 Crie um novo projeto Node.js com TypeScript.

bash

Copiar código

mkdir node-ts-prisma-project

cd node-ts-prisma-project

npm init -y

 **Instalar dependências:**

Instale as dependências necessárias para o Node.js, TypeScript, Prisma e PostgreSQL.

bash

Copiar código

npm install express @types/express prisma @prisma/client typescript ts-node-dev pg

Adicione as dependências de desenvolvimento:

bash

Copiar código

npm install --save-dev @types/node

 **Configuração do TypeScript:**

Crie o arquivo tsconfig.json com as seguintes configurações:

json

Copiar código

{

"compilerOptions": {

"target": "ES2020",

"module": "commonjs",

"strict": true,

"esModuleInterop": true,

"skipLibCheck": true,

"forceConsistentCasingInFileNames": true,

"outDir": "./dist"

},

"include": ["src"],

"exclude": ["node\_modules"]

}

 **Inicializar Prisma e PostgreSQL:**

Inicialize o Prisma no projeto:

bash

Copiar código

npx prisma init

Isso criará uma pasta prisma com o arquivo schema.prisma e um arquivo .env para a configuração do banco de dados. Configure a variável DATABASE\_URL no arquivo .env para se conectar ao seu banco de dados PostgreSQL:

makefile

Copiar código

DATABASE\_URL="postgresql://user:password@localhost:5432/mydatabase?schema=public"

 **Defina o modelo no Prisma:**

Edite o arquivo prisma/schema.prisma para refletir as entidades fornecidas:

prisma

Copiar código

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

generator client {

provider = "prisma-client-js"

}

model tb\_produto {

id Int @id @default(autoincrement())

descricao String

nome String

url\_imagem String

preco Float

itensPedido td\_item\_pedido[] // Relacionamento com os itens do pedido

}

model tb\_usuario {

id Int @id @default(autoincrement())

email String @unique

nome String

senha String

telefone String

data\_aniversario DateTime

pedidos tb\_pedido[] // Relacionamento com pedidos

}

model tb\_pedido {

id Int @id @default(autoincrement())

id\_usuario Int

status String

data DateTime

usuario tb\_usuario @relation(fields: [id\_usuario], references: [id])

itens td\_item\_pedido[]

}

model td\_item\_pedido {

id Int @id @default(autoincrement())

id\_pedido Int

id\_produto Int

qtde Int

preco Float

pedido tb\_pedido @relation(fields: [id\_pedido], references: [id])

produto tb\_produto @relation(fields: [id\_produto], references: [id])

}

 **Executar as migrações:**

Após configurar os modelos Prisma, crie as tabelas no banco de dados executando a migração:

bash

Copiar código

npx prisma migrate dev --name init

 **Gerar o cliente Prisma:**

Gere o cliente Prisma para que você possa interagir com o banco de dados.

bash

Copiar código

npx prisma generate

 **Criar a estrutura do projeto:**

Crie a estrutura de pastas do projeto:

bash

Copiar código

mkdir src

touch src/index.ts src/routes.ts src/controllers.ts

 **Implementação da API:**

* **Arquivo src/index.ts:**

typescript

Copiar código

import express from 'express';

import { routes } from './routes';

const app = express();

const port = 3000;

app.use(express.json());

app.use(routes);

app.listen(port, () => {

console.log(`Server running on http://localhost:${port}`);

});

* **Arquivo src/routes.ts:**

typescript

Copiar código

import { Router } from 'express';

import { createProduto, getProdutos, createUsuario, createPedido, getPedidos } from './controllers';

const routes = Router();

routes.get('/produtos', getProdutos);

routes.post('/produtos', createProduto);

routes.get('/usuarios', getUsuarios);

routes.post('/usuarios', createUsuario);

routes.post('/pedidos', createPedido);

routes.get('/pedidos', getPedidos);

export { routes };

