MICROARQUITETURA

1. Módulo Gastos:

- **Objetivo do módulo**: Este módulo concentra-se em fornecer uma estrutura para o cadastro e gerenciamento de informações financeiras, como valores, categorias e períodos associados às despesas dos clientes.
- **Tecnologias utilizadas**: o back-end da aplicação foi produzido em Python; o banco de dados padrão e utilizado foi o SQLite.

2. Estrutura do módulo:

- **Subdiretório:**
  - **admin.py:** Neste arquivo, importamos o modelo ‘Gasto’ do mesmo diretório e o registramos no painel de administração do Django usando admin.site.register(Gasto). Essa ação permite que os administradores do sistema tenham acesso e controlem as instâncias do modelo Gasto diretamente através da interface administrativa do Django.
  - **apps.py:** Configuração da aplicação Django. Neste arquivo, a classe GastosConfig herda de AppConfig e fornece configurações específicas para a aplicação 'gastos'. O atributo name define o nome da aplicação como 'gastos', e default_auto_field especifica o tipo de campo automático padrão para as migrações de banco de dados.
  - **migrations/:** Utilizado para migrações para o banco de dados.
  - **models.py:** O arquivo models.py é utilizado para a definição da estrutura de dados do módulo *gastos*. Neste arquivo, a classe Gasto é criada como uma subclasse de *models.Model*, usada como um modelo de banco de dados gerenciado pelo Django. A classe *Gasto* modela as informações essenciais sobre as despesas dos usuários. Outras classescomo *Usuario* e *Receitas* também estão nesse arquivo.
  - **templates/:** Subdiretório com arquivos HTML representando templates na aplicação.
  - **tests.py:** Arquivo com testes unitários.
  - **forms.py**: define classes de formulários que são usadas para facilitar a entrada e validação de dados. As classes de formulários fornecem uma maneira conveniente de criar formulários HTML a partir dos modelos do Django.
  - **views.py:** contém uma série de funções de visualização que desempenham papéis específicos na interação entre os usuários e o sistema.
  - **urls.py**: define os padrões de URL para mapear as funções de visualização às rotas específicas do sistema.

3. Modelos (models.py)

- **Classes Models:**

  Há quatro classes: ***CustomUser, Usuario, Receita*** e ***Gasto***.

  A classe *CustomUser* herda características do modelo de usuário padrão do Django, permitindo a adição de campos personalizados conforme necessário. A classe *Usuario* modela informações básicas do usuário, como nome de usuário, email e senha. Para proporcionar uma relação entre usuários e suas receitas, a classe *Receita* é definida com campos para categoria, valor e período, sendo vinculada a um usuário específico através de uma chave estrangeira. De maneira similar, a classe *Gasto* reflete informações sobre despesas, incluindo categoria, valor e período, associando-as também a um usuário específico.

4. Templates (templates/):

No diretório de templates do projeto, encontramos uma variedade de arquivos que são responsáveis pela interface do usuário.

- **ajuda.html:** Template para fornecer informações de ajuda ou suporte aos usuários.
- **atualizar_gasto.html:** Página de atualização de informações de um gasto existente.
- **atualizar_receita.html:** Página de atualização de informações de uma receita existente.
- **atualizar_usuario.html:** Página de atualização de informações do usuário.
- **base_home.html:** Template base para a página inicial do sistema.
- **base_logado.html:** Template base para páginas acessadas por usuários logados.
- **cadastrar_gasto.html:** Página de cadastro de novos gastos.
- **cadastrar_receita.html:** Página de cadastro de novas receitas.
- **cadastrar_usuario.html:** Página de cadastro de novos usuários.
- **configuracoes.html:** Página de configurações do usuário.
- **confirmar_remocao.html:** Página de confirmação para a remoção de um item.
- **deletar_receita.html:** Página para deletar uma receita existente.
- **guia_completo.html:** Guia completo do sistema.
- **home.html:** Página inicial do sistema para usuários não logados.
- **home_logado.html:** Página inicial para usuários logados.
- **lancamentos.html:** Página para visualizar todos os lançamentos financeiros.
- **listar_gastos.html:** Página para listar todos os gastos registrados.

