### Documentação Mentorin

*Codinome: Projeto Carcará*

*Por: Lucas Barbosa Brancalhão*

*Links:*

***Backend:***

*<https://docs.nestjs.com/> -- Documentação NestJS*

*<https://www.prisma.io/docs> -- Documentação Prisma*

*<https://www.passportjs.org/> -- Documentação Passport*

***Frontend:***

*<https://reactjs.org/docs/getting-started.html> -- Documentação React*

*<https://vitejs.dev/> -- Documentação Vite*

*<https://mui.com/material-ui/getting-started/overview/> -- Material UI*

#### *O que é o Mentorin?:*

O Mentorin consiste em um software de gestão e match entre um possível mentorado e um mentor, para que um possa auxiliar o outro.

O software possibilitará a Fomenta Vale automatizar o processo de combinar mentores e mentorados, que hoje é feito manualmente, além de dar uma visão geral de todos os mentores e mentorados cadastrados, o que pode ser estendido para outras empresas caso o projeto seja comercializado.

#### Tecnologias usadas:

***Backend:***

* *Typescript*
* *Nestjs*
* *MySql / Xampp*
* *Prisma*
* *Argon2*
* *Passport*
* *Nodemailer*
* *handlebars*

***Dependências Backend:***

* *Class-validator*
* *Reflect-metadata*
* *Uuidv4*
* *Nestjs/typeorm*
* *Passport-jwt*
* *Class-transformer*

***Frontend:***

* *React*
* *Vite*
* *MaterialUI*
* *Typescript*
* *Axios*
* *Zod*
* *Firebase*
* *RadixUI*

***Comentários sobre o Backend:***

O projeto foi produzido em cima do Framework NestJS, já que esse tem uma boa escalabilidade, além de fácil curva de aprendizado. Porém, ao que for trabalhar nesse projeto, é ideal que exita uma familiaridade com o superset Typescript, já que para que o projeto fosse mais seguro e de melhor manejo, foi utilizado a ferramenta.

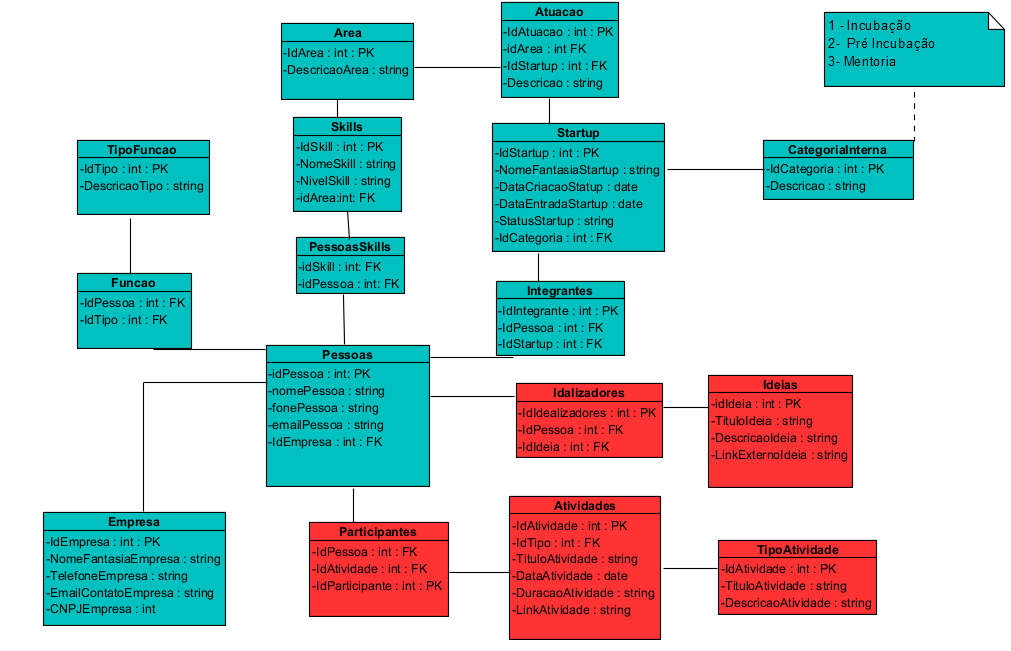
O banco de dados utilizado foi o MySQL, novamente por praticidade de uso, já que o mesmo oferece um ambiente pré-pronto para o fácil manejo e criação da base de dados.

No caso do ORM (Object-Relational-Mapping / Mapeamento de Objeto Relacional) foram feitas várias tentativas de diferentes ORMs, mas o que se viu mais apropriado foi o Prisma, já que o mesmo foi de fácil aprendizagem, boa implementação de relacionamento entre tabelas e boa escalabilidade.

