I-Mei

Integrantes : Cinthia Tengan, Lucas Corrêa, João Paulo Oliveira, Sthefany Silva e Sérgio Ruiz.

Escopo do Projeto

Com base nos Objetivos de Desenvolvimento Sustentável das Nações Unidas, escolhemos o item 8 - **Trabalho decente e crescimento econômico.** O modelo de desenvolvimento web escolhido foi uma rede social, cujo objetivo será divulgar produtos e serviços de pequenos produtores locais (consumíveis, artesanatos, mão de obra e etc).

Assim utilizando de trabalho voluntário para mapear os pequenos produtores, e também com profissionais qualificados que estejam dispostos a ajudar esses pequenos produtores com assessorias, suporte de melhoria para o negócio.

Através de nossa rede social, trabalhadores informais e microempreendedores poderão cadastrar os seus serviços e produtos na nossa plataforma, e assim divulgar para o público. Com o intuito de alavancar seu negócio e reerguer o pequeno empreendedor pós pandemia, bem como torná - lo mais competitivo.

Escolha do Nome

I - Mei - Significados

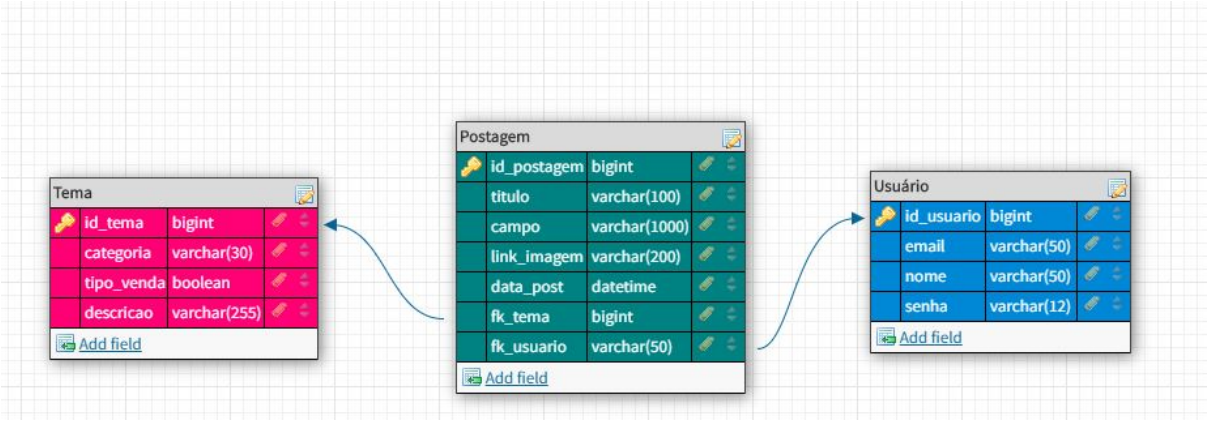
MEI - Micro Empreendedor Individual

I - eu em inglês

“Aimei” - Amei

Eu MEI

Descrição das tabelas e atributos do modelo DER



De acordo com as instruções propostas pelo instrutor, foram criadas 3 tabelas com seus respectivos atributos.

- **Tema**

id_tema: chave primária para poder identificar o tema escolhido; categoria: classificação do tema;

categoria: classificar o tema;

tipo_venda: de acordo com a regra de negócio, escolhemos que devem ter dois tipos produtos e serviços, e neste caso usamos um valor booleano no qual 0 representa produto e 1 o serviço;

descricao: descrever o tema selecionado com mais detalhe;

- **Usuário**

id_usuario: chave primária para identificar o usuário.

email: email do usuário.

Nome: nome do usuario completo;

senha: senha de acesso do usuário;

Nota: O atributo id_usuario foi criado posteriormente, pois ao utilizarmos o spring precisamos de uma chave primária com números, e não letras como havíamos escolhido o e-mail anteriormente.

- **Postagem**

  id_postagem: chave primária para identificar a postagem;

  fk_email: chave estrangeira da tabela usuario para poder identificar o usuário que fez a postagem;

  fk_tema: chave estrangeira da tabela tema para poder identificar o tema escolhido pelo o usuário na postagem;

  titulo: titulo da postagem;

  campo: texto principal da postagem do usuário;

  link_imagem: armazena a 'url' da imagem e 'linka' com a postagem;

  data_post: data da postagem feita do usuário.

Construção do Back End do projeto

Utilizamos o framework Spring para iniciar a construir a nossa API Rest, com as configurações de projeto maven, linguagem Java (versão 8) e foram adicionadas as dependências:

- **Spring Boot Dev Tools**
- **Spring Web**
- **Spring Data JPA**
- **Validation**
- **MySQL driver.**

Editado o arquivo Application Properties, para ligação com o banco de dados.

