COrders

Sistema de Órdenes Dinámicas

Aplicación web **drag & drop** para la creación de órdenes dinámicas orientadas a negocios de servicios y ventas.
El sistema permite construir formularios personalizados desde el frontend y almacenar las órdenes completas en el backend sin restricciones estructurales.

Backend – Base de Datos

Este repositorio incluye el diseño y la documentación del módulo **Backend – Base de Datos**, encargado de almacenar órdenes dinámicas utilizando **PostgreSQL** y el tipo de dato **JSONB**.

El objetivo principal del módulo es **guardar formularios dinámicos completos** sin normalizar campos ni interpretar la estructura del formulario.

Objetivo del Diseño

Diseñar una base de datos que permita:

- Almacenar formularios dinámicos completos
- Soportar cambios en la estructura del frontend sin modificar la base de datos
- Evitar la normalización de campos
- Evitar lógica de negocio en la base de datos
- Mantener un diseño flexible y escalable

> **Regla fundamental:**
> Si el frontend cambia la estructura del formulario, la base de datos **no requiere modificaciones**.

Enfoque Técnico

- **Gestor de base de datos:** PostgreSQL
- **Tipo de dato principal:** JSONB
- **Entorno:** PostgreSQL local (Windows)
- **Acceso:** `psql` desde PowerShell o VS Code
- **Docker:** Opcional

La base de datos **no valida ni interpreta** el contenido del formulario.
La validación y estructura del JSON son responsabilidad del **Backend-API**.

Estructura del Módulo Database

```
database/
├── migrations/
│   ├── 001_create_orders.sql
│   └── 002_indexes.sql
├── seeds/
│   └── 001_sample_order.sql
└── docs/
    ├── BACKEND_DATABASE_GUIDE.md
    ├── queries.md
    └── contract.md
```

Modelo de Datos

La tabla principal del sistema es `orders`, encargada de almacenar cada orden generada desde el frontend.

Campo	Tipo	Descripción
`id`	UUID	Identificador único de la orden
`title`	VARCHAR	Nombre opcional de la orden
`schema`	JSONB	Formulario dinámico completo
`created_at`	TIMESTAMP	Fecha de creación de la orden
`updated_at`	TIMESTAMP	Fecha de última actualización

> **Nota:**
> El campo `updated_at` es gestionado por el **Backend-API**.
> No se utilizan triggers a nivel de base de datos.

El Campo `schema` (JSONB)

El campo `schema` almacena el formulario dinámico completo generado desde el frontend.

- No es un archivo
- No es una carpeta
- No es un esquema SQL
- No es una tabla adicional

Es únicamente **una columna JSONB** dentro de la tabla `orders`.

Uso de PostgreSQL desde VS Code (PowerShell)

Ruta típica de `psql`:

```
C:\Program Files\PostgreSQL\16\bin\psql.exe
```

Verificar instalación:

```
& "C:\Program Files\PostgreSQL\16\bin\psql.exe" --version
```

▶ Ejecución Paso a Paso

Crear base de datos

```
CREATE DATABASE corders;
```

Ejecutar migraciones

```
& "C:\Program Files\PostgreSQL\16\bin\psql.exe" -U postgres -d corders -f
migrations/001_create_orders.sql
& "C:\Program Files\PostgreSQL\16\bin\psql.exe" -U postgres -d corders -f
migrations/002_indexes.sql
```

Insertar datos de prueba

```
& "C:\Program Files\PostgreSQL\16\bin\psql.exe" -U postgres -d corders -f
seeds/001_sample_order.sql
```

🔎 Consultas SQL de Ejemplo

```
SELECT * FROM orders;

SELECT *
FROM orders
WHERE created_at::date = CURRENT_DATE;

SELECT *
FROM orders
WHERE schema->'fields' @> '[{"type":"text"}]';
```

Docker (Opcional)

El uso de Docker es opcional.
Este repositorio **no incluye** `docker-compose.yml`.

🏁 Conclusión

El módulo Backend – Base de Datos almacena órdenes dinámicas de forma **flexible, escalable y desacoplada** del frontend.

Toda la validación, interpretación del formulario y actualización de datos corresponde al **Backend-API**.