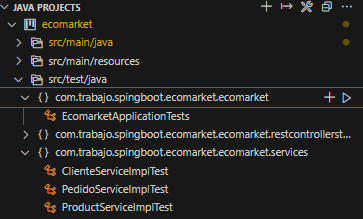


arriba del todo están las importancias necesarias para que funcione el sistema.

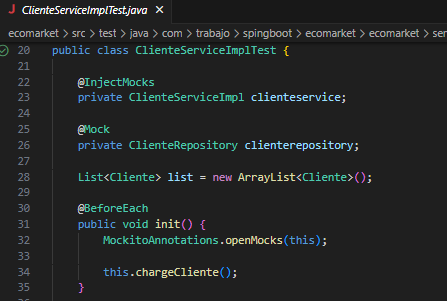
Posteriormente ocupamos SpringBootTest como modulo de apoyo al testing, MockitoBean el cual se usa para mockear (objetos simulados) servicios, repositorio o alguna otra dependencia, when definir comportamiento de un método dentro de mock, perform es usado dentro de las peticiones http. Este proceso es repetido por cada controlador.



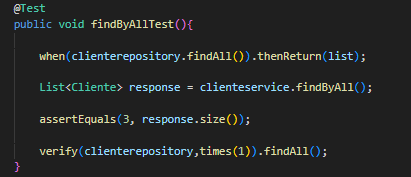
Dentro de la ruta src/test/java y dentro de este package creamos las clases ClienteServiceImplTest, PedidoServiceImplTest, ProductServiceImplTest



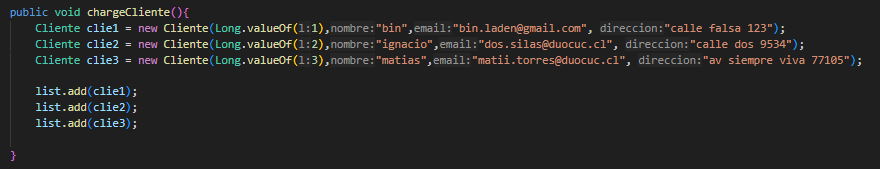
Dentro de ClienteServiceImplTest le creamos dos atributos el private service y repository según corresponda como se ve en la siguiente imagen y duplicamos la clase clienterepository con el uso de @mock. Además le ponemos @Injectmock esta escanea todas las clase que poseen anotación mock, cabe agregar el uso @before que abra los archivos mock con tal de hacerle configuraciones necesarias



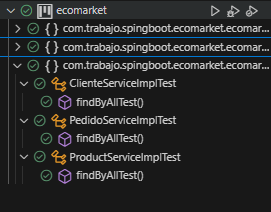
Método para buscar dentro del crud. Utilizamos mandos de Mockito(when y thenReturn, el cual nos ayuda llamar el repositorio con findAll este devolverá una lista y finalizar con la creación del objeto dentro de la lista



Método de llenado de la lista



finalmente ejecutamos las pruebas, en cada una procedimiento similar, pero con un llenado de lista diferente

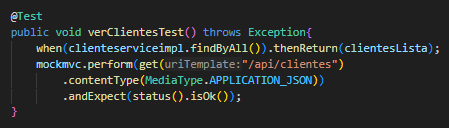


Pruebas Unitarias:

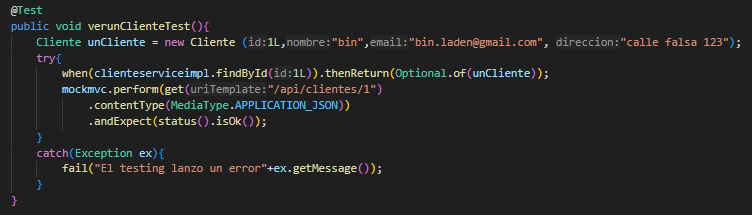
Los uso de framework en esta prueba son @SpringBootTest el cual nos añade test para ocupar en testing, @AutoConfigureMockMvc deja añadir mock. El que sera util para accesar las peticiones del controlador Rest

Seguimos con la implementación de métodos especiales para testing los cuales permitirán modificar o eliminar

