**1. Planificación de la Aplicación (Arquitectura)**

● Tarea: Crear un diagrama de arquitectura para la plataforma de comercio electrónico. Los candidatos deben decidir entre utilizar microservicios tradicionales, funciones serverless (por ejemplo, AWS Lambda) o una combinación de ambos. La arquitectura debe enfocarse en el backend y puede incluir recursos en la nube simulados mediante LocalStack u otra herramienta.

● Entregable: Un diagrama de arquitectura (puede ser una imagen o documento) que detalle los componentes y su interacción.

Diagrama

Descripción generada automáticamente

**2. Diseño en la Nube**

● Objetivo: Con la arquitectura como base, simular o esbozar teóricamente cómo se implementaría tu diseño en un entorno de nube. Puedes elegir el método que mejor muestre tus habilidades y conocimientos.

● Entregable: Código e instrucciones para ejecutar tu simulación local, junto con una explicación detallada de cómo funcionaría en un entorno real en la nube.

**Preparación Inicial**:

* Crear una cuenta AWS
* Instalar AWS CLI en tu computadora
* Instalar Docker en tu computadora

**Bases de Datos**:

* Crear 2 instancias de RDS Oracle:
  + Una para autenticación (BD Auth)
  + Otra para los servicios (mi base de datos BD Ecommerce)
* Usar la versión más simple (free tier si es posible)

Free Tier Es como un "período de prueba gratuito" que ofrece Amazon Web Services (AWS) donde puedes:

* Probar servicios gratis
* Aprender cómo funcionan
* Crear proyectos pequeños
* Experimentar sin preocuparte por costos

ECS (Elastic Container Service) quien se encarga de Hacer que todas estas partes funcionen juntas y Asegurarse de que la tienda nunca se caiga

**Cluster ECS**:

* Crear un cluster ECS llamado "ecommerce"

**Para cada servicio Spring Boot**:

* Crear un Dockerfile simple:

**Contenedores**:

* Crear un repositorio en ECR para cada servicio

**ECR** es donde guardamos todas las partes de los servicios

* El sistema de login
* El catálogo de productos
* El carrito de compras
* El sistema de pagos

Imagina que una VPC es como un edificio privado de oficinas:

**El Edificio (VPC)**

* Es tu espacio privado y seguro en la nube
* Nadie puede entrar sin permiso

**Configuración de red**:

* Usar una VPC por defecto para empezar
* Los servicios en la misma VPC pueden comunicarse entre sí

Para la el ecommerce tendria

**Área Pública**

* + Tu API Gateway
  + Donde los clientes hacen sus peticiones

**Área Privada**

* + Tus servicios Spring Boot (como los empleados trabajando)
  + Tus bases de datos Oracle (como la caja fuerte)

**Despliegue**: Para cada servicio

**3. Backend - Implementación Completa de CRUD**

● Tarea: Implementar el CRUD completo para dos servicios backend clave:

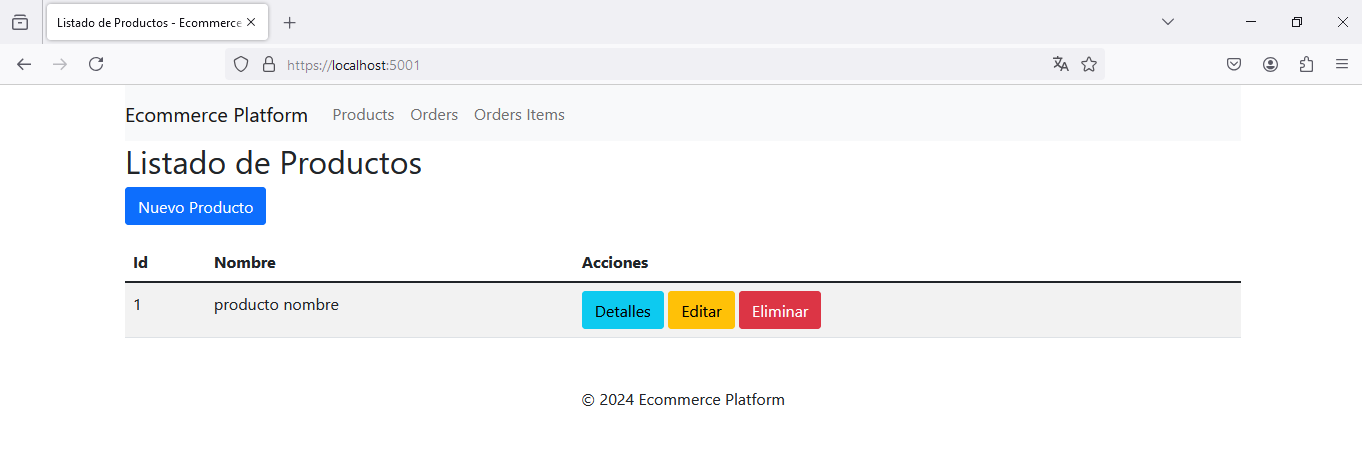
○ Product Catalog Service: Gestión completa del catálogo de

