Projeto: Controle de Despesas

Objetivo:

Desenvolver uma aplicação de controle de despesas utilizando TypeScript com conexão ao MongoDB, onde os usuários podem cadastrar, visualizar, alterar e excluir despesas. A aplicação deve ser executada no localhost e possuir uma interface amigável que exiba o **total das despesas registradas**.

Descrição do Projeto:

1. **Front-end :**
   - Deve incluir um formulário com campos para Descrição , Valor e Dados de despesa.
   - Um botão Cadastrar Despesa que permite adicionar novas despesas ao sistema.
   - Uma lista que exibe todas as despesas registradas, apresentando:
     - Descrição da despesa
     - Valor no formato monetário (R$)
     - Dados no formato dd/mm/aaaa.
   - Botões Alterar e Excluir para cada item, possibilitando ao usuário atualizar informações de uma especificação específica ou remover o sistema.
   - Exibir o Total das Despesas no formato estimado, calculado com base nas despesas específicas.

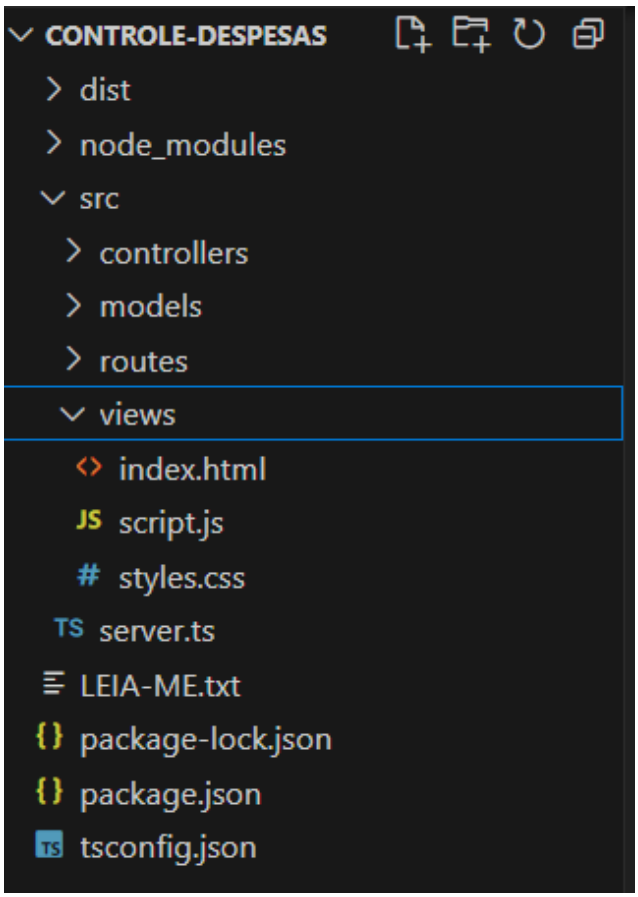
2. **Backend:**
    - Crie um banco de dados MongoDB com uma coleção chamada expenses.
    - Implemente um modelo chamado Expense com os seguintes atributos:
        - description(string): descrição da despesa.
        - amount(número): valor da despesa.
        - date(Data): data da despesa.
    - Desenvolvedor das operações CRUD:
        - Criar: Inserir novas despesas.
        - Leia: Listar todas as despesas cadastradas e calcular o total.
        - Atualização: Atualizar informações de uma despesa.
        - Excluir: remove uma despesa do banco de dados.
3. **Requisitos Funcionais:**
    - Inserção de Despesas: As despesas devem ser cadastradas ao clicar em "Cadastrar Despesa".
    - Edição de Despesas: Ao clicar em "Alterar", deve ser possível atualizar a descrição, valor ou dados da despesa.
    - Exclusão de Despesas: O botão "Excluir" deve remover a despesa da lista e do banco de dados.
    - Cálculo Total: O sistema deve recalcular o total das despesas sempre que uma nova despesa for cadastrada, alterada ou restaurada.
4. **Tecnologias:**
    - Frontend: HTML, CSS e JavaScript.
    - Backend: Node.js com TypeScript, MongoDB e Mongoose para conexão com o banco de dados.
    - Ferramentas: MongoDB Compass para visualização e gerenciamento de banco de dados.
5. **Interface do Usuário:**
    - A interface deve ser limpa e intuitiva, com um layout semelhante ao da imagem fornecida:
        - Título centralizado no topo da página.
        - Formulário de entrada alinhado.
        - Lista de despesas com botões de ação "Alterar" em azul e "Excluir" em vermelho.
        - Exibição do total das despesas centralizada.
6. **Requisitos Extras:**
    - Implemente a validação para evitar valores inválidos, como campos vazios ou valores negativos.

- Os dados devem ter valor padrão para os dados atuais, caso o usuário não insira um dado específico.

Entregáveis :

- Código-fonte completo do projeto, incluindo as instruções para instalação e execução.

Sugestão para organização da estrutura do projeto:



- No seu arquivo da pasta **controllers**, crie uma nova função getTotalExpensesque que usa o método `aggregate` para somar o valor de todas as despesas.

```
// Função para obter o somatório das despesas
export const getTotalExpenses = async (req: Request, res: Response) => {
  try {
      const total = await Expense.aggregate([
          {
              $group: {
                  _id: null,
                  totalAmount: { $sum: "$amount" }
              }
          }
      ]);

      // Caso não haja despesas, retornar total 0
      const totalAmount = total.length > 0 ? total[0].totalAmount : 0;

      res.json({ totalAmount });
  } catch (error) {
      res.status(500).json({ error: 'Erro ao calcular o total das despesas' });
  }
};
```

- Adicione uma Nova Rota para realizar o Somatório no seu arquivo que está dentro da pasta **routes**

```
// Nova rota para obter o somatório das despesas
router.get('/total', getTotalExpenses);
```

- No seu arquivo .js, crie uma função para buscar o total das despesas e exibi-lo na interface.

```
// Função para obter o somatório das despesas
async function fetchTotalExpenses() {
    try {
        const response = await fetch('/api/expenses/total');
        const data = await response.json();
        document.getElementById('totalExpenses').innerText = `Total das Despesas:
        R$${data.totalAmount.toFixed(2)}`;
    } catch (error) {
        console.error('Erro ao buscar o total das despesas:', error);
    }
}

// Carregar despesas e o somatório ao iniciar a página
window.addEventListener('DOMContentLoaded', () => {
    fetchExpenses();
    fetchTotalExpenses();
});
```

script.js:

- fetchExpenses(): busca e exibe todas as despesas cadastradas.
- fetchTotalExpenses(): busca e exibe o somatório das despesas.
- **Total das Despesas**: a função fetchTotalExpenses() é chamada para atualizar o total sempre que uma despesa for adicionada, editada ou deletada.