Página Principal (Home): `public/home.html`

Esta é a página central da aplicação Taskea, acessível após o login bem-sucedido. Ela exibe a lista de tarefas do usuário, permite a criação de novas tarefas (através de um link) e oferece funcionalidades básicas de gerenciamento, como a exclusão de tarefas.

Estrutura HTML e Estilização:

O `home.html` organiza o conteúdo principal da aplicação logada.

- **Cabeçalho (`<head>`)**: Similar às páginas de login/cadastro, inclui metadados, fontes (`Inter`), ícones (`Material Icons` , `Material Symbols`) e o link para a folha de estilo específica: `<link rel="stylesheet" href="styles/home/style.css">` . Notavelmente, há também um bloco `<style>` embutido que define estilos específicos para os botões de exclusão e cancelamento (`.botao-excluir` , `.botao-cancelar`) e para a barra de ações de exclusão (`#acoes-excluir`).
- **Corpo (`<body>`)**:
  - **Header Fixo (`<div class="container_header">`)**: Contém a barra superior da aplicação.
    - **Link "Nova tarefa"**: Um link (`<a>`) que direciona o usuário para `new_task.html` .
    - **Logo**: Exibe o logo da Taskea.
    - **Menu da Conta (`<div class="conta">`)**: Um ícone de conta (`account_circle`) que, ao ser clicado, revela um dropdown (`<div class="dropdown">`). O dropdown exibe uma saudação com o nome do usuário (preenchido via JavaScript) e um link "Sair" que leva de volta à página de login (`login.html`) e efetua o logout (via JavaScript).
  - **Container de Tarefas (`<div class="container_options_tasks">`)**: Agrupa a seção principal de visualização e gerenciamento de tarefas.
    - **Título e Opções**: Exibe o título "Suas tarefas" e um ícone de menu (`menu`) que, ao ser clicado, revela um dropdown (`<div class="dropdown-menu">`) com a opção "Excluir".
    - **Mensagem de Lista Vazia (`<p id="mensagem-vazia">`)**: Parágrafo exibido apenas quando não há tarefas para mostrar.
    - **Lista de Tarefas (`<div id="lista-tarefas">`)**: Container vazio que será preenchido dinamicamente com os cards de tarefas pelo JavaScript.

Funcionalidade JavaScript:

O script no final da página orquestra toda a interatividade da página principal.

1. **Verificação de Token**: Logo no início, o script verifica se existe um token JWT no `localStorage`. Se não houver (`!token`), o usuário é imediatamente redirecionado para a página de login (`window.location.href=\'/login.html\'`), protegendo o acesso à página.
2. **Gerenciamento de Menus Dropdown**: Implementa a lógica para exibir/ocultar os dropdowns da conta do usuário e do menu de opções de tarefas ao clicar nos ícones correspondentes. Também fecha os dropdowns se o usuário clicar fora deles.
3. **Carregamento de Tarefas (`carregarTarefas`)**:
    - Função assíncrona que envia uma requisição `GET` para o endpoint `/tarefas` do backend.
    - Inclui o token JWT no cabeçalho `Authorization: Bearer ${token}` para autenticar a requisição.
    - Se a requisição for bem-sucedida (`resp.ok`), armazena o array de tarefas retornado pela API na variável `tarefas`.
    - Chama a função `renderizarTarefas()` para exibir as tarefas na tela.
    - Captura e exibe erros de comunicação com a API.
    - É chamada automaticamente quando a página carrega.
4. **Renderização de Tarefas (`renderizarTarefas`)**:
    - Limpa o conteúdo atual do container `#lista-tarefas`.
    - Verifica se o array `tarefas` está vazio. Se sim, exibe a mensagem `#mensagem-vazia` e retorna.
    - Se houver tarefas, esconde a mensagem `#mensagem-vazia`.
    - Itera sobre o array `tarefas` usando `forEach`.
    - Para cada tarefa, cria dinamicamente um elemento `div` com a classe `task-card`.
    - **Modo Exclusão**: Se a variável `modoExclusao` for `true`, cria e adiciona um checkbox (`<input type=\'checkbox\'>`) no início do card. Um listener é adicionado ao checkbox para adicionar/remover o índice da tarefa do array `indicesSelecionados` quando o estado do checkbox muda.
    - Cria um `div` para as informações da tarefa (título e descrição) e o adiciona ao card.
    - Adiciona o card completo ao container `#lista-tarefas`.
5. **Ativação do Modo Exclusão**: O listener no botão "Excluir" do menu de opções de tarefas inverte o valor da variável booleana `modoExclusao`, chama `renderizarTarefas()` para redesenhar a lista (agora com ou sem checkboxes) e, se o modo exclusão foi ativado, chama `criarAcoesExcluir()`.