productos.

○ Order Management Service: Gestión completa de los pedidos.

Los candidatos deberán implementar las operaciones completas de Crear, Leer, Actualizar y Eliminar (CRUD) para cada uno de estos servicios. Se espera que utilices .NET (preferiblemente con Entity Framework para acceso a datos) o Java Reactivo (utilizando frameworks como Spring WebFlux). Es importante que el código siga principios SOLID y buenas prácticas de desarrollo. Entregable: Código fuente de los servicios backend, siguiendo principios RESTful y utilizando una arquitectura limpia o hexagonal.

Nota importante:se deja el código fuente dentro de la carpeta así mismo el script de creación de base de datos en este caso se utilizo mysql y la contraseña:5294 y el usuario root tener en cuenta esto para realizar las pruebas



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**4. Documentación**

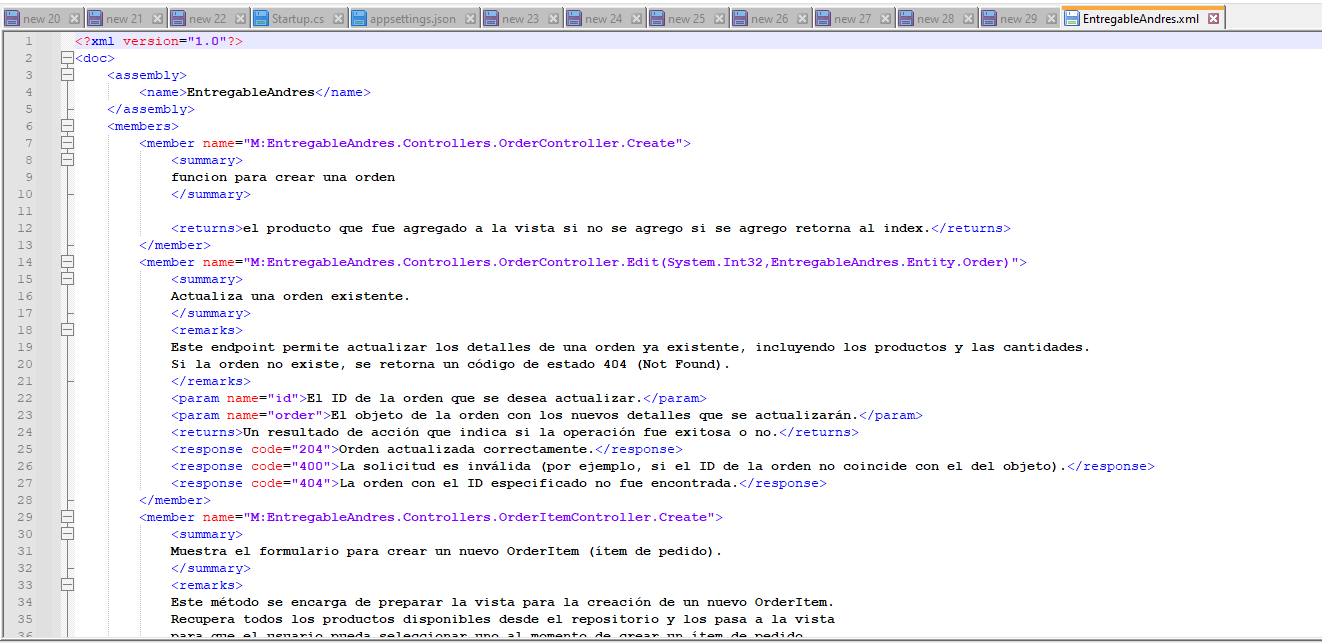
● Tarea: Generar documentación para los servicios utilizando Swagger

(OpenAPI) o Postman. Asegúrate de que los endpoints estén claramente

documentados para facilitar su comprensión y uso.

● Entregable: Un enlace o archivo que permita acceder a la documentación

generada, junto con instrucciones en el README sobre cómo visualizarla.



**5 Pruebas Unitarias**

● Tarea: Implementar pruebas unitarias que cubran las funcionalidades

principales de los servicios backend. Si estás utilizando Java, utiliza JUnit y

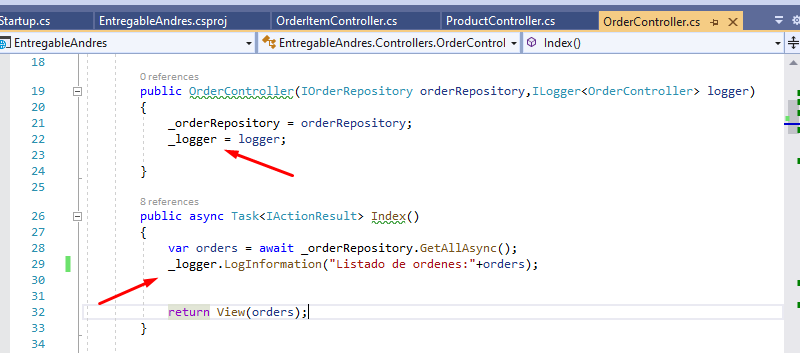
Mockito para tus pruebas. Si estás utilizando .NET, utiliza XUnit y considera

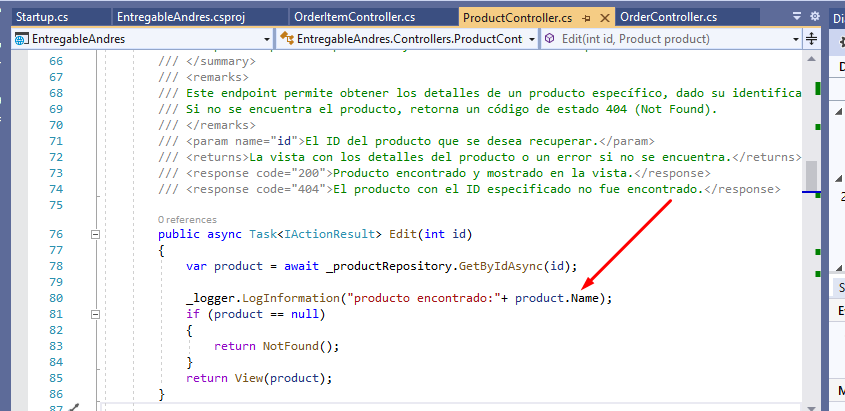
el uso de Entity Framework InMemory para pruebas de integración.

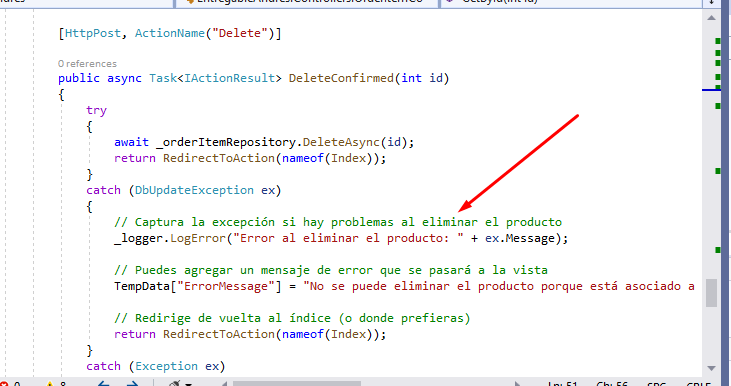
● Entregable: Código fuente con las pruebas unitarias implementadas, junto

con instrucciones en el README sobre cómo ejecutarlas.

Nota importante: para las pruebas unitarias se implemento logs imprimiendo los valores que iban llegando y de acuerdo a eso se fueron agregando condicionales y excepciones controladas







**6. Manejo de Errores**

Tarea: Implementar un manejo de errores robusto en tus servicios, asegurándote

de que los errores se manejen adecuadamente y se proporcionen respuestas

claras y significativas a los clientes de la API. Considera el uso de excepciones

personalizadas y el registro (logging) de errores para facilitar la depuración.