Além disso foram usadas outras tecnologias que serão mencionadas ao longo do documento.

***Tabelas:***

Foi feito uma estrutura prévia das tabelas que seriam usadas no projeto. Segue abaixo o desenho feito.



Como pode ser visto acima, foram prototipadas 16 tabelas, das quais 11 foram coloridas como ***Azuis*** e 5 como ***Vermelhas***.

***Azul - Primeira Fase:***

Tabelas que foram codificadas inicialmente. - ***MVP (Minimum Viable Product)***

***Vermelho - Segunda Fase:***

Tabelas que serão codificadas futuramente.

***Relacionamento:***

**(Ternário)** - Pode ser vazio ou não (Operador “? - Ternário”).

**(1:1)** - Relação “Um para Um”.

**(1:m)** - Relação “Um para Muitos”.

**(m:m)** - Relação “Muitos para Muitos”.

***Primeira Fase:***

**Funcao - TipoFuncao** - 1:1;

**Area - Skills** - 1:m;

**Pessoas - Skills** - m:m; - Como é o caso de muitos para muitos, foi utilizado uma tabela auxiliar PessoasSkills, em que cada tabela tem a relação 1:m, respectivamente.

**Funcao - Pessoas** - 1:m;

**Pessoas - Integrantes** - 1:m;

**Empresa - Pessoas** - 1:m;

**Startup - Integrantes** - 1:m;

**Atuacao - Startup** - 1:m; - Foi utilizada a relação 1:m (Um para Muitos), mas talvez seria interessante usar uma relação m:m (Muitos para Muitos), já que uma Startup pode ter muitas atuações, e muitas atuações podem pertencer a muitas Startups.

**Area - Atuacao** - 1:m - Da mesma forma foi utilizada a relação 1:m (Um para Muitos), mas da mesma forma como o exemplo acima, talvez seja bom usar uma relação m:m (Muitos para Muitos), já que uma área pode pertencer a muitas atuações e muitas áreas podem pertencer a uma atuação.

**CategoriaInterna - Startup** - 1:m - Talvez seria válido fazer uma relação da tabela CategoriaInterna com outras tabelas, como por exemplo Pessoas, já que ela também tem a função de categorizar se alguém é mentor ou mentorado.

***Segunda Fase:***

As relações da segunda fase de implementação não foram especificadas, já que são de uma fase futura que será codificada, mas pode se imaginar as seguintes relações:

**Idealizadores - Ideias** - 1:m;

**Idealizadores - Pessoas** -1-1;

**Pessoas - Participantes** - 1:1;

**Participantes - Atividades** - 1:m;

**Atividades - TipoAtividade** - 1:m;

Todas as relações entre tabelas funcionam manualmente, porém, para melhor manipulação do sistema, algumas relações foram comentadas para a futura automação da relação, como foi feito no caso das tabelas Usuário e Pessoa.

**Descrição de Tabelas:**

**Pessoa** - A tabela de pessoas registra todos os dados de uma pessoa física.

**Funcao** - Relaciona uma pessoa e uma função atribuída pela empresa/startup onde a mesma trabalha.

**TipoFuncao** - Descreve o tipo de uma determinada função atribuída.

**PesoasSkils** - Tabela intermediária que faz a relação entre a tabela Skills e a tabela Pessoas.

**Skills** - Define uma habilidade relacionada a uma determinada área de atuação que uma pessoa pode possuir.

**Area** - Descreve a área em que uma habilidade/skill pode ser melhor encaixada. A tabela está relacionada com a atuação.

**Atuacao** - Define a atuação em que uma Area se encaixa. Também define a atuação de uma determinada Startup cadastrada.

**CategoriaInterna** - Descreve a situação de uma Startup cadastrada, seja Incubada ou não, ou no caso de uma Pessoa, Mentor ou não.

**Startup** - Descreve uma Startup incubada ou não.

**Integrantes** - Descreve os integrantes de uma Startup e as relaciona com pessoas, já que cada integrante precisa ser obrigatoriamente uma pessoa.

**Empresa** - Define uma empresa cadastrada a quem um mentor ou mentorado pertence.

**Participantes** - Define os participantes do “Match” entre mentores e mentorados. Também pode ser usada para saber quais os participantes de uma atividade(Reunião, Mentoria, etc.).

