Criando uma aplicação para banco de dados.

Crie um diretorio no seu editor de códigos:

-Mkdir my-app

Esse comando no terminal, cria um diretorio com o nome da aplicação.

Inicialize o projeto com o Node.Js:

-yarn init -y

Aplicando esse comando no terminal da aplicação, vai criar um arquivo json, que possibilita acessar as configurações de lincesa e versao de recursos instalados na aplicação.

Configurando o arquivo Json para receber o script

-Instalação de recursos para a aplicação-

Instalamdo o typescript utilizando o yarn:

Typescript, nada mais é que uma linguagem de programação, fortemente tipada para javascript, adicionando vários recursos à linguagem.

-yarn add typescript -D

"D": Dependencias para desenvolvimento de aplicações, nao se aplica ao produto final.

Instalação do express:

Express é um framework de gerenciamento para requisiçoes e respostas HTTP e URLs, que possui um sistema de rotas, tratamento dentro da aplicação, integração de varios sistemas e templates, que facilitam criação de páginas web.

-yarn add express

-yarn add types@express -D

Types@, serve para que a biblioteca, tenha implementação do typescript.

Instaldando os tratamentos de erros com o express:

-yarn add express-async-errors

É uma biblioteca que facilita o tratamento de erros em uma aplicação Node.js com express, ajudando a lidar com lançamento de funções assincronas dentro de suas rotas e middlewares, melhorando e simplificando o tratamento de erros.

Adicionando o Cors para liberar o acesso para qualquer IP acessar:

-yarn add cors

O Cors (Cross-Origin Resouces Sharing), serve como um mecanismo de segurança implementado pelos navegadors web, controlando as requisições feitas por scripts de um dominio para outro dominio. É crucial ter ele em um projeto, para evitar problemas de segurança e requisições nao autorizadas de um domínio diferente.

-yarn add @types/cors

Faz com que o typescript seja configurado para a bilbioteca do cors.

Adicionando o mysql para a conexçao com o banco de dados:

O mysql é um sistema de gerenciamento de banco de dados relacionais, ele empenha o papel de organização, armazenamento e recuperação de informações.

-yarn install mysql

Adiciona o mysql na aplicação.

Instalando o modo de script do typescript para subir o servidor:

-yarn add ts-node-dev -D

Este commando reinicializa o processo de destino do Node.js, quando qualquer um dos aqeuivos necssarios é alterado, aynebtabdi a velocidade das reinicializações, o que faz que não tenha a necessidade de instanciar a compilação do node toda vez.

Para que ele rode corretamente, deve se criar o script no arquivo json com o seguinte comando:

Exemplo:

-Instalando o prisma e criando um base de dados com o migrate-

Instalando o prisma na aplicação:

O prisma é uma ferramenta que simplifica a criação e implementação com o banco de dados em aplicações web e mobile, oferecendo ferramentas para o acesso do banco de dados tornando-o mais facil e agil para a criação e gerenciamento, com um ORM(Object-Relational Mapping).

-yarn add prisma -D

Adiciona a biblioteca Prisma na aplicação, como uma dependencia de desenvolvimento.

-yarn add @prisma/client

Esse comando adiciona o client da bilbioteca prisma, que permite : criar, consultas, intergir e modificar o banco de dados.

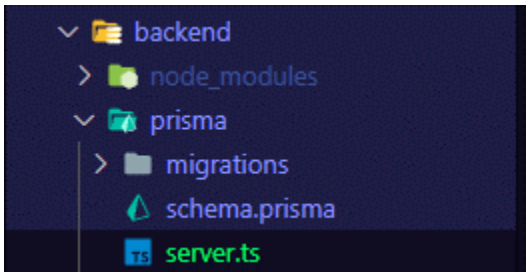
Apos essa parte, crie o banco de dados para a implmentação.

Exemplo:

```
generator client {
  provider = "prisma-client-js"
}

You, 2 weeks ago | 1 author (You)
datasource db {
  provider = "mysql"
  url      = "mysql://root@localhost:3306/apresentacao"
}

You, 2 weeks ago | 1 author (You)
model Usuarios {
  id             String    @id @default(uuid())
  nome           String
  email          String
  senha          String
  criado_em      DateTime? @default(now())
  atualizado_em DateTime? @default(now())

  @@map("usuarios")
}
```

Crie um server.ts dentro da pasta do prisma:

Exemplo:



Crie o codigo de servidor do prisma:

```
import { PrismaClient } from "@prisma/client";

const prismaClient = new PrismaClient()

export default prismaClient
```

-npx prisma init

Este comando, inicializa o projeto no diretorio atual, ele cria uma estrutura de diretorios, com arquivos padrão, como o 'schema.prisma', que é quem define o esquema de banco de dados, e o arquivo prisma com as informações de como conectar ao banco de dados.