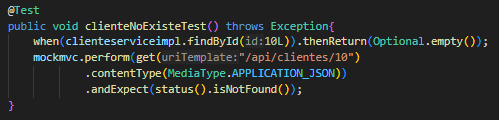
Método service.findByAll



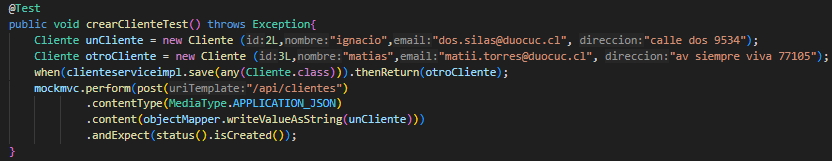
Método service.findById



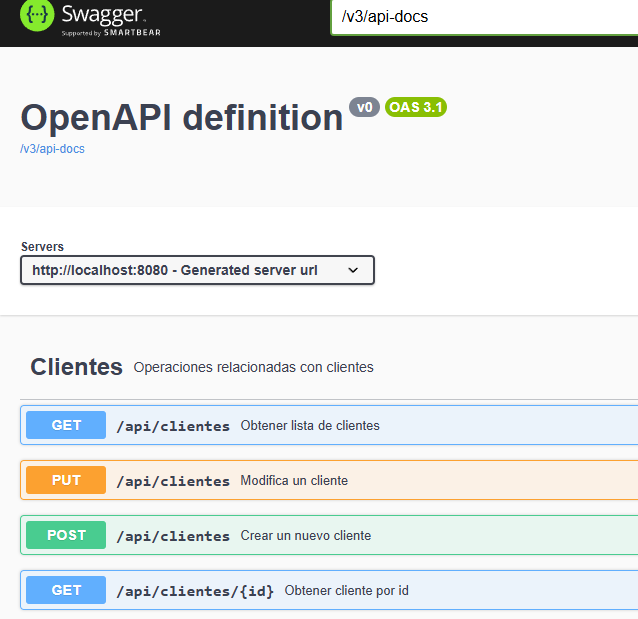
metodo clienteNoExisteTest



metodo crearClienteTest

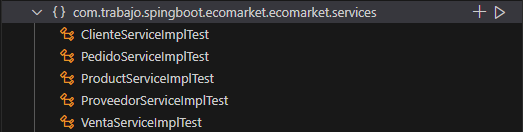


Preparamos la documentación necesaria para ejecucion de swagger

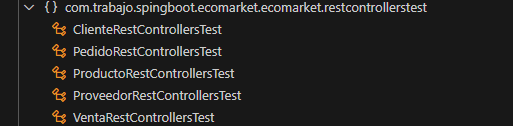


**3. Ejecución de Pruebas**

Pruebas unitarias: ejemplo -> (todas las 19 pruebas unitarias salieron correctamente dentro de los microservicios de Cliente, Pedido, Producto, Proveedor, Venta pasaron exitosamente el testing



Pruebas integracion: ejemplo -> (dentro del proceso cada prueba de integracion para los controladores en general se pudieron completar sin problemas

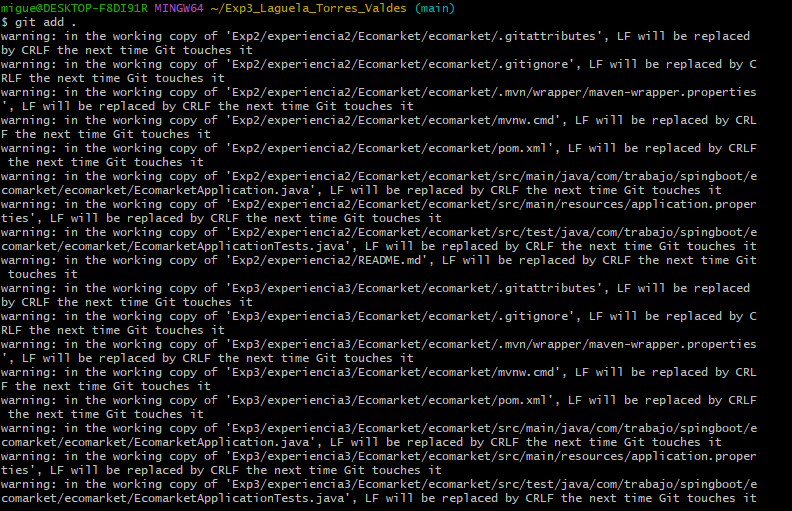


**4. Git hub:**

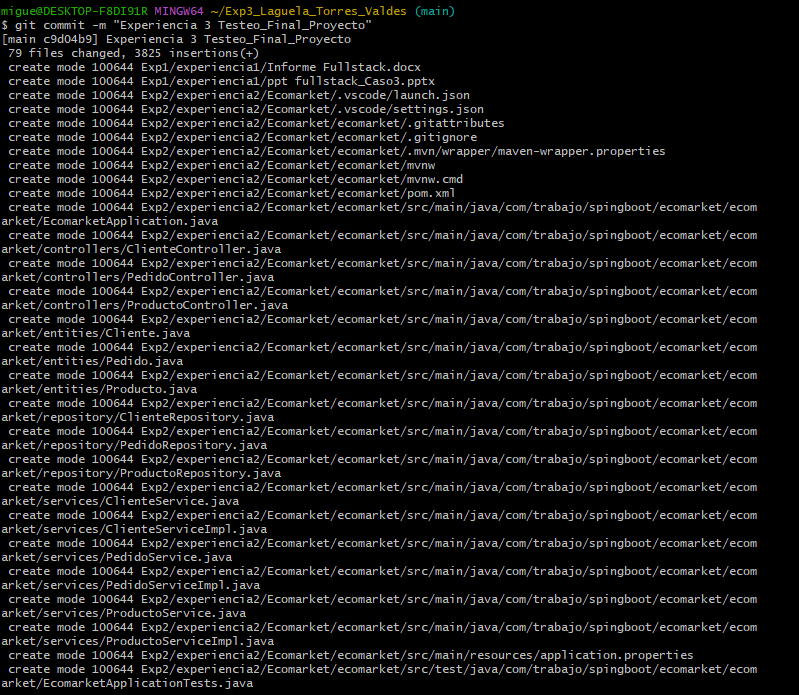
github es para usarlo en la colaboración de codigo ademas de subirlo al repositorio remoto alojado en GitHub

