**PROMPT — MVP CRM RBAC Kanban (NestJS + TypeORM + PostgreSQL 16 + React TS)**

**Rol del agente:** Actúa como arquitecto y dev full-stack senior. Entrega código listo para ejecutar (sin Docker), modular, seguro y escalable. Si hay ambigüedad, elige la opción más simple y documenta en 1 línea.

**0) Alcance del MVP (obligatorio)**

* **Auth**: login (JWT access + refresh), logout, refresh, hash **Argon2id**.
* **RBAC**: roles, permisos, módulos; decorador @Permissions() y RbacGuard.
* **Multicompañía**: companyId en todo dato de negocio.
* **Kanban**: proyectos, tableros, columnas, tareas; comentarios.
* **UI**: login → redirección a **tablero Kanban**; menú dinámico por permisos.
* **Operación local**: PostgreSQL 16 instalado localmente (sin Docker).

**1) Principios de escalabilidad (desde el día 1)**

* **Monolito Modular** (NestJS) con límites de dominio claros y **Ports & Adapters**:  
  domain/ (entidades, puertos), application/ (use-cases), infra/ (repos/adaptadores), ui/ (controllers).
* **Sin imports cruzados** entre dominios; comunicación via **interfaces (ports)**.
* **Schemas en PostgreSQL** por dominio (p. ej., auth\_rbac, crm), con **snake\_case** y PK id uuid.
* Preparado para extraer módulos a servicios independientes en el futuro (misma interfaz/port).

**2) Convenciones de nombres (estrictas)**

* **TypeScript**
  + Variables, funciones y métodos: **camelCase** → companyId, createTask, moveTaskToColumn.
  + Clases, DTOs, tipos y servicios: **PascalCase** → TasksService, CreateTaskDto.
  + Constantes: **UPPER\_SNAKE\_CASE** → DEFAULT\_PAGE\_SIZE.
  + Booleans con prefijos is/has/can/should → isActive, hasAccess.
  + Nombres de archivos: **kebab-case** → rbac.guard.ts, create-task.dto.ts.
* **SQL (PostgreSQL 16)**
  + Tablas/columnas: **snake\_case**, plural en tablas → users, role\_permissions, board\_column\_id.
  + Timestamps: created\_at, updated\_at; índices idx\_<tabla>\_<columna>.
* **REST**
  + Rutas: /api/v1/<recurso-plural> usando **kebab-case** → /api/v1/boards/:id/columns.

**3) Estructura del repo (monorepo pnpm)**

/apps/api # NestJS

src/

config/ # env + zod

common/ # decorators, guards, interceptors, pipes, utils

database/ # datasource, migrations, seeds

modules/

auth-rbac/

domain/ application/ infra/ ui/

crm/

projects/ boards/ tasks/ comments/

main.ts app.module.ts

/apps/web # React 18 + TS + Vite + Tailwind + shadcn/ui + React Query

/packages/shared # DTOs, tipos, (opcional) contratos OpenAPI

**4) Datos y migraciones (mínimo)**

* **Schemas**: auth\_rbac (usuarios, roles, permisos, módulos), crm (projects, boards, board\_columns, tasks, task\_comments).
* **Multitenancy**: company\_id en entidades de negocio; índices por (company\_id, …).
* **FK** con ON DELETE RESTRICT (o CASCADE solo donde tenga sentido).
* **Seeds idempotentes** por compañía:
  + Módulos (rbac, projects, boards, tasks).
  + Permisos por módulo (p. ej., tasks:create|read|update|move|delete).
  + Roles (DevAdmin, Admin, Manager, Contributor, Viewer) y asignaciones.
  + Usuario DevAdmin y datos Kanban demo (1 proyecto, 1 board, columnas To Do / Doing / Done, 3 tareas).

**5) Seguridad**

* **Argon2id** para passwords.
* **JWT**: access (15 min) + refresh (30 días) con **rotación**.
* **Guards**: JwtAuthGuard + RbacGuard; decorador @Permissions('module:action').
* **API hardening**: Helmet, CORS (lista blanca), rate limit, DTOs con class-validator.
* **Auditoría mínima**: tabla audit\_log para eventos críticos (cambios de rol/permiso, mover tarea).

**6) Frontend (React + TS)**

* **Login** → guarda tokens (memory o secure storage), fetch GET /me y GET /me/permissions.
* **Sidebar dinámico**: muestra solo módulos **activos** y permitidos.
* **Kanban**: drag & drop (HTML5 DnD u otra lib), operaciones condicionadas por permisos:
  + Crear tarea (tasks:create), mover (tasks:move), editar (tasks:update), comentar (tasks:comment).
* **Admin RBAC (solo DevAdmin)**: visible si rbac:\* y módulo con visibility=dev\_only (campo del módulo).