Inicializando o typescript:

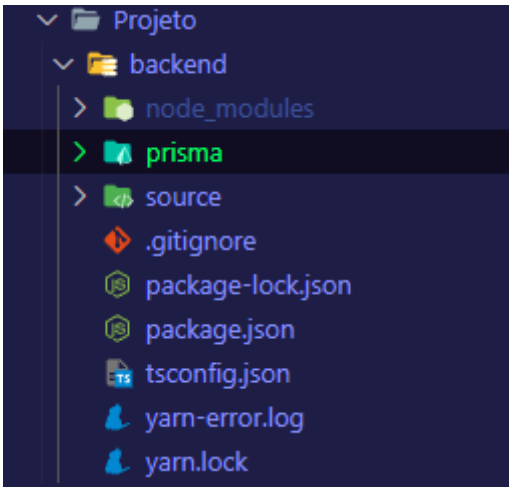
-yarn tsc –init

Funciona como um compilador web, para que ele funcione o typescript tem que estar instalado em todas as dependencias da aplicação.

-Codando o back end-

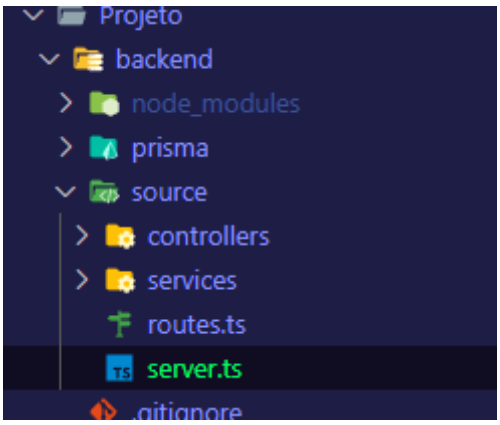
Em seguida, suba o servidor, com o xampp, para que o servidor mysql esteja rodando.

Agora, crie uma pasta chamada source dentro da aplicação:



E dentro dela um arquivo server.ts como no exemplo a seguir que vai servir como o servidor do express.

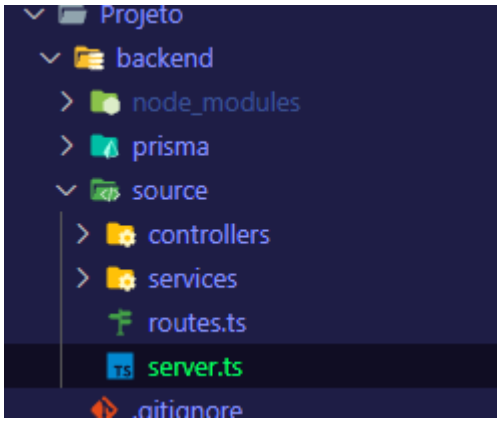
Exemplo 1:



Exemplo 2:

```
You, 2 weeks ago | 1 author (You)
import express, { Request, Response, NextFunction } from "express";
import "express-async-errors";
import cors from "cors";
import { router } from "./routes";

const app = express();
app.use(express.json());
app.use(cors());
app.use(router);

app.listen(3333, () => {
  console.log("Rodando na porta 3333");
});

Complexity is 4 Everything is cool!
app.use((err: Error, req: Request, res: Response, next: NextFunction) => {
  if (err instanceof Error) {
    return res.status(400).json({
      error: err.message,
    });
  }
  return res.status(500).json({
    status: "Erro",
    message: "Erro Interno",
  });
});
```

Crie um arquivo routes.ts, que vai servir como m arquivo de rotas.



Esse aqruivo vai definir a inserção de rotas, atualização, bucar e exluir os registros.

No terminal da aplicação, instale o axios:

O axios serve para facilitar as requisições HTTP, com GET, PUT, POST, DELETE, permitindo que a aplicação utilize API's externas, ajudando no tratamento de promessas, interceptação de Requisições e Respostas, gestçao de cookies e headers, alem do tratamento de erros.

-yarn add axios

Esse comando vai adicionar a biblioteca do axios

Criando o compnente de controladores e funcionalidades:

Controladores são os componetes fundamentais na aplicação, serve para atuar como o intermediario entre a interface (Front-End), e o usuario.

ALGUMAS FUNÇÔEs dos controladores:

-Gerenciamento de rotas.

-Integração com a visualização, conversando com a parte visual.

-Validar dados.

-Controlar o fluxo da aplicação.

-Reutilização dos codigos.

-Facilidade em testes na aplicação

Aqui um exemplo de controlador.

Este é um controlador de criação de usuario:

```
import { Request, Response } from "express"        You, 2 weeks ago • att
import { UsuarioServices } from "../../services/users/UsuariosSevices"

You, 2 weeks ago | 1 author (You)
class UsuarioController{

    async handle(req:Request , res:Response ){

        const { nome, email, senha } = req.body
        const usuarioServices = new UsuarioServices()
        const usuarios = await usuarioServices.excecute({
            nome,
            email,
            senha
        })

        return res.json(usuarios)

    }

}

export { UsuarioController }
```

Em resumo os controladores tem o papel de ajudar a manter a organização do codigo na aplicação, separação de responsabilidades e comunicação com o serviços.

Criando o componente de serviços e funcionalidades:

Serviços (Services), é um componente responsavel para lidar com alogica isolada da aplicação, aqui estao alguns exemplos de como ele funciona:

-Separação de responsabilidade, que ajuda na organização do codigo, deixando-o mais facil de entender.

-Facilidade em testes.

-Gerenciamento de estado.

-Integração de rescursos externos.

-Isolamento de dependencias.

Em resumo, o services serve para tratar os dados que entram na aplicação,os tratando e enviando/armazenando no banco de dados.

Um exemplo de services de criação de usuario: