Centro Universitário de Excelência

Sistema de Informação

Gabriel Frederico Martins - 218632021

Portfólio
Angular

Guarulhos
2025

GABRIEL FREDERICO MARTINS

PORTFÓLIO
ANGULAR

Trabalho apresentado ao Curso Sistema de Informação do Centro Universitário ENIAC para a disciplina Framework Angular.

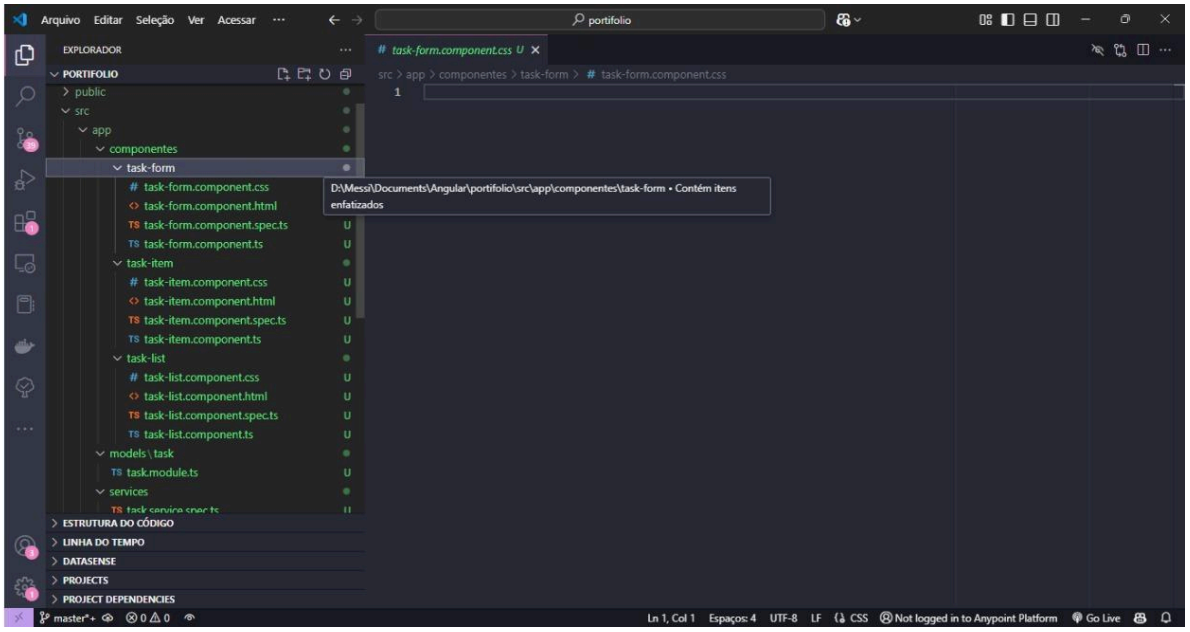
Prof.Nelson Luzetti

Guarulhos

2025

ENIAC
Centro Universitário de Excelência

Como você planeja
organizar a estrutura do projeto Angular para alcançar esse objetivo? Descreva os
principais diretórios, arquivos e componentes que você criará, e como eles se relacionam para criar o aplicativo de gerenciamento de tarefas.

- 

Diretórios e Arquivos

Diretório `app/`:

Esse diretório contém todos os arquivos principais do aplicativo, como componentes, serviços, modelos, módulos e a configuração da aplicação.

`components/`: Este diretório conterá os componentes principais da interface do usuário (UI). Aqui, eu criaria:

`task-list/`: Responsável por exibir a lista de tarefas.

tasklist.component.ts: Controlador que gerencia a lógica para exibir as tarefas.

tasklist.component.html: Template da lista de tarefas.

tasklist.component.css: Estilos específicos para o componente de lista de tarefas.

`task-item/`: Componente para representar uma tarefa individual na lista.

task-item.component.ts: Controlador do item de tarefa (editar, excluir, concluir).

task-item.component.html: Template para cada tarefa.

task-item.component.css: Estilos específicos para o item de tarefa.

`task-form/`: Componente para o formulário de criação e edição de tarefas.

task-form.components: Lógica de formulário (criar e editar tarefas).

task-form.component.html: Template do formulário de criação/edição.

task-form.component.css: Estilos específicos para o formulário.

`services/`:

task.service.ts: Serviço que gerencia a comunicação com o backend (API) para realizar as operações CRUD (criar, ler, atualizar, excluir) sobre as tarefas.

`models/`:

task.model.ts: Modelo que define a estrutura de uma tarefa (ex.: id, descrição, status).

`app.component.ts`: Componente raiz do aplicativo que faz a ligação entre os outros componentes.

`app.component.html`: Template do componente raiz.

`app.module.ts`: Módulo principal do Angular que importa outros módulos e componentes essenciais para o funcionamento do aplicativo.

`app-routing.module.ts`: Arquivo responsável pela configuração de rotas da aplicação, facilitando a navegação entre páginas ou seções do app.

Diretório `assets/`:

Este diretório armazena arquivos estáticos, como imagens e estilos globais que são utilizados em toda a aplicação.

Diretório `environments/`:

Contém arquivos de configuração para diferentes ambientes (desenvolvimento, produção, etc.), permitindo a separação de variáveis de ambiente.

`index.html`:

Arquivo de entrada da aplicação, onde o Angular irá inserir o conteúdo do aplicativo.

Relacionamento entre Componentes e Serviços

A arquitetura do Angular utiliza uma abordagem baseada em componentes, serviços e módulos. Neste caso, os componentes e serviços têm as seguintes interações:

O **componente `task-list`** interage com o **serviço `task.service.ts`** para buscar as tarefas e exibi-las na interface do usuário.

O **componente `task-item`** permite que o usuário edite, exclua ou marque uma tarefa como concluída. Ele também se comunica com o `task.service.ts` para realizar essas operações de CRUD.

O **componente `task-form`** permite a criação e edição de tarefas. Ao

submeter o formulário, ele aciona o **serviço `task.service.ts`** para salvar ou atualizar a tarefa no backend.

Conclusão Pessoal

Durante o desenvolvimento deste aplicativo de gerenciamento de tarefas utilizando o Angular, percebi que a organização da estrutura de arquivos e a divisão clara de responsabilidades entre componentes e serviços é fundamental para garantir escalabilidade e facilidade de manutenção. Utilizar a arquitetura baseada em componentes permite uma fácil reutilização e modificação do código, o que facilita a implementação de novos recursos no futuro.

Os principais desafios enfrentados foram relacionados à gestão do estado da aplicação e à comunicação eficiente entre os componentes e o backend. Para resolver isso, implementei o uso do serviço `task.service.ts` como o ponto central de comunicação entre a interface do usuário e o servidor, além de adotar práticas de gerenciamento de estado, como o uso de Observables e Subject para garantir que as mudanças no estado da aplicação se refletissem corretamente na interface.

A experiência de trabalhar com o Angular também destacou a importância de garantir que a interface fosse responsiva e amigável ao usuário, o que exigiu o uso de boas práticas de CSS e design de componentes interativos. No geral, este projeto foi uma excelente oportunidade de aplicar conceitos fundamentais do Angular e aprimorar habilidades no desenvolvimento de aplicações web escaláveis e dinâmicas.