**Idealizadores** - Define os idealizadores de uma ideia, pessoas mentoradas.

**Ideias** - Define as idéias de pessoas idealizadoras querem implementar / ser mentoradas.

**Atividades** - Define uma atividade a ser realizada por participantes, como Reunião, Mentoria, Etc.

**TipoAtividade** - Define o tipo da Atividade sendo executada (Reunião, Mentoria, etc.)

Uma determinada alteração em uma tabela é feita através de uma migration no Prisma. Toda alteração em uma tabela precisa de uma migration para ser atualizada. Olhar documentação Prisma.

***Implementação:***

***Tabelas:***

Ao implementar as tabelas acima em Nest, foram seguidos os seguintes passos:

* Criar projeto NestJS.
* Gerar cruds com o gerador do framework.
* Implementar Prisma e a conexão com o Banco de dados.
* Definir regra de negócio no arquivo de serviços.
* Importar Prisma Service no módulo de exportação do crud.
* Importar modulo do crud no modulo principal (app.module).

***Autorização:***

A autorização consiste em uma permissão para que todo aquele que estiver logado possa realizar operações na API, mais especificamente, operações no crud de Usuários. Caso o cliente não esteja autorizado, ele não será capaz de realizar operações. Para a implementação, foram usadas as seguintes tecnologias:

***Passport / JWT:***

Consiste em um middleware de autenticação para NodeJS que pode ser usada em qualquer aplicação baseada em ExpressJS. Ele se integra com o JWT para gerar um login baseado em Token. O JWT gera o token a partir do JSON no banco de dados e o Passport confirma se o usuário informado bate com o usuário no banco de dados. Caso o usuário bata com o banco de dados, o Passport permite que o cliente acesse a ferramenta.

***Argon2:***

Consiste em um algoritmo de encriptação. No caso do sistema, ele foi usado para encriptar a senha do usuário para que o analista não veja a senha original no banco de dados.

***Guard:***

Basicamente um verdadeiro “Guarda” ou “Barreira”. É usado para proteger o crud desejado de acessos indevidos. Pode ser utilizado para proteger a rota de um crud inteiro ou apenas uma rota de função de um crud, como um create ou delete.

Foram seguidos os seguintes passos:

* Criado o crud de Auth.
* Criado o DTO para os dados necessários para a autenticação.
* Criado o arquivo de Guard.
* Criado o arquivo de jwt.strategy para validação do token.
* Inserido o Guard no Crud para ser protegido.
* Senha do usuário encriptada com o Argon2 e salvo no BD.

***Envio de Email:***

Foi iniciada a implementação de envio de email para recuperação de senhas usando handlebars e sendMail, porém a funcionalidade não foi finalizada. Foram seguidos os seguintes passos:

* Criado o crud de Mail com o gerador de cruds do framework.
* Criado o dto dos dados necessários
* Criado o serviço de envio de email usando nodemailer e sendMail
* Criado modelos handlebars.
* Testes feitos na plataforma SendMail e email recebido.

É necessário finalizar os modelos de email no handlebars e enviar para uma plataforma de email fechada, como o Gmail. Também é necessário finalizar o serviço de troca de senha por link.

***Plataformas Externas:***

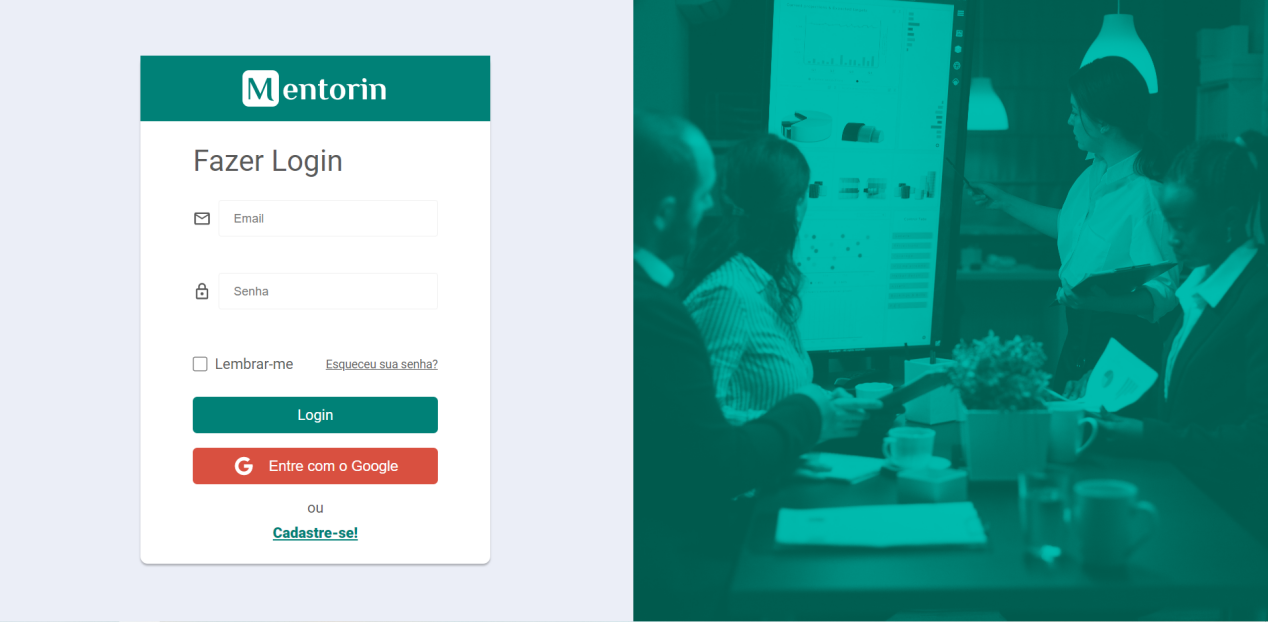
Para os testes de funcionalidade da API foram utilizadas algumas ferramentas externas:

