



PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

Pontifícia Universidade Católica de Campinas

Sistemas de Informação

Escola Politécnica

PROJETO INTEGRADOR IV

Entrega 02 - Front end - 30/11/2023

Integrantes:

RA:

FELIPE ANDREAS SILVA

21004431

GABRIEL LEITE SILVA

21012871

MURILO ALMEIDA TREVISAN

21004881

RODRIGO MARQUES JUSTA

21002177

UEBERSON ANDRADE PEREIRA

21011648

Campinas

2023

Explicações sobre as etapas feitas no Front end , Boas Práticas e SOLID

VS CODE:

```
{ } extensions.json
```

Explicação : Representa uma configuração para recomendações de extensões. Nesse caso, a recomendação específica é para a extensão chamada "Angular Language Service" relacionada ao Angular

```
{ } launch.json
```

Explicação : Representa um arquivo de configuração para a execução e depuração de uma aplicação Angular no Visual Studio Code

```
{ } tasks.json
```

Explicação : Define duas tarefas que utilizam o npm para executar scripts do package.json: uma chamada "start" e outra chamada "test". Ambas são configuradas para rodar em segundo plano e lidar com problemas relacionados ao TypeScript durante a execução. Essas tarefas automatizam processos comuns no desenvolvimento.


```
{ } tsconfig.spec.json
```

Explicação : especifica o diretório de saída para os arquivos transpilados de teste, inclui o tipo "jasmine" para suporte ao framework de teste Jasmine, e define os padrões de inclusão para os arquivos de teste e declarações TypeScript no diretório "src" e seus subdiretórios.

SOLID : o código fornecido é uma configuração de compilação e não contém implementações específicas de código que possam ser avaliadas aos princípios do SOLID

Boas Práticas :

- . Inclui um comentário informativo no início do arquivo, indicando onde obter mais informações sobre o arquivo de configuração .
- . Usa formatação consistente e espaçamento adequado para facilitar a leitura do arquivo.
- . A configuração é explícita e específica, indicando claramente o diretor de saída e os tipos incluídos.

 tsconfig.json

Explicação : define opções de compilação, como o diretório de saída, força consistência na nomenclatura de arquivos, habilita configurações estritas para melhorar a robustez do código, utiliza o sistema de módulos ES 2022, ativa experimentos como decoradores e importa ajudantes. As opções do Angular Compiler abordam questões relacionadas à internacionalização, injeção de parâmetros rigorosos e acesso rigoroso a modificadores de entrada e templates.

SOLID : não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . Inclui um comentário informativo no início do arquivo, indicando onde obter mais informações sobre o arquivo de configuração.
- . Usa nomes de propriedades e valores claros e descritivos, como "outDir" e "strict" e evita abreviações obscuras.
- . Usa formatação consistente e espaçamento adequado para facilitar a leitura do arquivo.
- . Adere aos valores padrão do Angular quando apropriado, evitando configurações excessivamente personalizadas, a menos que necessário.
- . Evita duplicação de configurações, usando a extensão de configuração para reutilizar opções do arquivo base.

 tsconfig.app.json

Explicação : estende configurações de outro arquivo, define o diretório de saída para os arquivos transpilados, especifica o arquivo principal ("main.ts"), inclui declarações TypeScript e deixa as opções de tipo vazias.

SOLID : não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . Uso de comentários..
- . Uso de nomenclatura..

- . Evita duplicação.
- . Clareza na inclusão de arquivos.

```
{ } package.json
```

Explicação : Define scripts npm para iniciar, construir, observar e testar o projeto. As dependências incluem várias bibliotecas e frameworks Angular, Bootstrap, Font Awesome, e outras. As dependências de desenvolvimento abrangem ferramentas e bibliotecas para o desenvolvimento Angular, testes e compilação TypeScript

SOLID : Não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . Organização clara das dependências.
- . Scripts npm intuitivos.
- . Atualização das versões para as mais recentes disponíveis.

```
{ } package-lock.json
```

Explicação : Utilizado para garantir a consistência nas versões dos pacotes e suas dependências em um projeto. Ele armazena informações sobre versões específicas dos pacotes instalados, evitando discrepâncias entre ambientes de desenvolvimento e produção.