Entregable: Código que demuestre el manejo adecuado de errores, junto con

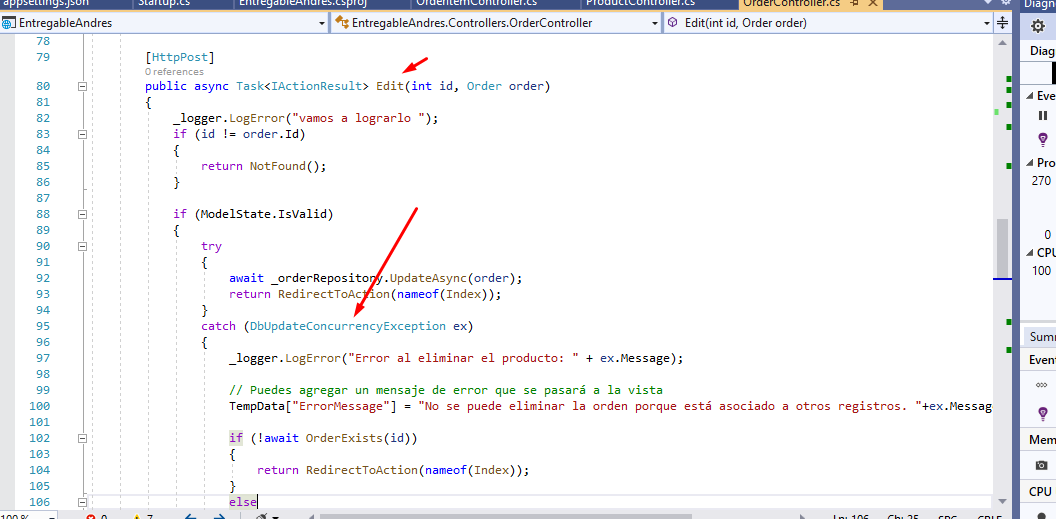
ejemplos en la documentación o en el README de cómo se manejan diferentes

tipos de errores.

**Solucion**

Para el manejo de excepciones se implementaron dos para manejar llaves foranes y alguna excepción con el retorno de un carácter no valido

**DbUpdateConcurrencyException->** se utiliza para atrapar la excepción durante el llamado a la base de datos



Y se utilizo Exception que es la excepción general para atrapar toda clase de excepción pero en este caso se resalta el uso de DbUpdateConcurrencyException para controlar actualizaciones en la base de datos y no devolver un mensaje de error a la vista esto se implemento tanto para producto como para orderItem

**7. Dockerización**

Tarea: Incluir un Dockerfile que permita ejecutar el servicio backend dentro de

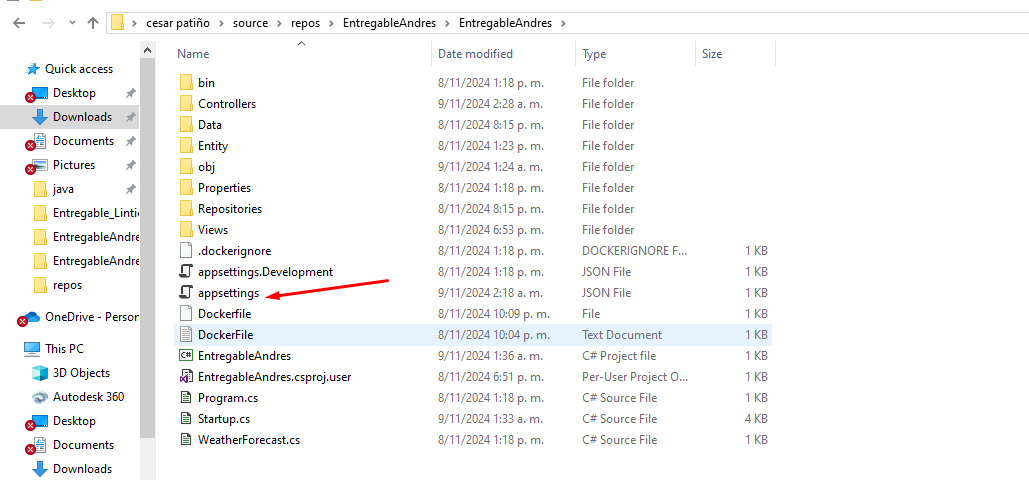
un contenedor Docker. Este paso es de gran importancia para que los

entrevistadores puedan ejecutar la aplicación sin inconvenientes. Proporciona

instrucciones claras en el archivo README sobre cómo construir y ejecutar la

aplicación usando Docker, especificando todos los comandos necesarios paso a

paso.