6. **Criação da Barra de Ações de Exclusão (`criarAcoesExcluir`)**:
    - Remove qualquer barra de ações existente.
    - Cria um novo `div` com `id="acoes-excluir"` .
    - Cria dois botões: "Excluir selecionadas" (com classe `.botao-excluir`) e "Cancelar" (com classe `.botao-cancelar`).
    - Adiciona listeners aos botões: o botão "Excluir" chama `excluirSelecionadas()` , e o botão "Cancelar" desativa o `modoExclusao` , renderiza as tarefas novamente (sem checkboxes) e remove a barra de ações.
    - Insere a barra de ações no DOM, antes do container da lista de tarefas.
7. **Exclusão de Tarefas Selecionadas (`excluirSelecionadas`)**:
    - Verifica se há tarefas selecionadas (`indicesSelecionados.length === 0`).
    - Itera sobre os `indicesSelecionados` .
    - Para cada índice, obtém o ID da tarefa correspondente (`tarefas[idx].id`).
    - Envia uma requisição `DELETE` assíncrona para `/tarefas/${tarefa.id}` , incluindo o token no cabeçalho `Authorization` .
    - Observação: As requisições são enviadas em sequência (*for...of* com *await*). Para melhor performance com muitas exclusões, *Promise.all* poderia ser usado para enviá-las em paralelo.
    - Após todas as exclusões (ou falhas), exibe um alerta de sucesso, desativa o `modoExclusao` , chama `carregarTarefas()` para atualizar a lista na tela e remove a barra de ações.
    - Captura e exibe erros durante o processo de exclusão.
8. **Exibição do Nome do Usuário**: Recupera o nome do usuário do `localStorage` e o insere no elemento `#username` dentro do dropdown da conta.
9. **Logout**: O listener no link "Sair" do dropdown da conta remove o `token` e o `usuarioNome` do `localStorage` . A navegação para `/login.html` (definida no `href` do link) completa o processo de logout.

Fluxo do Usuário:

Ao acessar `home.html` , o usuário vê suas tarefas carregadas da API. Ele pode clicar em "Nova tarefa" para ir à página de criação. Clicando no ícone da conta, ele pode ver seu nome e clicar em "Sair" para fazer logout. Clicando no ícone de menu ao lado de "Suas tarefas", ele pode selecionar "Excluir". Isso ativa o modo de exclusão, exibindo checkboxes ao lado das tarefas e uma barra de ações na parte inferior. O usuário seleciona as tarefas que deseja excluir, clica em "Excluir selecionadas", confirma (implicitamente, pois não há diálogo de confirmação), as tarefas são excluídas no backend e a lista é atualizada. Ele pode também clicar em "Cancelar" para sair do modo de exclusão.

Página de Nova Tarefa (`public/new_task.html`)

Esta página fornece a interface para que o usuário adicione novas tarefas à sua lista. Ela contém campos para o título e a descrição da tarefa e botões para salvar a nova tarefa ou voltar para a página principal.

Estrutura HTML e Estilização:

O `new_task.html` foca na simplicidade para a entrada de dados.