SOLID : não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . Fornecer uma estrutura organizada e informações detalhadas sobre as dependências
- . A especificação explícita de versões contribui para a estabilidade do projeto e facilita a manutenção

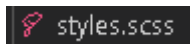
```
{ } angular.json
```

Explicação : Define a estrutura do projeto, especifica a linguagem de estilo para os componentes (SCSS), e configura o processo de construção (build) para gerar a aplicação, com opções específicas para os ambientes de produção e desenvolvimento. Além disso, há configurações para o servidor de desenvolvimento, extração de internacionalização (i18n) e testes usando o Karma.

SOLID : Não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . Os nomes de seções e propriedades são significativos, facilitando a compreensão.
- . A configuração é mantida relativamente simples, evitando complexidade desnecessária.
- . As configurações seguem um padrão consistente, o que ajuda os desenvolvedores a entender e manter o arquivo.
- . O arquivo é estruturado de forma clara, com seções bem definidas para diferentes partes da configuração.



Explicação : Inicia dinamicamente o módulo da aplicação usando a função `bootstrapModule`, após importar as dependências necessárias. O tratamento de erros está incluído para lidar com possíveis falhas durante o processo de inicialização.

SOLID : não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . Os nomes de seções e propriedades são significativos, facilitando a compreensão.
- . Possui uma estrutura concisa e precisa



Explicação : Define a estrutura básica de uma página web para uma aplicação Angular. Inclui elementos padrão, como configurações no `<head>` (como título e viewport) e a inclusão do componente principal (`<app-root>`) no `<body>`. O `<base href="/">` define o caminho base da aplicação.

SOLID : Não há evidências diretas nos dados fornecidos que indiquem conformidade ou não conformidade com os princípios SOLID

Boas Práticas :

- . O código é conciso e fácil de entender, seguindo o princípio de simplicidade do Clean Code.
- . Os nomes de elementos, como <app-root>, são descritivos e fornecem uma ideia clara do seu propósito.
- . O uso de configurações padrão, como o conjunto de caracteres UTF-8 e a escala inicial no viewport, segue a ideia de evitar configurações complexas e desnecessárias.

```
TS main.ts
```

Explicação : Inicia dinamicamente o módulo da aplicação. Importa a função `platformBrowserDynamic` e o módulo `AppModule`. Em seguida, utiliza `platformBrowserDynamic().bootstrapModule(AppModule)` para inicializar a aplicação Angular e imprime eventuais erros no console .

SOLID :

- . O código não possui evidências de nenhum princípio definido pelo SOLID

Boas Práticas :

- . Utilizar nomes de variáveis e funções descritivas e autoexplicativas.
- . Evitar redundância, manter funções curtas, e garantir que o código seja fácil de entender.

```
service
```

Explicação : Cada serviço é responsável por interagir com APIs RESTful, como autenticação, publicação de conteúdo e manipulação de listas de livros .

SOLID :

- . Princípio da Inversão de Dependências : seguem a injeção de dependência .

Boas Práticas :

- . Utilizar nomes de variáveis e funções descritivas e autoexplicativas.
- . Cada serviço parece ter uma única responsabilidade bem definida, promovendo a coesão.
- . O código é auto explicativo, evitando a necessidade de comentários óbvios
- . Os serviços são estruturados de uma forma que facilita a escrita de testes, como pode ser visto nos arquivos de teste.

PAGES:

select-livros

Explicação : Os códigos consistem em um componente Angular chamado SelectLivrosComponent, cuja função principal é exibir a mensagem "select-livros works!". Este componente é composto por um arquivo de código-fonte que define a classe do componente, juntamente com seu seletor, modelo e estilos associados e seu respectivo teste unitário.

SOLID :

. Princípio da Responsabilidade Única : O componente é responsável por exibir informações de resenhas, o que sugere uma responsabilidade única.

Boas Práticas :

- . O código segue a prática de ter uma classe/componente por arquivo
- . O nome do componente e do arquivo são descritivos.
- . O teste unitário segue uma estrutura padrão de inicialização do componente e valida se foi criado com sucesso.

resenhas

Explicação : Os códigos representam um componente Angular chamado ResenhasComponent e a estrutura associada ao seu modelo (resenhas.component.html) e estilos (resenhas.component.scss). Este componente faz parte de uma aplicação web e exibe informações relacionadas a resenhas de livros

SOLID :

. Princípio da Responsabilidade Única : O componente é responsável por exibir informações de resenhas, o que sugere uma responsabilidade única.