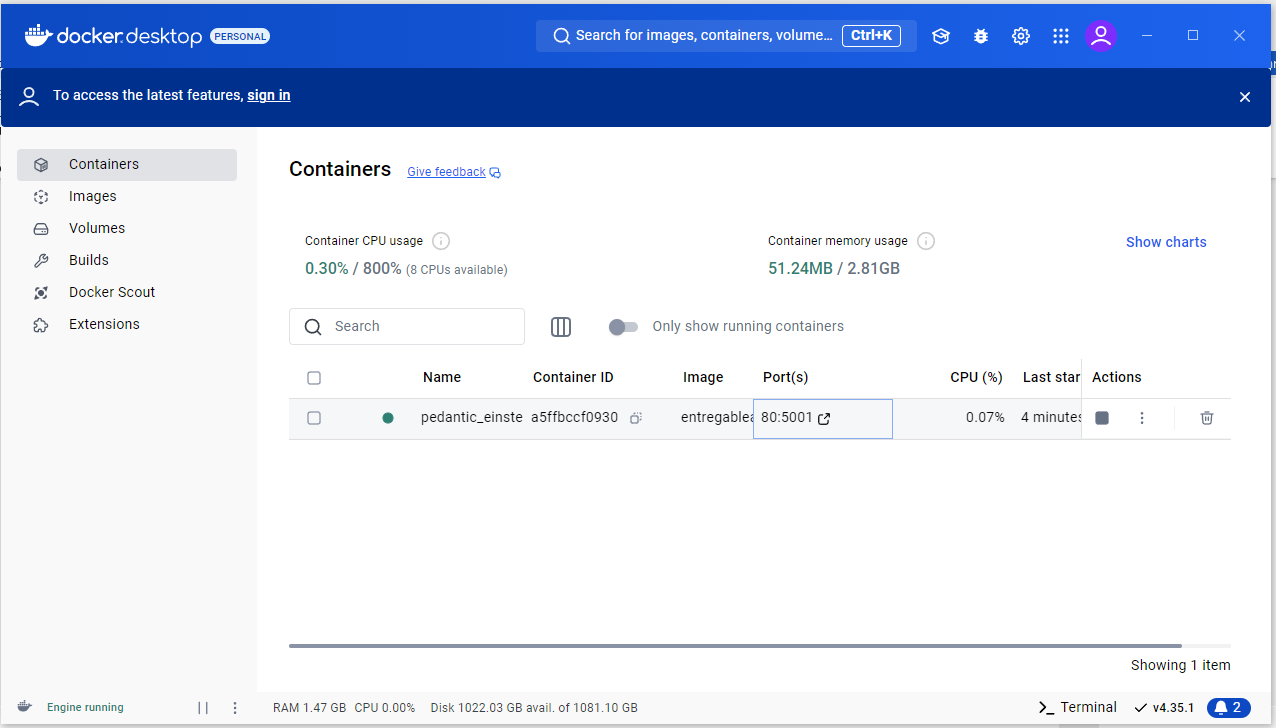
En este paso es muy importante configurar la conexión si se va a usar Docker para que quede como Docker.internal también es importante que en user y password se modifique el archivo appsetting.json que esta dentro del proyecto adjunto

"ConnectionStrings": {

"EcommerceDatabase": "Server=host.docker.internal;Database=entregableAndres;User=root;Password=5294;"

}

Y si se va a usar en local se debe dejar en server por localhost, lo mismo el usuario y la clave antes de correr la aplicación esto porque no se uso variables de entorno



Y muestro que esta ejecutándose la aplicación

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

El archivo dockerFile se deja en el proyecto solución

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Primer paso**

docker build -t entregableandres:v1.0.1 .