- **Cabeçalho (`<head>`)**: Inclui metadados básicos (`charset` , `viewport`), o título (`<title>Nova tarefa</title>`), o link para a fonte `Inter` e o link para a folha de estilo específica: `<link rel="stylesheet" href="styles/new_task/style.css">` .
- **Corpo (`<body>`)**: Contém um `<main class="container">` .
    - **Logo**: Exibe o logo da Taskea.
    - **Caixa Central (`<div class="center-box">`)**: Agrupa os elementos de entrada e os botões.
        - **Campos de Entrada (`<div class="write">`)**:
            - Um `<input type="text" id="text" placeholder="Nova tarefa">` para o título.
            - Um `<textarea id="desc" placeholder="Adicionar uma descrição">` para a descrição.
            - Ambos possuem o atributo `required` , embora a validação explícita no JavaScript verifique apenas o título.
        - **Botões de Ação (`<div class="options">`)**:
            - Um botão "Voltar" (`<div id="voltar" class="buttons">`) dentro de um link (`<a>`) que direciona de volta para `home.html` .
            - Um botão "Salvar" (`<div id="salvar" class="buttons">`) que aciona a lógica JavaScript para criar a tarefa.

Funcionalidade JavaScript:

Um script no final da página gerencia a criação da nova tarefa.

1. **Listener no Botão Salvar**: Um event listener é adicionado ao `click` do botão com `id="salvar"` .
2. **Coleta de Dados**: Obtém os valores do título (`#text`) e da descrição (`#desc`), removendo espaços em branco com `trim()` .
3. **Validação de Título**: Verifica se o campo `titulo` está vazio. Se estiver, exibe um alerta (`alert()`) e interrompe a execução (`return`). A descrição não é validada como obrigatória no script, apesar do `required` no HTML.

4. **Verificação de Token**: Recupera o token JWT do `localStorage`. Se não existir, exibe um alerta informando que o usuário não está logado e o redireciona para `login.html`.
5. **Requisição `fetch`**: Se o título estiver preenchido e o token existir, envia uma requisição assíncrona `POST` para o endpoint `/tarefas` do backend.
    - `method: 'POST'`
    - `headers`: Inclui `Content-Type: application/json` e o cabeçalho `Authorization: Bearer ${token}`.
    - `body: JSON.stringify({ titulo, descricao })`: Envia o título e a descrição da nova tarefa. Observação: Conforme apontado na documentação do backend, o endpoint *`/tarefas`* no *`tarefaController`* atual não utiliza o *`usuario_id`* do token para associar a tarefa, o que é uma falha. O backend deveria extrair o *`usuario_id`* do *`req.usuario`* injetado pelo middleware.
6. **Tratamento da Resposta**:
    - **Sucesso (`resp.ok`)**: Se o backend retornar sucesso (status 201), exibe um alerta "Tarefa criada!" e redireciona o usuário de volta para a página principal (`window.location.href = '/home.html'`), onde a nova tarefa deverá aparecer na lista.
    - **Erro (`else`)**: Se o backend retornar erro, lê a mensagem de erro do JSON e a exibe em um alerta.
7. **Tratamento de Falha na Rede**: O bloco `try...catch` captura erros de comunicação e exibe um alerta.

Fluxo do Usuário:

O usuário navega para esta página (geralmente a partir do link "Nova tarefa" na `home.html`). Ele preenche o título e, opcionalmente, a descrição. Ao clicar em "Salvar", o JavaScript valida o título, verifica se o usuário está logado (pelo token), envia os dados para o backend. Se a criação for bem-sucedida, o usuário é notificado e retorna à `home.html`. Se houver erro, uma mensagem é exibida. O usuário também pode clicar em "Voltar" a qualquer momento para retornar à `home.html` sem salvar.

Página de Curiosidade (`public/curiosity.html`)

Esta página é acessada através do link "Esqueceu a senha?" na tela de login (`login.html`). No entanto, ela não implementa um fluxo de recuperação de senha funcional. Em vez disso, apresenta uma mensagem estática e um tanto humorística, instruindo o usuário a retornar.

Estrutura HTML e Estilização:

O `curiosity.html` é uma página HTML muito simples, com a maior parte de sua estilização definida diretamente em um bloco `<style>` dentro do `<head>`, em vez de um arquivo CSS externo.

