Curso: Engenharia de Software
Disciplina: Arquitetura de Software
Período: 7º Ano/Semestre: 2024/1
Docente: Rafael C. Sacoman
Avaliação Prática: vlr10pts
Entrega 14/06/2024

Caso de Uso: Sistema de Vendas de Produtos Naturais

- **Ator Principal**: Funcionário (acesso apenas no lançamento de vendas)
- **Ator Secundário**: Administrador do Sistema (acesso total)

Casos de Uso:

- **UC01 - Autenticação:**
  - O funcionário acessa o sistema e é solicitado a autenticar-se inserindo seu nome de usuário e senha.
  - Se as credenciais estiverem corretas, o cliente é autenticado e pode acessar as funcionalidades do sistema.
  - Estrutura de dados: login, nome, email.

- **UC02 - Cadastro de Produto:**
  - O administrador do sistema pode adicionar novos produtos ao catálogo.
  - Após o cadastro, os produtos são exibidos no site para os clientes.
  - Estrutura de dados: nome, descrição, preço custo, preço vendas, peso, quantidade comprado, quantidade vendida, Fabricante, Grupo e Subgrupo

- **UC03 - Cadastro do Fabricante:**
  - O administrador pode cadastrar informações sobre os fabricantes dos produtos naturais.
  - Esses dados são associados aos produtos para fornecer informações adicionais aos clientes.
  - Estrutura de dados: nome fantasia, razão social, cnpj, endereço, telefone, email, vendedor

- **UC04 - Cadastro de Grupo e UC05 - Subgrupo:**
  - Os produtos naturais podem ser agrupados em categorias e subcategorias para facilitar a navegação.
  - O administrador pode criar novos grupos e subgrupos, por exemplo, "Alimentos Orgânicos" pode ser um grupo, com "Frutas", "Verduras" e "Grãos" como subgrupos.
  - Estrutura de dados: nome e descrição

- **UC06 - Lançamento de Vendas:**
  - O funcionário pode adicionar itens que foram escolhidos para compra.
  - O sistema calcula valor de venda e quantidade do produto.
  - Ao finalizar todos os itens escolhidos, o sistema registra a venda, atualiza o estoque.

- Estrutura de dados: Produto, Fabricante, Grupo, SubGrupo, preço venda, quantidade, data e hora venda

- **UC07 - Visualizar Vendas:**
  - O funcionário pode visualizar através de gráficos a situação atual do comércio.
  - Visualiza na mesma tela 3x3 gráficos* com as seguintes temáticas
    - **Gráfico de Linha**: apresentar o valor do custo Total e o valor venda Total mensal do ano corrente. (Vendas)
    - **Gráfico de Barras**: apresentar o valor da quantidade comprado total e a quantidade vendida total mensal do ano corrente. (Produto)
    - **Gráfico de Dispersão**: apresentar o percentual de lucro dos produtos vendidos mensal do ano corrente. (Produto)
    - **Gráfico de Pizza**: apresentar os 3 produtos mais vendidos em quantidade mensal do ano corrente. (Vendas)
    - **Gráfico de Barras e Linha**: apresentar os 4 grupos de produtos mais vendidos mensal do ano corrente com a meta de quantidade >= 1000 unidades. (Vendas)
    - **Tabela Analítica**: listar os produtos que estão com estoque baixo e ordem decrescente.

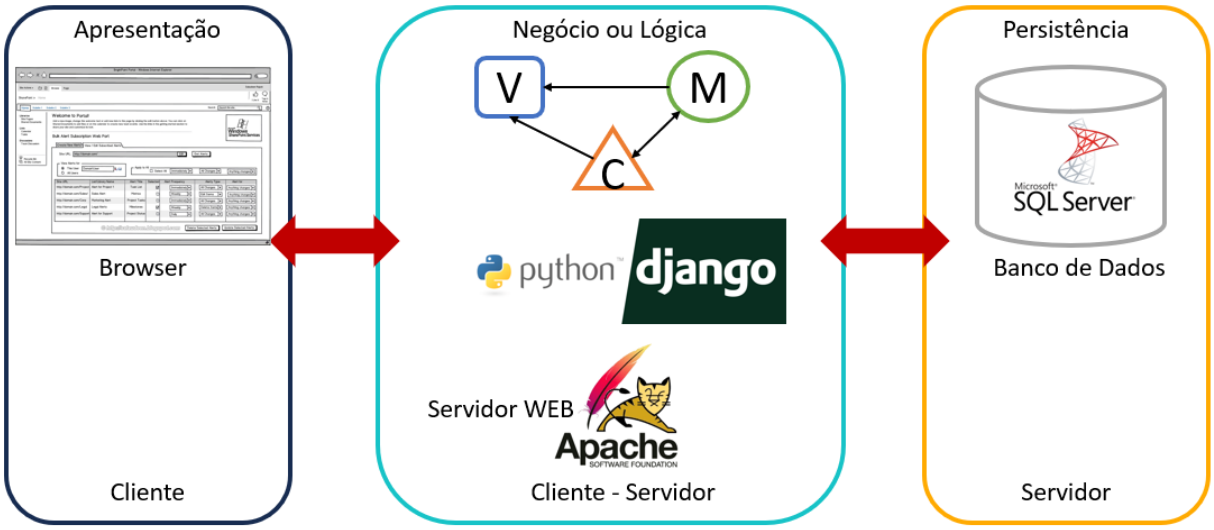
*Use as libs plotly.express e plotly.graph_objs

Fluxo de Eventos Típico:

1. O funcionário acessa o sistema da loja online de produtos naturais.
2. Ele faz login usando seu nome de usuário e senha.
3. O funcionário navega pelos produtos, filtrando por fabricante, grupo ou subgrupo, se desejar.
4. Ele adiciona os produtos desejados aos itens de compras e o sistema calcula valor de venda e quantidade do produto.
5. Ao finalizar os itens escolhidos, o sistema registra a venda, atualiza o estoque

Arquitetura do sistema

Linguagem: Python
Framework: Django
ServidorWeb: Apache usar módulo *mod_wsgi*
Banco de dados: SQlite
Estilo de arquitetural: estilo em camadas (3) com estilo MVC, conforme ilustração abaixo:



Design Patterns:

- **Padrões de criação:**
  - **Factory Method**: Fornece uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
  - **Singleton:** Garante que uma classe tenha somente uma instância e fornece um ponto global de acesso para ela.
- **Padrões comportamentais:**
  - **Observer**: Define uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são automaticamente notificados e atualizados