Boas Práticas :

- . O código HTML possui uma estrutura clara, com indentação adequada
- . O uso de classes CSS bem nomeadas e regras de estilo bem definidas contribuem para a manutenção do código.
- . A responsividade é considerada através do uso de media queries.

relendo

Explicação :

Os códigos fornecem um componente Angular chamado RelendoComponent que faz parte de uma aplicação web relacionada a livros sendo relidos. Este componente exibe

informações em uma estrutura organizada, incluindo um perfil com uma imagem, um menu de navegação e uma seção de notícias recentes sobre livros.

SOLID :

. Princípio da Responsabilidade Única : O componente é responsável por exibir informações de resenhas, o que sugere uma responsabilidade única.

Boas Práticas :

- . As classes e IDs em HTML, bem como as classes em CSS, são nomeadas de maneira descritiva, facilitando a compreensão do propósito de cada elemento.
- . O código HTML e CSS segue uma estrutura bem organizada e indentação consistente, facilitando a leitura e manutenção do código.
- . A responsividade é considerada com o uso de media queries para ajustar o layout em diferentes tamanhos de tela.
- . Há um teste unitário associado ao componente, seguindo a prática de garantir a qualidade do código por meio de testes automatizados

quero-ler

Explicação : Os códigos do QueroLerComponent em Angular são responsáveis por gerenciar a interação do usuário com a lista de livros que ele deseja ler . O componente apresenta uma função `listarLivrosQueroLer`, que interage com o serviço `QueroLerService` para obter a lista de livros desejados pelo usuário.

SOLID :

. Princípio da Inversão de Dependência :O componente utiliza injeção de dependência corretamente, dependendo do serviço `QueroLerService`. Isso facilita a substituição do serviço sem modificar o componente.

Boas Práticas :

- . As variáveis e funções têm nomes descritivos, contribuindo para a legibilidade do código.
- . O código está relativamente livre de comentários.
- . O componente lida com erros ao recuperar livros desejados, mas a lógica de tratamento de erro pode ser expandida para fornecer mensagens mais informativas ou ações alternativas.

lidos

Explicação : Os códigos fornecidos são parte de uma aplicação Angular, e sua funcionalidade principal é criar uma página de perfil com diversas seções. A estrutura da

página é organizada em uma div centralizada que contém informações de perfil, como uma foto, nome de usuário e um menu de navegação com links para diferentes seções, como "Estante", "Resenhas", "Recados", "Amigos", "Grupos", "Seguidores" e "Seguidos". Além de que a página inclui uma seção de notícias recentes sobre livros, onde cada notícia é representada por uma imagem, título e data .

SOLID :

- . Princípio da Responsabilidade Única: Há a presença de uma classe que tem apenas uma única responsabilidade

Boas Práticas :

- . O código HTML e CSS está relativamente bem estruturado, facilitando a leitura e os seletores da classe são seletivos.
- . O código se apresenta de forma modular e organizada.
- . Há a presença de comentários benéficos.
- . A inclusão de testes unitários é uma boa prática.

lendo

Explicação : Os códigos criam uma página com navegação e exibição de notícias recentes sobre livros. Além disso, há um componente relacionado à ação de listar livros sendo lidos

SOLID :

- . Princípio da Responsabilidade Única: O código segue o princípio SRP, pois cada componente ou classe parece ter uma responsabilidade específica. Por exemplo, o componente LendoComponent está focado na interação com os livros sendo lidos.

Boas Práticas :

- . As classes e IDs no HTML têm nomes descritivos, como "center," "meu_perfil," "foto," etc.
- . O código HTML e CSS está organizado em seções e classes, facilitando a leitura e manutenção e os estilos CSS estão organizados de forma clara, e a estrutura do TypeScript segue as convenções do Angular
- . Há a presença de comentários benéficos.
- . A inclusão de testes unitários é uma boa prática.

home

Explicação : Os códigos representam uma página principal de perfil em um aplicativo Angular. A página exibe informações do usuário, como nome e foto de perfil, e permite a

edição desses detalhes por meio de um modal. Um menu de navegação direciona para diferentes seções, como "Amigos," "Seguindo," "Minhas Publicações," "Grupos," "Editoras," e categorias de leitura como "Lidos," "Lendo," "Quero Ler," "Relendo," "Abandonei," e "Resenhas."

SOLID :

. Princípio da Responsabilidade Única: No código do componente HomeComponent, a classe é responsável por interações com o perfil do usuário, incluindo a exibição e edição.