Criação do pacote Model por meio do padrão DAO com as classes:

- **Tema** - Atributos necessários para o tema da rede social
- **UserLogin** - Atributos para o login
- **Usuario** - Atributos para os dados do usuário
- **Postagem** - Atributos para a postagem.

O padrão **DAO** (Data Access Object) é um padrão de projeto que abstrai e encapsula os mecanismos de acesso a dados escondendo os detalhes da execução da origem dos dados.

Na model foi mapeado o relacionamento entre as tabelas do banco de dados:

OneToMany (um para muitos**)** ex: um tema pode ter N postagens.

ManyToOne (muitos para um) ex: N postagens para um usuário.

Criado o pacote repository para a persistência dos dados com as interfaces e seus métodos:

- **PostagemRepository** (findAllByTitulo)
- **TemaRepository** (findAllByCategoria, findByDescricao)
- **Usuario Repository**(findAllByNome, findByEmail)

Definindo por meio do repository as regras de negócio da nossa rede social, firmando os contratos com o JPARepository, estes métodos de busca serão utilizados na aplicação.

Criado o pacote controller com as classes:

- **PostagemController**
- **TemaController**
- **UsuarioController**

Nessas classes foram criados os métodos CRUD: Get, Post,Put e Delete com suas respectivas anotações e paths. Assim podemos enviar e receber requisições por meio do protocolo HTTP.

CRUD é o acrônimo da expressão do idioma Inglês, Create (Criação), Read (Consulta), Update (Atualização) e Delete (Destruição). Este acrônimo é comumente utilizado para definir as quatro operações básicas usadas em Banco de Dados Relacionais.

Criação da camada de Security

Para a camada de security foram adicionadas as dependências de *Spring security* e *Commons codec* ao arquivo de pom.xml do maven e construídas as classes:

- **Usuario** (modelo para classe usuário)
- **UserLogin** (modelo que entrega resposta quando o usuário logar)
- **UsuarioController** (controle para classe usuario)
- **UsuarioService** (classe contendo métodos para cadastro e login)
- **UsuarioRepository** (interface para a classe usuario)

Criado o pacote de segurança com as classes:

- **BasicSecurityConfig** (classe contendo as configurações de segurança)
- **UserDetailsImpl** (utiliza a regra de negócio da UserDetails do Spring)
- **UserDetailsServiceImpl** (classe de serviço que utiliza as regras de UserDetailsService do Spring)

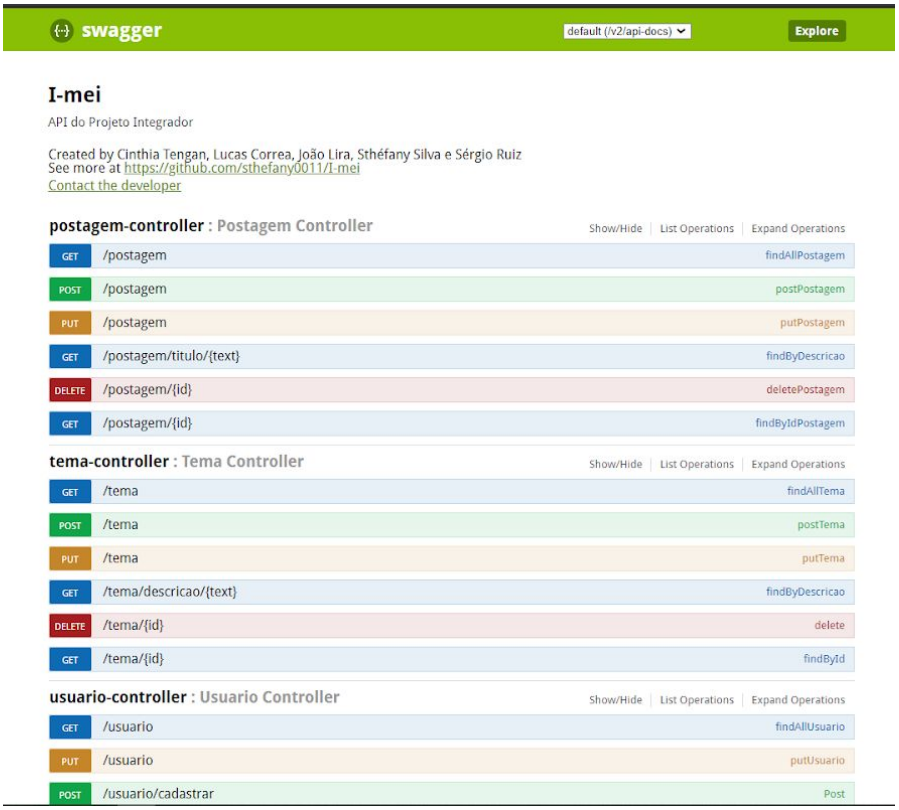
Criado o pacote service com a classe:

- **UsuarioService**

Nesta classe criamos a lógica para regra de negócio no cadastramento do usuário, encriptando a senha do usuário antes de salvá-la no banco de dados, bem como criando o token de acesso.