* **Arquivo src/controllers.ts:**

typescript

Copiar código

import { PrismaClient } from '@prisma/client';

import { Request, Response } from 'express';

const prisma = new PrismaClient();

export const getProdutos = async (req: Request, res: Response) => {

const produtos = await prisma.tb\_produto.findMany();

res.json(produtos);

};

export const getUsuarios = async (req: Request, res: Response) => {

try {

const usuarios = await prisma.tb\_usuario.findMany();

res.json(usuarios);

}

catch (error) {

res.status(500).json({ error: 'Erro ao buscar usuários' });

}

};

export const createProduto = async (req: Request, res: Response) => {

const { descricao, nome, url\_imagem, preco } = req.body;

const produto = await prisma.tb\_produto.create({

data: { descricao, nome, url\_imagem, preco },

});

res.json(produto);

};

export const createUsuario = async (req: Request, res: Response) => {

const { email, nome, senha, telefone, data\_aniversario } = req.body;

const usuario = await prisma.tb\_usuario.create({

data: { email, nome, senha, telefone, data\_aniversario },

});

res.json(usuario);

};

export const createPedido = async (req: Request, res: Response) => {

const { id\_usuario, status, data, itens } = req.body;

try {

// Criar o pedido

const pedido = await prisma.tb\_pedido.create({

data: {

id\_usuario,

status,

data,

itens: {

create: itens.map((item: any) => ({

id\_produto: item.id\_produto,

qtde: item.qtde,

preco: item.preco

}))

}

},

include: {

itens: true

}

});

res.json(pedido);

} catch (error) {

res.status(500).json({ error: 'Erro ao criar pedido' });

}

};

export const getPedidos = async (req: Request, res: Response) => {

const pedidos = await prisma.tb\_pedido.findMany({

include: { itens: true, usuario: true },

});

res.json(pedidos);

};

 **Rodar o servidor:**

Agora, rode o servidor usando ts-node-dev:

bash

Copiar código

npx ts-node-dev src/index.ts

Testando a API

Objetos produto

{ "id": 1, "descricao": "Um smartphone de última geração com câmera de alta resolução.", "nome": "Smartphone X", "url\_imagem": "https://example.com/imagens/smartphone-x.jpg", "preco": 2999.99 }

{ "id": 2, "descricao": "Um notebook ultrafino com bateria de longa duração e tela 4K.", "nome": "Notebook Ultra 2024", "url\_imagem": "https://example.com/imagens/notebook-ultra-2024.jpg", "preco": 8499.99 }

{ "id": 3, "descricao": "Uma TV 4K de 55 polegadas com tecnologia HDR e som Dolby Atmos.", "nome": "Smart TV 4K 55''", "url\_imagem": "https://example.com/imagens/tv-4k-55.jpg", "preco": 3599.90 }

{ "id": 4, "descricao": "Um par de fones de ouvido sem fio com cancelamento de ruído ativo.", "nome": "Fones de Ouvido Bluetooth", "url\_imagem": "https://example.com/imagens/fones-bluetooth.jpg", "preco": 499.99 }

Objetos usuários

{ "id": 1, "email": "joao.silva@example.com", "nome": "João Silva", "senha": "senhaSegura123", "telefone": "+55 11 98765-4321", "data\_aniversario": "1990-05-20T00:00:00.000Z" }

{ "id": 2, "email": "maria.oliveira@example.com", "nome": "Maria Oliveira", "senha": "minhaSenhaForte2024", "telefone": "+55 21 99876-5432", "data\_aniversario": "1985-11-15T00:00:00.000Z" }

Objetos pedido

{ "id\_usuario": 1, "status": "Pendente", "data": "2024-10-06T12:00:00.000Z", "itens": [ { "id\_produto": 1, "qtde": 2, "preco": 2999.99 }, { "id\_produto": 2, "qtde": 1, "preco": 8499.99 } ] }

{ "id\_usuario": 2, "status": "Concluído", "data": "2024-10-06T15:30:00.000Z", "itens": [ { "id\_produto": 3, "qtde": 1, "preco": 1599.90 }, { "id\_produto": 4, "qtde": 3, "preco": 499.99 } ] }