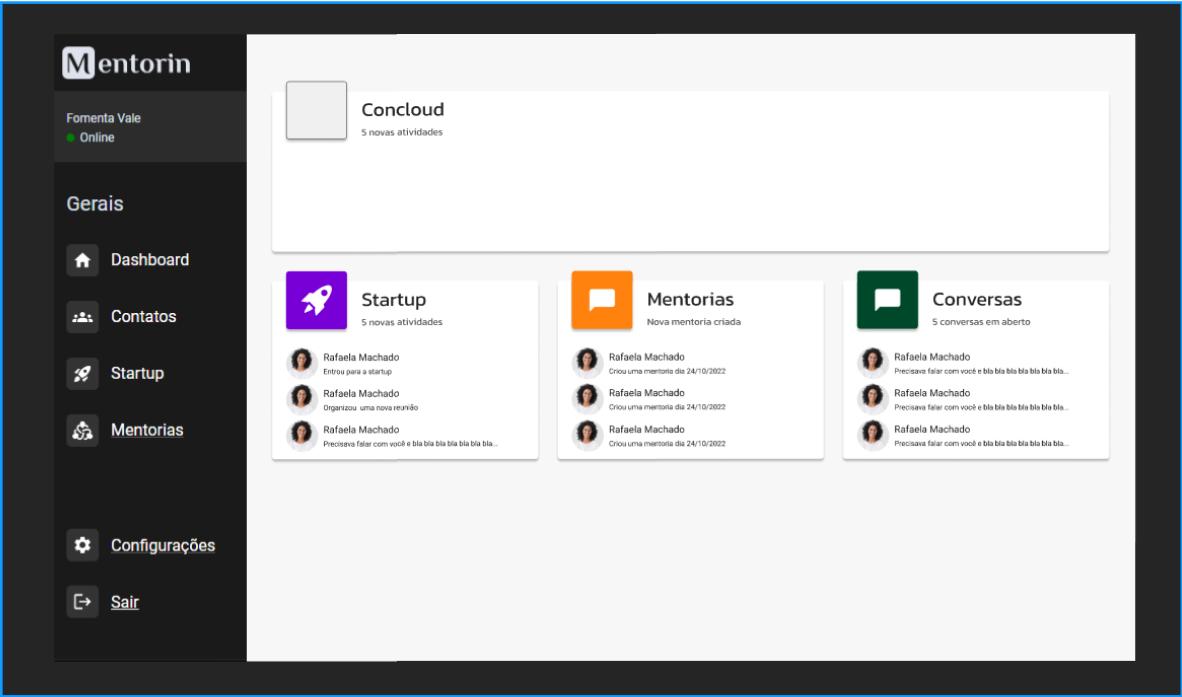
***Postman*** - Foi utilizado para testar cada rota da aplicação. Pode ser substituido pelo Insomnia.

***SendMail***- Para os testes de envio de email foi utilizada a ferramenta SendMail para interceptar os emails lançados. Caso seja necessário, crie uma conta nova e substitua os dados de autorização no módulo mail.

