Taller Célula – Sistema de Gestión de Clientes y Pedidos

Objetivo General

Desarrollar una **API RESTful completa** utilizando **.NET 8**, aplicando **arquitectura por capas (DDD)** y **Entity Framework Core**, que permita gestionar clientes y los pedidos que estos realizan.

🏢 Contexto

Una empresa necesita un sistema básico para administrar sus **clientes** y los **pedidos** que estos hacen.
Cada pedido puede tener **varios productos**, una **fecha de creación** y un **estado** que indica su progreso (por ejemplo: *Pendiente*, *Enviado* o *Cancelado*).

Tu tarea será crear la base de este sistema, estructurado de forma limpia y mantenible.

Requisitos Técnicos

1️⃣ Arquitectura por Capas (DDD)

El proyecto debe estar dividido en **4 capas** principales:

- **Api** → Contendrá los controladores y endpoints.
- **Application** → Contendrá los servicios (lógica de negocio).
- **Domain** → Contendrá las entidades (modelos del dominio).
- **Infrastructure** → Contendrá la configuración de Entity Framework Core, el `DbContext` y los repositorios.

2️⃣ Base de Datos

- Se debe utilizar **Entity Framework Core**.
- Puede usarse **SQLite** o **SQL Server LocalDB**.
- Debe existir una **migración inicial** (`dotnet ef migrations add InitialCreate`).
- Debe generarse la base de datos con `dotnet ef database update`.

3️⃣ Entidades Mínimas

Debes implementar al menos las siguientes **entidades**:

- **Customer**
  - Id (int)
  - Name (string)
  - Email (string)
- **Order**
  - Id (int)
  - CustomerId (int)
  - OrderDate (DateTime)
  - Status (string) → valores posibles: *Pendiente*, *Enviado*, *Cancelado*
- **OrderDetail**
  - Id (int)
  - OrderId (int)
  - ProductName (string)
  - Quantity (int)
  - UnitPrice (double)

*(Opcionalmente puedes agregar la entidad **Product** para ampliar la práctica).*

5️⃣ Buenas Prácticas y Evaluación

✅ El código debe seguir los principios de **arquitectura limpia**.
✅ Los controladores **no deben tener lógica de negocio**, solo delegar a los servicios.
✅ Los servicios deben interactuar con los **repositorios** definidos en la capa **Infrastructure**.
✅ La base de datos debe ser persistente y funcional.
✅ Se deben poder probar los endpoints con **Postman**.