Boa noite Professor, tudo bem?
Aluno: Bruno Cardoso da Silva

Vou comentar sobre o que eu realizei no TPE de desenvolvimento web:

Nosso trabalho do TPE foi baseado em um sistema de biblioteca, onde um usuário pode fazer cadastro de livros. Os usuários podem ter dois papéis, user e admin, onde o user pode apenas visualizar e o admin tem acesso a fazer todas as operações do crud.

Nosso trabalho foi feito em dupla e dividimos entre back-end e front-end e eu fiquei com a parte do backend. Optei por utilizar typescript por conta de já ter familiaridade com a linguagem e para facilitar nos autocompletes do código e intellisense.

Utilizei a estrutura padrão de pastas no backend, tudo dentro da pasta src -> Controller, Services, Middlewares, Types (para tipagem das requisições com POST), Database para armazenar o arquivo de conexão com o banco.

Eu sempre inicio pelo login, onde utilizei o express-session que, por incrivel que pareça, foi uma das minhas maiores dificuldades. Por nunca ter utilizado essa biblioteca, tive que pesquisar video aulas, documentação e utilizar a IA para entender como funcionava a sintaxe pra fazer as sessões funcionarem. Sempre optei por utilizar JWT, como no trabalho A3 foi utilizado, mas, no tpe, por ser algo mais simples, optei por usar o expres-session pra facilitar mas não facilitou nadakkk

Pra parte do login eu fiz um middleware que faz a verificação de autenticação de forma bem simples, apenas vendo se tem session e id do usuário dentro da sessão, caso sim, ele manda o next() e segue pra utilização da rota. No controlador de autenticação tem dois métodos, o de login e o de registrar um usuário novo:

O método de registrar primeiro valida se ja existe um usuário com as credenciais que foram recebidas, caso não, ele já cria dentro do banco de dados.

O método de login apenas compara as credenciais recebidas no corpo da requisição com o que já tem no banco de dados e se forem iguais ele deixa passar pra utilizar a aplicação.

Agora, vou dar uma explicada em cada arquivo seguindo a hierarquia das pastas, começando de cima pra baixo:

Controllers:

O arquivo do auth-controller.ts expliquei acima.

book-controller.ts tem todos os métodos relacionados aos livros, getAllBooks que pega todos os livros cadastrados, getBookById que pega um livro específico por ID, e os métodos de create, update e delete book que vão fazer, respectivamente, a criação, atualização e deletar um livro que está no banco de dados.

Database:

O arquivo prisma.ts faz a conexão do prisma, biblioteca utilizada como ORM para fazer as operações no banco de dados, com o banco de dados em si, então ele exporta uma

constante que se conecta com o banco de dados, permitindo o acesso a todas as tabelas do banco.

Middleware:

auth-middleware.ts já foi explicado acima também.

Routes:

Os arquivos de rotas são bem simples, tem um pra cada entidade, livros e usuários, e a classe que possui os métodos são instanciadas, então é definido qual a URL que vai ser acessada, qual middleware vai passar primeiro e qual controlador vai ser acionado em cada rota.

Services:

Os serviços tem a logica do negócio dentro de cada arquivo, então os métodos de fazer as buscas e envios de dados estão aqui, tanto para usuários quanto para os livros. Os métodos possuem nomes auto-explicativos. O prisma facilita bastante a busca de dados no bdd por conta dos métodos próprios, então somente utilizando o prisma o método já está praticamente pronto, somente nomear e chamar com o prisma o que você precisa que cada método retorne, não tem muito segredo.

Types e Utils

Na pasta types tem a tipagem de usuário e livro, então o que cada envio de dados pro banco precisa ter, isso facilita tanto na hora do intellisense do TS quanto para a validação dos dados que vem no corpo da requisição, então manter interfaces criadas em uma pasta ajuda bastante tanto na tipagem, intellisense quanto na validação de dados. Tem um arquivo também que faz a extensão da tipagem de requisição, que adiciona o usuáiro dentro da requisição para conseguir verificar o papel de aprovação dele.

Já na pasta utils, são arquivos úteis apenas que são utilizados para facilitar dentro do código. Tem um arquivo que simula um JWT, criando tokens aleatórios e verificando se ele existe mesmo ou não. E o outro arquivo é relacionado com a senha, pra hashar e comparar com o que tem no banco, dois mtétodos diferentes, um pra cada coisa.

Os dois últimos arquivos são o app e o server, o server é o servidor de fato onde definimos porta de conexão e tudo mais, o app é um arquivo que vai chamar métodos que são necessários para o funcionamento correto da aplicação, aí importamos o app no server e fazemos eles escutar a porta que desejamos que nossa API rode.

Acredito que o trabalho ficou simples, mas funcional e bem bacana na parte do backend, que foi o que eu fiz de fato, principalmente por conta do TS, ele tras uma camada de segurança e complexidade pro nosso código. A parte mais dificil foi com a autenticação do usuário, por que eu sempre utilizei JWT e o express-session tem algumas regrinhas chatas diferentes do JWT convencional. Utilizei a IA principalmente na parte do middleware de autneticação, por mais

simples que ele esteja, eu não sabia da onde recuperar os dados do usuáiro, por isso parti pra adicionar tudo na requisição cmoo se fosse JWT.

É isso, gostei do desenvolvimento do backend com ts e acho que ficou um trabalho honesto.