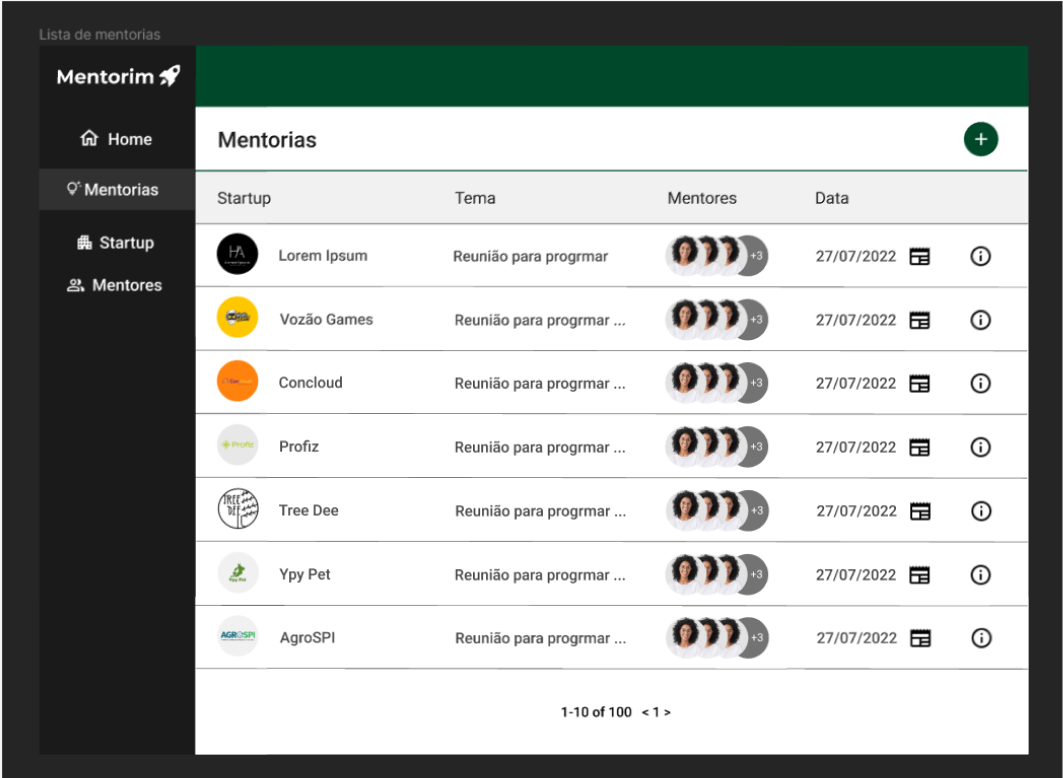
***Comentários sobre o Frontend:***

No caso do frontend, o projeto foi realizado em React, usando Vite para rapidez. Seguindo o sistema do React, foram produzidos componentes de cada elemento do front. Antes do front ser produzido, foram criados alguns protótipos que podem ser visualizados abaixo:









Não são todos os protótipos necessários, alguns ainda precisam ser feitos.

***Componentes:***

***cadastroMentoria*** *- Componente de modal para cadastro de nova mentoria.*

***infosMentoria*** *- Componente de informações de uma determinada mentoria.*

***Sidebar*** *- Componente de barra lateral (Esquerda) para navegação dentro da plataforma.*

***Mentorias*** *- Componente para listagem de mentorias cadastradas no backend.*

***Profile*** *- Componente da página para edição de perfil do usuário.*

***User*** *- Componente de contexto para confirmar se o usuário está logado e permitir acesso a plataforma.*

***Home*** *- Componente de geração da página home.*

***ForgetPasswort*** *- Componente de recuperação de senha.*

***SignIn*** *- Componente de SignIn (Login) na plataforma.*

***SignUp*** *- Componente de SignUp (Cadastro) na plataforma.*

***Login*** *- Componente que gera tela de Login para usuário.*

***Router*** *- Componente que controla as rotas da aplicação.*

***Main*** *- Componente raiz.*

***App***  *- Componente principal.*

***Serviços:***

Alguns arquivos do sistema não são componentes, mas também contribuem com funcionalidades. São eles:

***Axios****:*

Biblioteca Javascript que permite uma aplicação se comunicar com qualquer API disponível.

***Firebase****:*

*Aplicação da google que facilita e automatiza funcionalidades, como Login com uma conta google.*

***Comentários adicionais:***

Para a realização de testes em pequena escala, foi criado um pequeno projeto para que fosse usado na implementação. Após a implementação em pequena escala, a funcionalidade estaria pronta para ser implementada em grande escala. O nome dado ao projeto de pequeno porte foi “Projeto Carcarazinho” por motivos óbvios e comicos.

*<https://github.com/lucas51512/Carcarazinho> - Link Projeto Carcarazinho*