Boas Práticas :

- . As variáveis e métodos geralmente têm nomes descritivos, como nomeEscritor e abrirModal. Isso contribui para a legibilidade do código.
- . O código está organizado em seções, como profile-info e profile-posts, facilitando a compreensão da estrutura da página
- . O uso de espaçamento e separadores, como <!--profile-info--> e <!--profile-posts-->, contribui para a clareza estrutural do código.
- . O código é relativamente fácil de entender, contribuindo para a manutenção .

efeito-colaborativo

Explicação : Os códigos representam uma página web na qual os usuários podem selecionar gêneros literários de sua preferência. Após selecionar os gêneros desejados e clicar no botão "Selecionar", os gêneros escolhidos são exibidos na página. O código também inclui estilos CSS para a aparência da página.

SOLID :

. Princípio da Responsabilidade Única: No código fornecido é possível observar a separação de responsabilidades entre a estrutura HTML (marcação), o estilo CSS e um componente Angular.

Boas Práticas :

- . Variáveis e elementos têm nomes descritivos, como "checkbox-group" e "selected-options", contribuindo para a compreensão do código.
- . O código está estruturado em seções (container, options) e usa classes CSS bem definidas para melhor organização.
- . Há ousa consistente de espaçamento e indentação melhora a legibilidade do código.

abou-you

Explicação: O código representa uma página de formulário no contexto de uma aplicação Angular. A página coleta informações sobre o usuário e suas preferências de leitura

SOLID :

. O código da página de formulário em Angular não apresenta diretamente princípios específicos do SOLID, pois está mais relacionado à estruturação e estilização de uma interface do usuário.

Boas Práticas :

- . O HTML está bem estruturado, usando tags semânticas para criar um formulário compreensível e o CSS está organizado e fornece estilos de forma clara.
- . As classes e IDs têm nomes significativos, facilitando a compreensão do propósito de cada elemento.

abandonei

Explicação: O código fornecido representa uma página de perfil em uma aplicação Angular. Ele possui uma estrutura geral organizada em uma seção central que contém informações do perfil do usuário, incluindo uma foto vinculada à página principal. Além disso, apresenta um menu de navegação com abas para diferentes seções, como "Estante", "Resenhas", "Recados", "Amigos", "Grupos", "Seguidores" e "Seguidos". A seção de notícias recentes exibe imagens de livros, títulos fictícios e datas de postagem simuladas.

SOLID :

. É possível identificar a aplicação do Princípio da Responsabilidade Única (SRP) em alguns trechos do código.

Boas Práticas :

- . As classes, métodos e variáveis têm nomes significativos, facilitando a compreensão do propósito de cada elemento.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . A maioria dos estilos está definida no arquivo CSS separado, evitando estilos inline diretamente no HTML.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . Existem regras de mídia no CSS para lidar com a responsividade, ajustando o layout para diferentes tamanhos de tela.

. Existe um teste unitário associado ao componente Angular, mostrando uma prática de desenvolvimento orientado a testes.

COMPONENTS:

modal-publicacao

Explicação: Os códigos fornecidos incluem um componente Angular chamado ModalPublicacaoComponent, responsável por exibir uma modal de publicação de informações de livros. O componente possui um formulário com campos para o título e descrição do livro, utilizando a funcionalidade ngModel para vinculação bidirecional. A modal é estilizada com Bootstrap e contém botões para fechar a modal e enviar o formulário.

SOLID :

. Princípio da Responsabilidade Única: O componente tem a responsabilidade de gerenciar a modal e interagir com o formulário

Boas Práticas :

- . Apresenta nomes de variáveis e métodos significativos
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . A maioria dos estilos está definida no arquivo CSS separado, evitando estilos inline diretamente no HTML.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . Existem regras de mídia no CSS para lidar com a responsividade, ajustando o layout para diferentes tamanhos de tela.

modal

Explicação: Os códigos apresentam um componente Angular chamado ModalComponent que representa uma modal para editar o perfil do usuário. O modal possui campos para editar o nome e selecionar uma nova foto, além de botões para salvar as alterações e fechar a modal. A funcionalidade principal é permitir ao usuário editar seu perfil.

SOLID :

. Princípio da Responsabilidade Única: O componente adere a esse princípio, pois se concentra na exibição da modal de edição de perfil e na interação com o formulário.

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.

- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . Existem regras de mídia no CSS para lidar com a responsividade, ajustando o layout para diferentes tamanhos de tela.

login

Explicação: Os códigos representam um componente Angular chamado LoginComponent, responsável por exibir um formulário de login. Os usuários podem inserir seu e-mail e senha para realizar o login. O componente também inclui um link para a página de cadastro caso o usuário não tenha uma conta. O login é efetuado através de um serviço (LoginService), que parece realizar requisições HTTP para autenticar as credenciais do usuário.