Swagger

Na implementação do Swagger, foram incluídas as dependências *Springfox Swagger2* e *Springfox Swagger ui*, e então criado o pacote config com uma classe chamada SwaggerConfig para possibilitar o acesso a documentação dos métodos existentes nos controllers do projeto.



Deploy:

Por meio do Maven como gerenciador de *BUILD*, fizemos o deploy do nosso projeto resultando no arquivo .jar

Construção da Marca I-mei

Começamos por um Brainstorming para o estudo das cores utilizadas no projeto:

Elementos sensitivo:

Comunidade
Desenvolvimento
Feira
Artesanal
Comunidade
Eu amei
Eu mei
Ajuda aos pequenos Empreendedores
Localizar os pequenos Produtores
Divulgação
Tecnologia acessível
Amor
Paixão
Oportunidades
União
Empresa
Empreendedorismo
Inovação
Compartilhamento
Ligação

Por meio dessas palavras construímos um Moodboard:

A partir desse moodboard surgiu nossa paleta de cores:

Gráficos:
Texturas lisas e enrugadas, cores frias, neutras, terrosas, cores de plantas, brilho e reflexos.

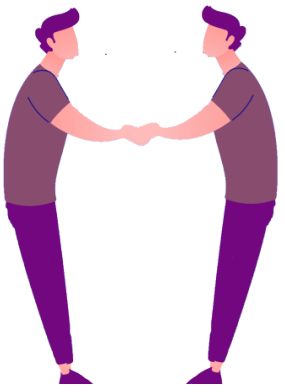
Paleta de cor secundária

Paleta de cores e seus respectivos códigos Hexadecimais:
Escolha do Logo:

Cada integrante fez sua versão do logo e por meio de votação escolhemos o seguinte logo e sua versão reduzida:

Logo branco aplicado em fundos de outras cores:

Este logo foi criado com o conceito de um coração e duas pessoas apertando as mãos fazendo negócios.



Construção do Front End do projeto

Por meio da plataforma de desenvolvimento Angular iniciamos nosso projeto, utilizando o editor Visual Studio Code e instalamos a última versão LTS do Node.js para podermos utilizar o pacote NPM

Criação dos componentes

Os componentes criados para o projeto foram instalados na pasta app em que ficarão todas as pastas de componentes como:

- cadastro (formulário para cadastramento do usuário)
- home (página inicial do site, sobre nós)
- inicio (página com as publicações postadas)
- login (página de login do usuário na plataforma)
- menu (menu superior presente nas páginas)
- rodape (rodapé inferior presente nas páginas)

Consumo de API

O componente Model foi criado para se relacionar com a model do Back end, desta forma esta pasta possui 4 arquivos que se relacionam com as 4 classes da model da API por meio dos mesmos atributos.

O componente service foi criado para conseguirmos consumir a API, esses services acessam os end points por meio de métodos. Qual endpoint? No Controller do back end fizemos os paths para serem acessados por meio de protocolo HTTP.

CRUD do Front end

Para os componentes tema e postagem foram construídos os CRUD de cada um, com os métodos do protocolo HTTP de post, get, put e delete para serem implementados diretamente no front.

- tema (para adicionar, editar e apagar um tema)
- postagem (para adicionar, editar e apagar uma postagem)

Por meio de Metainformação, dentro de suas pastas no arquivo typescript recebem a anotação @component e um selector app + nome do component, EX: <app-menu> desta forma podemos incluir estes seletores no código de qualquer outro component para trazê-lo para dentro do código.

Demais arquivos Angular que precisamos editar:

app.routing.module.ts - por meio deste arquivo e através do import de Routes e RouterModule, podemos fazer a navegação entre as páginas do nosso site, adicionando o caminho de cada component, EX: /login. Quando o site for carregado pela primeira vez, o routes irá direcionar o usuário para a página designada no arquivo.

app.component.html - por meio de selectores e do router trazemos o menu, rodapé e o component utilizado naquele momento.

index.html - mudamos a linguagem para Pt-br e adicionamos um site externo para termos ícones no site.

Style.css - Onde podemos colocar formatações css globais.

Após finalização de todos os itens e design das páginas finalizamos com o deploy da aplicação, e faremos o host dela no Heroku.

Integrantes/Informações para contato:

Cinthia Tengan	Lucas Corrêa	João P. Oliveira	Sthefany Silva	Sérgio Ruiz