git init -> Inicia repositorio de forma local en git

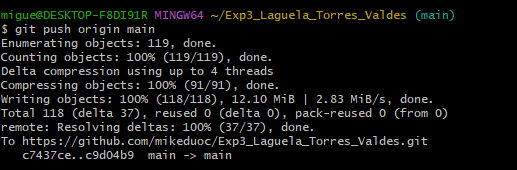
git add -> agrega los archivos dentro del repositorio

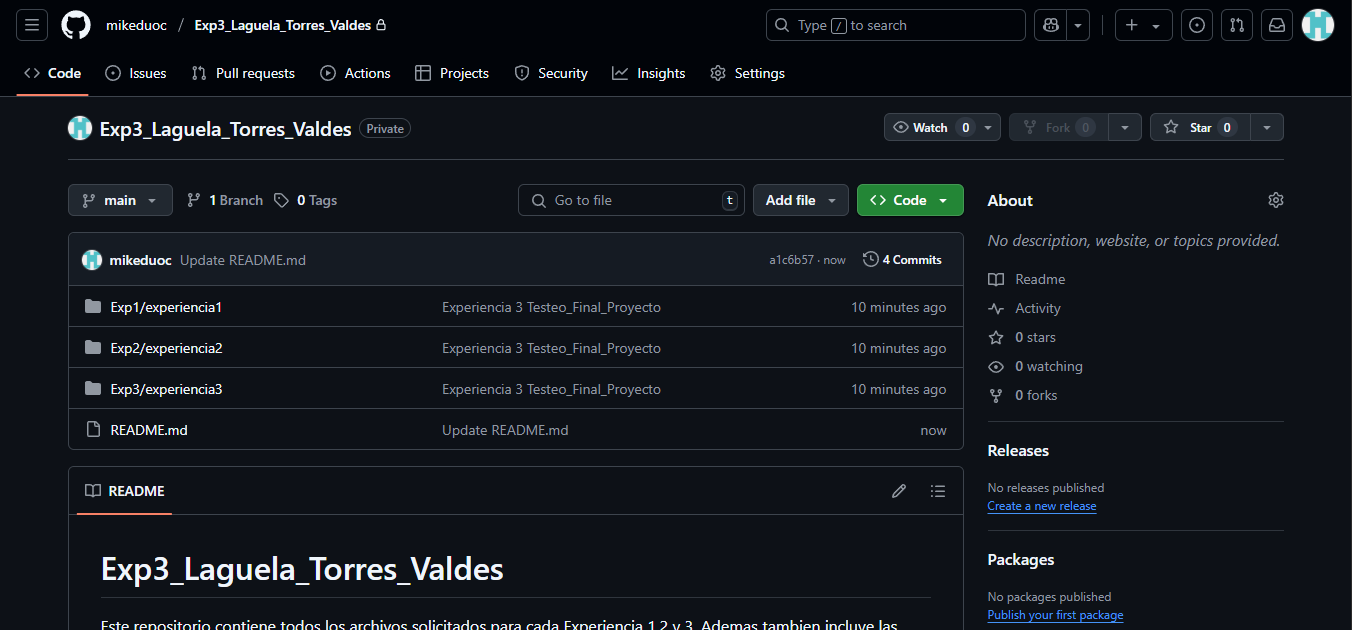


git commit -m -> guarda los cambios total de forma local en el repositorio



git push origin main -> sube los cambios del commit al repositorio local de github





**5. Conclusión:**

En esta tercera etapa sobre el proyecto de mejora del proceso de microservicios dentro de la compañía chilena EcoMarket Spa, se pudo lograr una arquitectura de microservicios usando Spring Boot y MySql WorkBench. Con estas se pudo lograr una mejoría en la escalabilidad y la resiliencia del sistema, esto ayudó a facilitar el despliegue y mantenimiento.

El uso de pruebas unitarias y de integración fue importante en el ámbito de la calidad del código y una mejora entra la conexión de microservicios en el proyecto. Para esto se utilizó implementaciones como Junit, Mockito y Spring Boot Test.

Por otra parte, la utilización de GitHub dejó mantener un uso útil de las versiones y la plataforma permite alojar el proyecto de forma que se pudiera ir modificando constantemente y guardando los cambios, además ayudó a interconectar el sistema y tener cada fase del proyecto alojada.

Para finalizar, el proceso de microservicio es realmente util para mantener un constante crecimiento de forma estable, además ayuda a entablar un piso, que permita el mejoramiento próximo del proyecto de negocio de EcoMarket Spa y eso finalmente llevará a aumentar la comunicación entre cada microservicio, mejorará la seguridad,rendimiento y disponibilidad ,reduciendo la probabilidad de fallos más adelante.