SOLID :

- . Princípio da Responsabilidade Única: O componente LoginComponent tem a responsabilidade única de gerenciar a interface de usuário relacionada ao login.

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . Uso de serviços para separar a lógica de negócios (lógica de login) do componente

header

Explicação: Os códigos representam um componente Angular chamado HeaderComponent, que define o cabeçalho de uma aplicação. O cabeçalho contém um logotipo do Skoob e uma barra de pesquisa com um campo de entrada e um botão para pesquisar títulos, editoras ou ISBN. O cabeçalho é estilizado com borda inferior verde e espaçamento interno.

SOLID :

. Princípio da Responsabilidade Única: O HeaderComponent tem uma única responsabilidade: gerenciar a apresentação do cabeçalho da aplicação.

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . O componente é testável, como evidenciado pela presença de testes unitários.

footer

Explicação: Os códigos representam um componente Angular chamado FooterComponent, que define o rodapé (footer) de uma aplicação. O rodapé inclui informações sobre a marca Skoob, links para redes sociais, links para download do aplicativo, e links de navegação para seções específicas do site.

SOLID :

. Princípio da Responsabilidade Única: Apresenta uma única responsabilidade que é gerenciar a apresentação do rodapé da aplicação.

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . O componente é testável, como evidenciado pela presença de testes unitários.

curtir

Explicação: Os códigos representam um componente Angular chamado CurtirComponent que consiste em um botão "Curtir" dentro de uma seção identificada pela classe "like-section".

SOLID :

. Princípio da Responsabilidade Única: Possui uma única responsabilidade que é renderizar um botão de "Curtir".

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . O componente é testável, como evidenciado pela presença de testes unitários.

compartilhar

Explicação: Os códigos representam um componente Angular chamado CompartilharComponent. Esse componente consiste em um botão "Compartilhar" dentro de uma seção identificada pela classe "share-section".

SOLID :

. Princípio da Responsabilidade Única: Possui uma única responsabilidade que é renderizar um botão de "Compartilhar".

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . O componente é testável, como evidenciado pela presença de testes unitários.

comentario

Explicação: Os códigos do componente de comentário são parte de uma aplicação Angular, gerenciando a exibição e interação de comentários em posts. Ele oferece a funcionalidade de adicionar, remover e curtir comentários, com a capacidade de alternar a visibilidade dos comentários. Esse componente permite uma interação dinâmica e persistente dos usuários em relação aos comentários em posts.

SOLID :

. Princípio da Responsabilidade Única: Possui uma única responsabilidade que é gerenciar a seção de comentários, lidando com a exibição, adição, remoção deles

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.
- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . O componente é testável, como evidenciado pela presença de testes unitários.
- . Há consideração para responsividade usando media queries, adaptando o layout para tamanhos de tela menores.

cadastro

Explicação: Os códigos representam a implementação de um formulário de cadastro em uma aplicação Angular. O componente é responsável por exibir o formulário e coletar informações do usuário, como nome, e-mail e senha. O formulário inclui campos para o nome, e-mail e senha do usuário, e um botão de cadastro. Ao preencher o formulário e clicar no botão "Cadastrar", os dados são enviados para um serviço de cadastro por meio de uma solicitação HTTP. O serviço processa a solicitação, cadastrando o usuário e tratando os casos de sucesso ou erro.

SOLID :

. Princípio da Responsabilidade Única: Concentra-se em lidar com a lógica de apresentação e interação com o usuário para o formulário de cadastro. Além disso, a responsabilidade de interação com o serviço de cadastro é delegada a CadastroService

Boas Práticas :

- . Nomes de variáveis e métodos são significativos, facilitando a compreensão do código.
- . O código segue uma boa formatação, com indentação consistente e espaçamento apropriado, tornando-o mais legível.

- . Há uma clara separação de preocupações entre HTML, CSS e TypeScript, seguindo o princípio de "Separation of Concerns".
- . O código parece seguir uma abordagem de modularização, mantendo responsabilidades específicas dentro do componente.
- . Comentários são utilizados com parcimônia e apenas quando necessário para explicar decisões não triviais.
- . O componente é testável, como evidenciado pela presença de testes unitários.
- . Há consideração para responsividade usando media queries, adaptando o layout para tamanhos de tela menores.