- **Cabeçalho (`<head>`)**: Inclui metadados básicos, o título (`<title>Deixa de ser curioso</title>`), um link para a fonte `Inter` do Google Fonts, e um bloco `<style>` contendo todo o CSS da página.
    - **CSS Embutido**: Define variáveis CSS (`:root`) para cores (`--cor-primaria` , `--cor-secundaria` , etc.) e a fonte principal (`--fonte-principal`). Aplica estilos básicos ao `html` e `body` (definindo a fonte, removendo margens, definindo cor de fundo e escondendo overflow). Estiliza o container principal (`.container`) para centralizar o conteúdo vertical e horizontalmente usando flexbox. Estiliza o título `<h1>` (tamanho grande, cor primária), o parágrafo `<p>` (tamanho, cor secundária) e o botão de voltar (`a.button`) com cores, preenchimento, bordas arredondadas e alinhamento.
- **Corpo (`<body>`)**: Contém um `<main class="container">` .
    - **Conteúdo (`<div class="content">`)**: Agrupa o título e o parágrafo.
        - Um `<h1>` exibindo ":/".
        - Um `<p>` com a mensagem "**Deixa de ser curioso,** volte para a página anterior."
    - **Botão Voltar (`<a class="button" href="/login.html">`)**: Um link estilizado como botão que direciona o usuário de volta para a página de login.

Funcionalidade JavaScript:

Esta página **não contém nenhum código JavaScript**. É puramente estática.

Fluxo do Usuário:

Se o usuário clicar no link "Esqueceu a senha?" na página de login, ele será direcionado para esta página. Ele verá a mensagem e o botão "Voltar". Clicar no botão "Voltar" o levará de volta à página de login. A página não oferece nenhuma funcionalidade real de recuperação de senha.

Estilização (CSS)

A aparência visual de cada página da aplicação Taskea é definida por arquivos CSS dedicados, localizados em subdiretórios dentro de `public/styles/` . Cada página HTML

principal (`login.html` , `register.html` , `home.html` , `new_task.html`) possui seu próprio diretório (`login/` , `register/` , `home/` , `new_task/`) contendo um arquivo `style.css` .

- **`public/styles/login/style.css`** : Define a estilização específica para a página de login, incluindo layout do formulário, cores, fontes, posicionamento do logo e da caixa geométrica.
- **`public/styles/register/style.css`** : Contém os estilos para a página de cadastro, mantendo uma aparência consistente com a página de login, mas adaptada aos elementos específicos do formulário de registro.
- **`public/styles/home/style.css`** : Estiliza a página principal da aplicação, incluindo o cabeçalho fixo, o menu da conta, a área de exibição das tarefas, os cards de tarefas (`.task-card`) e os menus dropdown. Os estilos para os botões de exclusão (`.botao-excluir` , `.botao-cancelar`) são definidos em um bloco `<style>` embutido no `home.html` , mas poderiam ser movidos para este arquivo para melhor organização.
- **`public/styles/new_task/style.css`** : Define a aparência da página de criação de nova tarefa, estilizando os campos de input, textarea e os botões de ação ("Voltar", "Salvar").

A página `curiosity.html` utiliza CSS embutido em um bloco `<style>` no `<head>` , definindo suas próprias variáveis de cor e layout simples.

Essa abordagem de ter um CSS separado por página permite modularidade, mas também pode levar a alguma duplicação de estilos comuns (cores, fontes, layouts básicos). Uma abordagem alternativa seria ter um arquivo CSS base com estilos comuns e arquivos específicos para sobrescrever ou adicionar estilos por página.

Recursos Visuais (Assets)

Os recursos visuais estáticos da aplicação estão armazenados no diretório `public/assets/` .

- **`public/assets/Taskea_logo.png`** : O logo principal da aplicação Taskea, utilizado nas páginas de login, cadastro, home e nova tarefa.
- **`public/assets/Taskea_logo.pdf`** : Uma versão em PDF do logo (provavelmente para outros usos, não diretamente na interface web).
- **`public/assets/AdobeStock_536912514.png`** : Uma imagem de estoque da Adobe, cujo propósito ou uso na aplicação não está imediatamente claro a partir da estrutura dos arquivos HTML (pode ser um remanescente ou usada de forma não óbvia).

Esses assets contribuem para a identidade visual da aplicação.