TRABAJO EVALUATIVO

Construcción de un CRM Básico con Express, TypeScript y Prisma

KATALINA GIRALDO RAMIREZ ERIKA YULIEH LADIDO

FICHA: 2901483

1 CONFIGURACION DEL ENTORNO:

```
"name": "trabajo",
"version": "1.0.0",
"main": "index.js",
"scripts": {
 "start": "node dist/index.ts",
  "dev": "nodemon src/index.ts"
},
"keywords": [],
"author": "",
"license": "ISC",
"dependencies": {
  "@prisma/client": "^6.6.0",
  "@types/express": "^5.0.1",
  "@types/node": "^22.15.2",
  "body-parser": "^2.2.0",
  "express": "^5.1.0",
  "nodemon": "^3.1.10",
  "prisma": "^6.6.0",
  "trabajo": "file:",
  "ts-node": "^10.9.2",
  "typescript": "^5.8.3"
"description": ""
```

2 CONFIGURAR SERVIDOR LOCAL

```
[nodemon] restarting due to changes...
[nodemon] starting `ts-node src/index.ts`
Bienvenido al servidor en http://localhost:3000
```

3 RUTAS Y PARAMETROS

```
import express, { Request, Response } from 'express';
import bodyParser from 'body-parser';
import {PrismaClient, Stage} from "@prisma/client";
const app = express();
const PORT = 3000;
app.use(bodyParser.json());
app.use(express.json());
const prisma = new PrismaClient()
app.get('/contact', async (req: Request, res: Response) => {
    //const stage = req.query.stage;
    const a = res.header("api_key", "hola")
   console.log(a);
    const contact = await prisma.contact.findMany();
    res.status(200).json({"ms": contact});
  });
// Ruta para buscar un contacto por ID
app.get('/contact/:id', async (req: Request, res: Response) => {
    const id = parseInt(req.params.id);
   try {
      const contact = await prisma.contact.findUnique({ where: { id } });
      if (contact) res.json(contact);
      else res.status(404).json({ error: 'Contacto no encontrado.' });
    } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Error al buscar el contacto.' });
  });
// Crear una nueva oportunidad
app.post('/Opportunity', async (req: Request, res: Response) => {
 const { name, description, estimatedValue, currency, stage,
expectedCloseDate, companyId } = req.body;
 try {
      const nuevaOportunidad = await prisma.opportunity.create({
          data: {
```

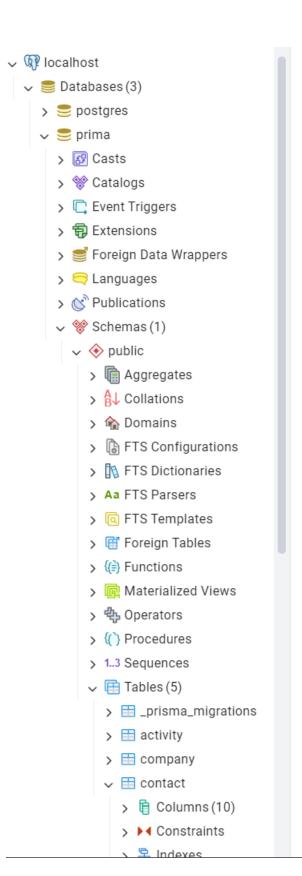
```
name,
              description,
              estimatedValue,
              currency,
              stage,
              expectedCloseDate: new Date(expectedCloseDate),
              companyId,
          },
      });
      res.status(201).json(nuevaOportunidad);
  } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Error al crear la oportunidad.' });
});
// Listar oportunidades por etapa (stage)
app.get('/listarpor', async (req: Request, res: Response) => {
 const { stage } = req.query;
 try {
     const oportunidades = await prisma.opportunity.findMany({
          where: { stage: stage as Stage },
      });
      res.json(oportunidades);
 } catch (error) {
      res.status(500).json({ error: 'No se pudieron obtener las
oportunidades.' });
 }
});
```

5 CONEXIÓN BD

```
# Environment variables declared in this file are automatically made available to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema
```

```
# Prisma supports the native connection string format for PostgreSQL, MySQL,
SQLite, SQL Server, MongoDB and CockroachDB.
# See the documentation for all the connection string options:
https://pris.ly/d/connection-strings

DATABASE_URL="postgresql://postgres:123456@localhost:5432/trabajo?schema=public"
```

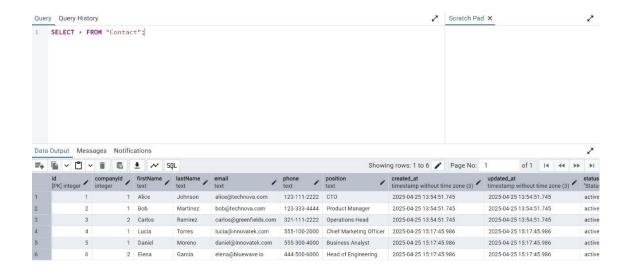


Tablas ----

Opprtunity

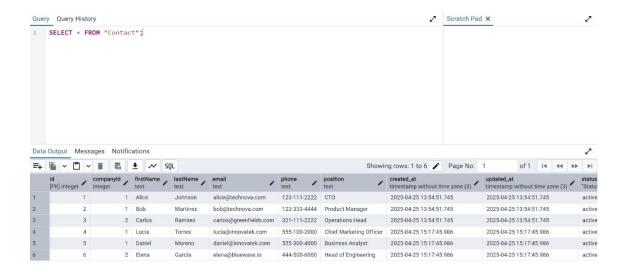


Contact

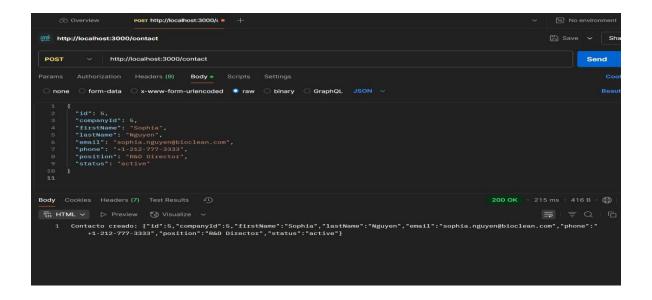


6 funcionalidad del orm

Consultar datos



Crear oportunidad



Lista de oportunidades

Respuestas http

```
app.listen(PORT, () => {
   console.log(`Bienvenido al servidor en http://localhost:${PORT}`);
});
```