Order Builder

Descripción del proyecto

Order Builder es una aplicación web que permite crear plantillas de órdenes de pedido de forma visual mediante un sistema de drag and drop.

El objetivo es que cualquier negocio pueda diseñar su propio formato de orden sin necesidad de programar, construyendo la plantilla a partir de campos configurables (texto, número, fecha, selector, etc.).

Cada plantilla puede ser creada, editada, guardada y reutilizada.

⚠ Este proyecto NO procesa pedidos reales.
Su único propósito es definir, configurar y gestionar plantillas de órdenes.

Objetivo del proyecto

- Permitir la creación de plantillas de órdenes personalizadas
- Facilitar la configuración de campos mediante una interfaz visual
- Separar claramente la lógica de:
    - Construcción visual (Frontend)
    - Validación y persistencia (Backend)

El proyecto está diseñado bajo una arquitectura por responsabilidades, donde cada capa tiene límites claros.

Stack tecnológico

Frontend

- React
- Vite
- @dnd-kit (drag and drop)
- JavaScript (ES6+)

Backend *(en desarrollo)*

- Node.js
- Express

Base de datos *(en diseño)*

- PostgreSQL

Funcionalidades actuales

- Drag and drop de campos desde una paleta al canvas
- Reordenamiento de campos dentro del canvas
- Selección de campos
- Edición de propiedades básicas
- Eliminación de campos
- Renderizado dinámico de la plantilla

Funcionalidades en desarrollo

- Configuración avanzada por tipo de campo
- Persistencia de plantillas en base de datos
- API REST para gestión de plantillas
- Validaciones de estructura y negocio
- Versionado de plantillas
- Generación automática de plantillas mediante IA

Arquitectura de la aplicación

Frontend (React)

- Builder visual (drag & drop)
- Editor de campos
- Renderizado de plantillas
- Producción del contrato JSON

Backend (Node + Express)

- API REST
- Validación del contrato JSON
- Lógica de negocio
- Orquestación de persistencia

Base de datos (PostgreSQL)

- Persistencia de plantillas
- Versionado

- Integridad de datos

Estructura actual del proyecto

order-builder/

```
├─ src/
│  ├─ components/
│  │  ├─ FieldPalette.jsx
│  │  ├─ Canvas.jsx
│  │  ├─ SortableField.jsx
│  │  ├─ FieldEditor.jsx
│  ├─ App.jsx
│  └─ main.jsx
├─ package.json
└─ README.md
```

División de responsabilidades

Este proyecto se desarrolla bajo una arquitectura por responsabilidades.

👉 Cada módulo debe funcionar sin conocer la implementación interna de los demás.
👉 La comunicación entre capas se realiza únicamente mediante contratos de datos (JSON).
👉 No se permiten dependencias directas entre capas.

Roles definidos

- Arquitectura / Integración
- Frontend – Lógica de campos
- Frontend – UI/UX
- Backend – API
- Backend – Base de datos

Cada integrante trabaja exclusivamente dentro de su rol asignado.

Frontend – Builder (Drag & Drop)

Responsabilidades

- Interfaz de creación de plantillas
- Drag and drop de campos
- Edición visual de propiedades
- Gestión del estado local del Builder
- Producción del JSON de la plantilla

Puede hacer

- Crear componentes React
- Manipular estado de campos
- Reordenar campos mediante order
- Trabajar con datos mock (JSON)

NO debe hacer

- Llamar directamente a base de datos
- Definir reglas de negocio
- Validar reglas del backend
- Asumir cómo se guardan los datos

📌 El frontend solo construye y consume el contrato JSON.

Backend – API

Responsabilidades

- Recepción de plantillas desde el frontend
- Validación del contrato JSON
- Aplicación de reglas de negocio
- Exposición de endpoints REST

Puede hacer

- Validar estructura y tipos
- Rechazar JSON inválido
- Controlar versionado
- Orquestar persistencia

NO debe hacer

- Decidir cómo se ve la interfaz

- Depender de componentes del frontend
- Manipular drag and drop

📌 El backend no conoce React, solo conoce JSON válido.

Base de datos

Responsabilidades

- Persistencia de plantillas
- Versionado
- Integridad de datos

Puede hacer

- Diseñar tablas
- Crear relaciones
- Definir migraciones

NO debe hacer

- Lógica de negocio
- Validaciones de UI
- Asumir comportamiento del frontend

📌 La base de datos no interpreta significado, solo almacena datos.

Contrato de datos (JSON)

Las plantillas de órdenes se representan mediante un contrato JSON único, que define:

- Identidad de la plantilla
- Lista de campos
- Configuración de cada campo
- Orden visual

Reglas del contrato JSON

1. Todo campo debe tener id, type, order y props
2. props.label no puede ser vacío
3. El orden visual se determina únicamente por order
4. El frontend no valida reglas de negocio

5. El backend no asume comportamiento visual
6. Campos desconocidos deben ser rechazados por el backend

📄 El contrato JSON se define y mantiene en un documento separado.

Instalación y ejecución (Frontend)

Desde la ruta:

../COrders/order-builder

Ejecutar:

npm install

npm run dev

La aplicación se ejecuta por defecto en:

http://localhost:5173

Estado actual del proyecto

- El frontend base del Builder visual está funcionando
- El contrato JSON se encuentra definido y estabilizado
- El backend y la persistencia están en fase de diseño e implementación

---

Repositorio

📎 https://github.com/Diamond-01/COrders