Projeto de Programação Web: Gestor de Finanças Pessoais

Objetivo do Projeto

Desenvolver um sistema web monolítico para gestão de finanças pessoais utilizando as tecnologias PHP, HTML, CSS e JavaScript. O sistema permitirá a inserção e gerenciamento de receitas e despesas, seguindo os princípios de separação de responsabilidades entre as camadas e persistência de dados utilizando SQLite e PDO.

Requisitos do Projeto

1. **Estrutura do Projeto**
    - O projeto deve ser um monolito em PHP.
    - Separar claramente as responsabilidades entre as camadas:
        - **HTML** para a estrutura das páginas.
        - **CSS** para a estilização.
        - **JavaScript** para a interação dinâmica no front-end.
        - **PHP** para a lógica de negócios e acesso ao banco de dados.
2. **Funcionalidades Principais**
    - **Gestão de Receitas e Despesas**:
        - Inserir receitas (ex.: valor, descrição, data, categoria).
        - Inserir despesas (ex.: valor, descrição, data, categoria).
        - Listar receitas e despesas cadastradas.
        - Editar e excluir receitas e despesas.
    - **Persistência de Dados**:
        - Utilizar banco de dados SQLite para armazenar receitas e despesas.
        - Utilizar PDO para interagir com o banco de dados.
3. **Tecnologias Permitidas**
    - **PHP**:
        - Orientação a objetos.
        - Persistência com PDO.
        - Recomenda-se o uso de PHP puro para a lógica de negócios, mas podem ser utilizados Twig para templates e Slim ou MDMVC para estruturar o projeto.
    - **HTML e CSS**:
        - Utilização de bibliotecas de estilização (ex.: Bootstrap).
    - **JavaScript**:
        - Utilização de bibliotecas como Swal para diálogos e notificações.

Instruções Detalhadas

1. **Estrutura de Pastas**
    - `src/` - Código fonte do projeto.
        - `controllers/` - Controladores para gerenciamento das requisições.
        - `models/` - Modelos representando a lógica de negócio e interação com o banco de dados.
        - `views/` - Arquivos HTML e templates Twig (se aplicável).

    - `public/` - Arquivos acessíveis publicamente (CSS, JavaScript, imagens).
    - `vendor/` - Dependências gerenciadas pelo Composer (se aplicável).
    - `database/` - Arquivos relacionados ao banco de dados SQLite.
2. **Configuração do Banco de Dados**
    - Criar um arquivo SQLite (`financas.db`) para armazenar os dados.
    - Tabelas sugeridas:
        - `receitas` (id, valor, descrição, data, categoria)
        - `despesas` (id, valor, descrição, data, categoria)
3. **Funcionalidades Específicas**
    - **Inserção de Receitas e Despesas**:
        - Formulário para adicionar novas receitas/despesas.
        - Validação dos dados no front-end e back-end.
    - **Listagem e Gerenciamento**:
        - Página para listar todas as receitas e despesas.
        - Opções para editar e excluir registros.
        - Utilização de JavaScript para confirmar ações (ex.: Swal para confirmar exclusões ou alert/confirm com js puro).
4. **Orientação a Objetos**
    - Criar classes para representar as receitas e despesas.
    - Classes para gerenciar a conexão e operações no banco de dados (utilizando PDO).
5. **Front-end e Estilização**
    - Utilizar HTML e CSS para criar uma interface amigável.
    - Utilizar bibliotecas como Bootstrap, Bulma, etc.. para facilitar a estilização.
    - Utilizar JavaScript para melhorar a experiência do usuário (ex.: Swal para notificações).
6. **Entrega e Apresentação**
    - **Código Fonte**: Subir o projeto em um repositório Git (ex.: GitHub, GitLab).
    - **Documentação**: Incluir um arquivo README.md com instruções para configurar e executar o projeto.
    - **Apresentação**: Cada grupo deverá apresentar seu projeto, destacando as principais funcionalidades e o código desenvolvido.

Critérios de Avaliação

1. **Funcionalidade**: O sistema atende aos requisitos propostos.
2. **Qualidade do Código**: Boa prática de programação, uso de orientação a objetos, organização do código.
3. **Interface do Usuário**: Interface intuitiva e bem estilizada.
4. **Documentação**: Clareza e completude da documentação fornecida.
5. **Apresentação**: Capacidade de explicar e demonstrar o funcionamento do sistema, em vídeo.

Dicas e Recomendações

- Planejem bem a divisão de tarefas dentro da equipe.

- Usem um sistema de controle de versão (Git) para gerenciar o desenvolvimento colaborativo.
- Testem as funcionalidades de forma incremental para evitar problemas na integração final.
- Consultem os materiais apresentados em sala de aula e utilizem a documentação oficial das bibliotecas e frameworks escolhidos.