Etapa 5 – Front

LINK DO REPOSITÓRIO NO GIT HUB - https://github.com/ViniOS/Chaves\_Web1

Este base deste arquivo foi criada com o comando ng g service services/ChaveService, no terminal com o local da pasta raiz do projeto.  
  
Texto

Descrição gerada automaticamente

*Imagem – chave-service.service.ts*

ChaveService é uma classe que é decorada com @Injectable(), o que significa que ela pode ser injetada como uma dependência em outras partes do aplicativo Angular.

private apiUrl = 'http://127.0.0.1:8000' define a URL base para a API com a qual este serviço interage.

O construtor da classe injeta uma instância do HttpClient do Angular, que é usado para fazer solicitações HTTP.

listarChavesDisponiveis() é um método que faz uma solicitação GET para a rota /chaves\_disp da API e retorna um Observable que emite uma lista de chaves disponíveis. Ele usa o operador map do RxJS para transformar a resposta em um formato mais útil.

enviarDados(dados: Chaves) é um método que faz uma solicitação POST para a rota /chaves da API, enviando os dados fornecidos como corpo da solicitação, e retorna um Observable que emite a resposta da API. Este método pode ser usado para enviar novos dados para a API.

A estrutura base deste componente foi criada automaticamente com o comando ng g c componentes/busca-chaves. (destacada em vermelho)

Tela de computador com fundo preto

Descrição gerada automaticamente

*Imagem - busca-chaves.component.html*

Texto

Descrição gerada automaticamente

*Imagem - busca-chaves.component.ts*

Este é um componente Angular chamado BuscaChavesComponent que usa o serviço ChaveService para buscar e exibir chaves disponíveis.

No template HTML do componente, há um loop \*ngFor que itera sobre a matriz chaves e exibe as propriedades id, nome, situacao e status de cada item.

Há também um botão personalizado app-botao que, quando clicado, aciona o método listarChavesDisponiveis().

No arquivo TypeScript do componente, o método listarChavesDisponiveis() chama o método listarChavesDisponiveis() do serviço ChaveService, que retorna um Observable. Quando a resposta é recebida, o método listarChavesDisponiveis() do componente itera sobre a resposta, cria um novo objeto Chaves para cada item e o adiciona à matriz chaves.

Se ocorrer um erro durante a requisição, ele será registrado no console.

Este componente é um exemplo de como buscar e exibir dados de um servidor em um aplicativo Angular.

A estrutura base deste componente foi criada automaticamente com o comando ng g c componentes/botao. (destacada em vermelho).

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

*Imagem - botao.component.html*

*Texto

Descrição gerada automaticamente*

*Imagem - botao.component.ts*

Este é um componente Angular chamado BotaoComponent que representa um botão personalizado.

No template HTML do componente, há um elemento de botão que, quando clicado, aciona o método click(). O texto do botão é vinculado à propriedade label do componente.

No arquivo TypeScript do componente, a propriedade label é decorada com @Input(), o que significa que o valor pode ser passado para o componente a partir de um componente pai.

Há também um EventEmitter chamado onClick que é decorado com @Output(), o que significa que ele pode emitir eventos para o componente pai.

O método click() emite um evento onClick quando é chamado.

Este componente é um exemplo de como criar um botão personalizado em um aplicativo Angular que pode aceitar entradas e emitir eventos para o componente pai.

A estrutura base deste componente foi criada automaticamente com o comando ng g class models/chaves. (destacada em vermelho).

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

*Imagem – chaves.ts*

Este é um modelo Angular chamado Chaves. Ele define a estrutura de um objeto Chaves que tem quatro propriedades: id, nome, situacao e status.

A propriedade id é um número que é inicializado como 0. As propriedades nome, situacao e status são strings que são inicializadas como strings vazias.

Este modelo pode ser usado para criar novos objetos Chaves em um aplicativo Angular. Por exemplo, quando você está buscando dados de um servidor, você pode mapear os dados recebidos para um array de objetos Chaves usando este modelo. Isso facilita o trabalho com os dados no seu aplicativo, pois você pode ter certeza de que todos os objetos têm a mesma estrutura.