Taller de Práctica: Modelado de Dominio y Persistencia con TypeORM (Puro)

Información del Taller

Campo	Detalle
Carrera	Software
Nivel	Quinto
Asignatura	Aplicación para el Servidor Web
Docente	John Cevallos
Período Lectivo	2025-2026(1)
Número de Taller	3
Paralelos	A y B
Fecha/Horas	Octubre 2025 (2 horas académicas)
Tecnología	**Node.js con TypeScript (TypeORM Puro)**

Tema y Objetivo

- **Tema:** Persistencia del Dominio y Lógica de Acceso a Datos utilizando TypeORM sin Frameworks (Node.js/TypeScript).
- **Objetivo:** Modelar el dominio completo del proyecto autónomo definiendo **entidades** y **relaciones** robustas. Implementar la conexión y la lógica **CRUD** directamente a través del **DataSource** de TypeORM. El resultado se probará mediante un *script* de *seeding* y se visualizará en un Diagrama Entidad-Relación (DER).

Modalidad de Trabajo y Distribución del Dominio

El trabajo es en **grupos de 3 personas**. Cada integrante es responsable del modelado y la lógica de datos de sus entidades asignadas:

Integrante	Entidades Asignadas (Ejemplo)	Foco Principal
Integrante 1	Entidades Maestras (Catálogos, Configuraciones)	Definición de Datos Estáticos
Integrante 2	Entidades de Negocio Principal (Clientes, Productos)	Lógica de Negocio y Relaciones Clave
Integrante 3	Entidades Transaccionales (Ventas, Compras)	Manejo de Datos Dinámicos y Relaciones Múltiples

Instrucciones Detalladas (TypeORM Puro)

La implementación se centrará en las capas de **Entidad** y **Servicio/Lógica de Negocio** (clases de TypeScript) que interactúan con el **DataSource** de TypeORM. **No se implementarán Controladores ni Módulos de NestJS.**

1. Configuración del Proyecto y Conexión

- Crear un proyecto **Node.js/TypeScript** e iniciar sus repositorios **Git** individual.
- Instalar dependencias clave: `typeorm`, el *driver* para el SGBD elegido (ej. `sqlite3`).
- Definir el **DataSource** de TypeORM (ej. en `src/data-source.ts`) para centralizar la configuración de la base de datos (tipo, entidades).

2. Desarrollo del Modelo de Dominio (Entidad)

- **Definir Entidades:** Crear la clase *Entity* para cada recurso.
- **Propiedades:** Incluir la clave primaria autoincremental (`@PrimaryGeneratedColumn()`) y definir al menos **4 propiedades** (columnas) con decoradores TypeORM apropiados (`@Column`, `@Entity`).
- **Implementar Relaciones:** Establecer las relaciones (One-to-Many, Many-to-Many, etc.) entre las entidades del dominio utilizando los decoradores de TypeORM.

3. Lógica de Persistencia (Servicio/Repositorio)

- **Crear Clases de Servicio:** Desarrollar una clase TypeScript (`<Entidad>Service.ts`) que encapsule la lógica CRUD.
- **Acceso a Repositorio:** El servicio debe obtener el repositorio utilizando el **DataSource** inicializado.
- **Implementar CRUD:** La clase de Servicio debe implementar los **5 métodos CRUD obligatorios**:
  - `create(data)`: Para guardar una nueva entidad.
  - `findAll()`: Para obtener todos los registros.
  - `findOne(id)`: Para obtener un registro por su ID.

    - update(id, data): Para actualizar un registro.
    - remove(id): Para eliminar un registro.

4. Prueba Funcional y Seeding

- **Crear Script Principal:** Crear un *script* ejecutable (src/main.ts) que sea el punto de entrada para probar la aplicación.
- **Inicializar Conexión:** Dentro de main.ts, inicializar el **DataSource** de TypeORM.
- **Implementar el Seed de Datos:** El *script* debe obtener los **Servicios** e insertar datos de prueba significativos, **demostrando la correcta implementación de las relaciones** y la lógica de create().
- **Probar el CRUD:** Ejecutar y registrar llamadas a findAll(), findOne(), update(), y remove() de los Servicios para probar la funcionalidad completa.

5. Generación del Diagrama Entidad-Relación (DER)

- Utilizar una herramienta o *plugin* (ej. TypeORM CLI) para generar la **visualización del esquema** de la base de datos a partir de las entidades TypeORM definidas.

Entregables del Taller

1. **Repositorio Git Individual**:
    - Código fuente completo con todas las **Entidades** y **Servicios** implementados en TypeScript.
    - *Commits* por integrante del equipo.
2. **Archivo README.md**:
    - Documentación de **todas las entidades y sus relaciones**.
    - Instrucciones de instalación y ejecución del *Script* de *Seeding* (main.ts).
3. **Diagrama Entidad-Relación (DER):**
    - Imagen o captura del esquema de base de datos generado a partir del modelo TypeORM.
4. **Demostración del Seeding y CRUD:**
    - La ejecución del *script* de *Seeding* y la prueba de las operaciones CRUD deben ser demostradas al docente **de forma presencial** durante la revisión. No se requieren capturas de pantalla de la consola.