- **listar_usuario.html:** Página para listar informações de usuários.
- **login.html:** Página de login do sistema.
- **quem_somos.html:** Página que apresenta informações sobre a equipe ou o propósito do sistema.
- **relatorio.html:** Página para visualizar relatórios e análises financeiras.

5. Views (views.py):

Funções:

- **Função 'home':**
  - Tem como propósito renderizar a página inicial do sistema para usuários não logados.
- **Função 'atualizar_gasto':**
  - Atualiza as informações de um gasto existente com base no formulário fornecido.
- **Função 'remover_gasto':**
  - Remove um gasto específico, solicitando confirmação antes de excluir.
- **Função 'listar_gastos':**
  - Lista todos os gastos registrados no sistema.
- **Função 'cadastrar_gasto':**
  - Gerencia o cadastro de novos gastos, salvando no banco de dados quando o formulário é válido.
- **Função 'login_usuario':**
  - Lida com o processo de login do usuário, autenticando as credenciais fornecidas.
- **Função 'cadastrar_usuario':**
  - Gerencia o cadastro de novos usuários no sistema.
- **Função 'atualizar_usuario':**
  - Atualiza as informações de um usuário existente com base no formulário fornecido.
- **Função 'listar_usuario':**
  - Lista todos os usuários registrados no sistema.
- **Função 'cadastrar_receita':**
  - Gerencia o cadastro de novas receitas, salvando no banco de dados quando o formulário é válido.
- **Função 'atualizar_receita':**
  - Atualiza as informações de uma receita existente com base no formulário fornecido.
- **Função 'listar_receita':**
  - Lista todas as receitas registradas no sistema.
- **Função 'remover_receita':**
  - Remove uma receita específica, solicitando confirmação antes de excluir.

- **Função 'relatorio_gastos_receitas':**
    - Gera um relatório que inclui informações sobre receitas e gastos, calculando o total de gastos, total de receitas.
- **Função 'quem_somos':**
    - Renderiza a página que fornece informações sobre a equipe.
- **Função 'guia_completo':**
    - Renderiza a página contendo um guia completo do sistema.
- **Função 'configuracoes':**
    - Renderiza a página de configurações do usuário.
- **Função 'home_logado':**
    - Renderiza a página inicial para usuários logados.

MACROARQUITETURA

1. Descrição do projeto

- Nosso gerenciador Financeiro é uma plataforma que reúne a organização financeira de nossos usuários. O sistema atua concentrando todas as finanças sejam elas despesas e/ou investimentos. Nosso sistema NÃO É uma corretora, ele atua como uma forma de organizar sua vida financeira lhe dando um norte de como você está andando com seus gastos e atuando como recomendador no caso de investimentos para determinado perfil de investidor.

2. Tecnologias Utilizadas no Front-End

- No front-end foi utilizado o bootrstrap (framework de design responsivo) e o Django (framework web em Python).

3. Descrição dos Componentes do Sistema:

- **projeto_gfp/:** Representa a estrutura geral do projeto.
- **gastos/:** Módulo central de gerenciamento financeiro.
  - **models.py:** Define a estrutura de dados relacionada aos gastos.
  - **views.py:** Lógica de visualização e interação com os dados.
  - **templates/:** Templates HTML para a interface do usuário.
  - **tests.py:** Testes relacionados ao módulo de gastos.
- **projeto_gastos/:** Configurações e estrutura do projeto Django.
  - **settings.py:** Configurações globais do projeto.
  - **urls.py:** Configuração de URLs.

4. Relação Entre os Componentes:

- **projeto_gfp/ e projeto_gastos/:** A pasta principal do projeto contém as configurações gerais do Django, enquanto **projeto_gastos/** lida especificamente com as configurações do projeto Django, como URLs e e ajustes gerais.
- **gastos/ e projeto_gastos/settings.py:** O módulo **gastos/** interage com as configurações globais do projeto definidas em **projeto_gastos/settings.py**.
- **gastos/models.py e gastos/views.py:** As views definidas em **views.py** interagem com os modelos definidos em **models.py**.

- **gastos/templates/ e gastos/views.py:** Os templates HTML presentes em **gastos/templates/** são renderizados pelas views definidas em **gastos/views.py**.