**segundo paso**

**docker run -p 80:5001 -e ASPNETCORE\_URLS="http://+:5001" entregableandres:v1.0.1**

**tercer paso si es opcional**

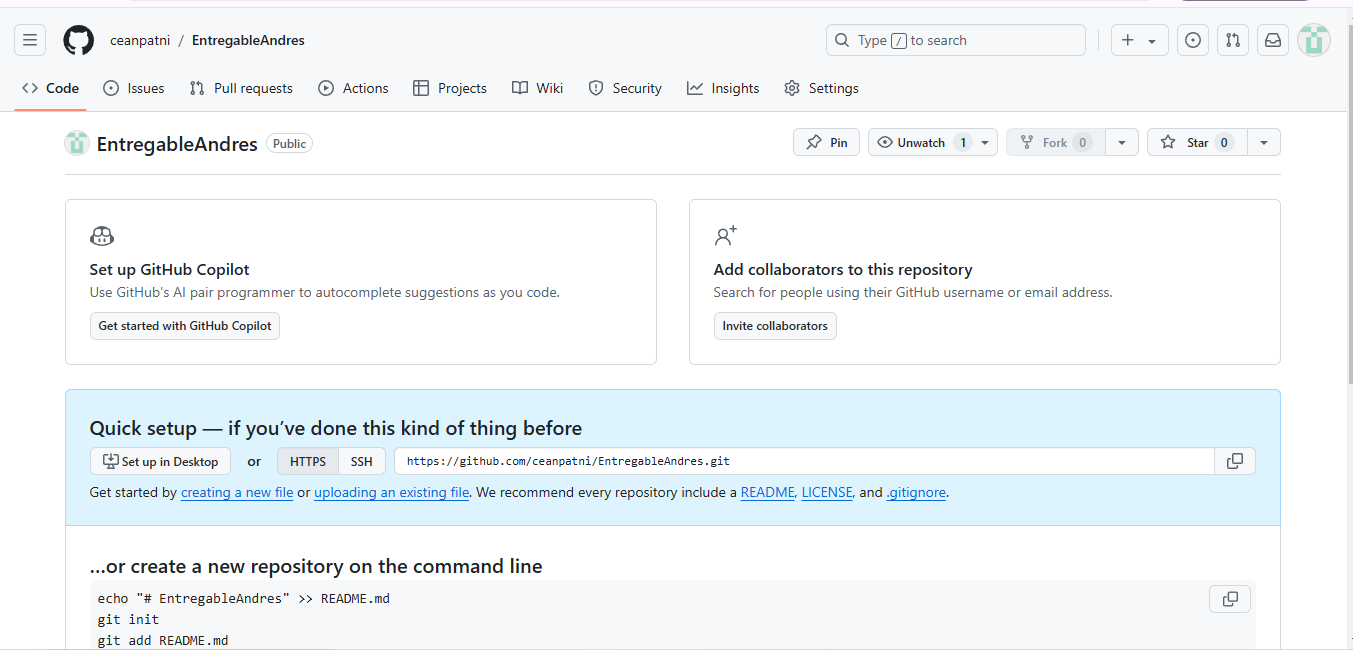
docker save -o entregableandres\_v1.0.1.tar entregableandres:v1.0.1

**Envio:**

● Sube tu código a un repositorio de GitHub.

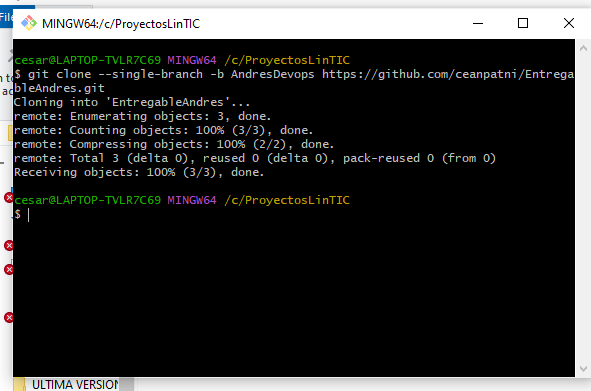
● Incluye un archivo README con INSTRUCCIONES CLARAS Y

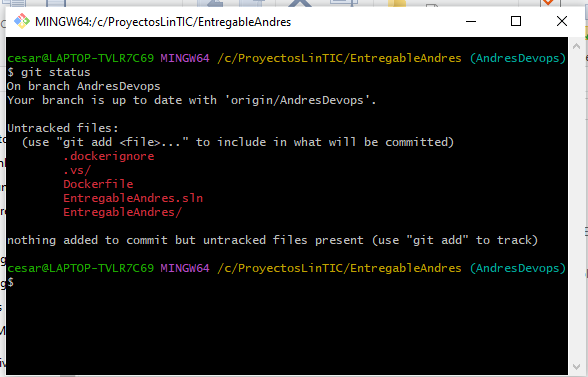
DETALLADAS sobre cómo configurar y ejecutar la aplicación.



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente





Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Adjunto url donde se evidencia cambios y donde se sincroniza con la MAIN

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

https://github.com/ceanpatni/EntregableAndres/tree/main