**7) Entregables (obligatorios)**

* Código de **API** y **Web** listo para correr local (sin Docker).
* **Migraciones** y **Seeds** reproducibles.
* **Colección Postman** o .http con: auth, me, rbac (roles/permissions/modules), kanban CRUD.
* **README** con pasos, variables .env, y comandos.
* **Pruebas e2e básicas**: login, permiso denegado/permitido, CRUD tarea.

**8) Comandos esperados (sin Docker)**

* Root:
  + pnpm dev:api / pnpm dev:web
  + pnpm migrate:run / pnpm migrate:revert / pnpm migrate:gen <Name>
  + pnpm seed
* Prepara .env.example (API) con: host 127.0.0.1, puerto 5432, crm\_db, crm\_user/crm\_pass, y secretos JWT.

**9) Ejemplos breves (respetar convenciones)**

**DTO (TS – camelCase, PascalCase para clase)**

export class CreateTaskDto {

title: string;

description?: string;

projectId: string; // uuid

boardId: string; // uuid

boardColumnId: string; // uuid

assignedTo?: string; // userId (uuid)

priority?: 'low' | 'medium' | 'high' | 'critical';

dueDate?: string; // ISO

}

**Service (métodos con verbo; scoping por companyId)**

@Injectable()

export class TasksService {

constructor(private readonly repo: Repository<TaskEntity>) {}

async createTask(companyId: string, userId: string, dto: CreateTaskDto) { /\* ... \*/ }

async moveTaskToColumn(companyId: string, taskId: string, targetColumnId: string) { /\* ... \*/ }

}

**Tabla (SQL – snake\_case, índices, timestamps)**

CREATE SCHEMA IF NOT EXISTS crm;

CREATE TABLE crm.tasks (

id uuid PRIMARY KEY DEFAULT gen\_random\_uuid(),

company\_id uuid NOT NULL,

project\_id uuid NOT NULL,

board\_id uuid NOT NULL,

board\_column\_id uuid NOT NULL,

title text NOT NULL,

description text,

priority text CHECK (priority IN ('low','medium','high','critical')) DEFAULT 'medium',

status text NOT NULL DEFAULT 'open',

assigned\_to uuid,

due\_date timestamptz,

created\_by uuid NOT NULL,

created\_at timestamptz NOT NULL DEFAULT now(),

updated\_at timestamptz NOT NULL DEFAULT now()

);

CREATE INDEX idx\_tasks\_company ON crm.tasks (company\_id);

CREATE INDEX idx\_tasks\_board\_column ON crm.tasks (board\_id, board\_column\_id);

**10) Criterios de aceptación (MVP)**

1. Puedo **logear** y ver mi **tablero Kanban** inicial.
2. El **menú** muestra solo módulos permitidos según mis **permisos**.
3. Puedo **crear**, **mover** y **editar** tareas si tengo permisos; si no, el backend **rechaza** (403) y el frontend deshabilita acciones.
4. **Seeds** crean compañía demo, usuario DevAdmin, roles/permisos y un board con 3 columnas.
5. El código cumple **naming**: camelCase (TS), snake\_case (SQL), rutas REST en kebab-case.
6. La arquitectura está lista para **extraer módulos** sin reescritura (ports/adapters y schemas separados).

**Modelo de Datos (mínimo requerido)**

-- Empresas

CREATE TABLE company (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

name TEXT NOT NULL

);

-- Usuarios

CREATE TABLE "user" (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

company\_id UUID NOT NULL REFERENCES company(id) ON DELETE CASCADE,

email TEXT UNIQUE NOT NULL,

name TEXT NOT NULL,

password\_hash TEXT NOT NULL

);

-- Roles

CREATE TABLE role (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

company\_id UUID NOT NULL REFERENCES company(id) ON DELETE CASCADE,

name TEXT NOT NULL,

description TEXT

);

-- Relación Usuario Rol

CREATE TABLE user\_role (

user\_id UUID NOT NULL REFERENCES "user"(id) ON DELETE CASCADE,

role\_id UUID NOT NULL REFERENCES role(id) ON DELETE CASCADE,

PRIMARY KEY (user\_id, role\_id)

);

-- Módulos (agrupadores de permisos)

CREATE TABLE module (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

company\_id UUID NOT NULL REFERENCES company(id) ON DELETE CASCADE,

name TEXT NOT NULL, -- ej: "Compras"

module\_id UUID REFERENCES module (id) ON DELETE CASCADE,

key TEXT NOT NULL, -- ej: "purchases"

status TEXT CHECK (status IN ('active', 'inactive'))

);

-- Permisos (acciones dentro de un módulo)

CREATE TABLE permission (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

module\_id UUID NOT NULL REFERENCES module(id) ON DELETE CASCADE,

name TEXT NOT NULL, -- ej: "approve\_order"

description TEXT

);

-- Relación Rol Permiso

CREATE TABLE role\_permission (

role\_id UUID NOT NULL REFERENCES role(id) ON DELETE CASCADE,

permission\_id UUID NOT NULL REFERENCES permission(id) ON DELETE CASCADE,

PRIMARY KEY (role\_id, permission\_id)

);