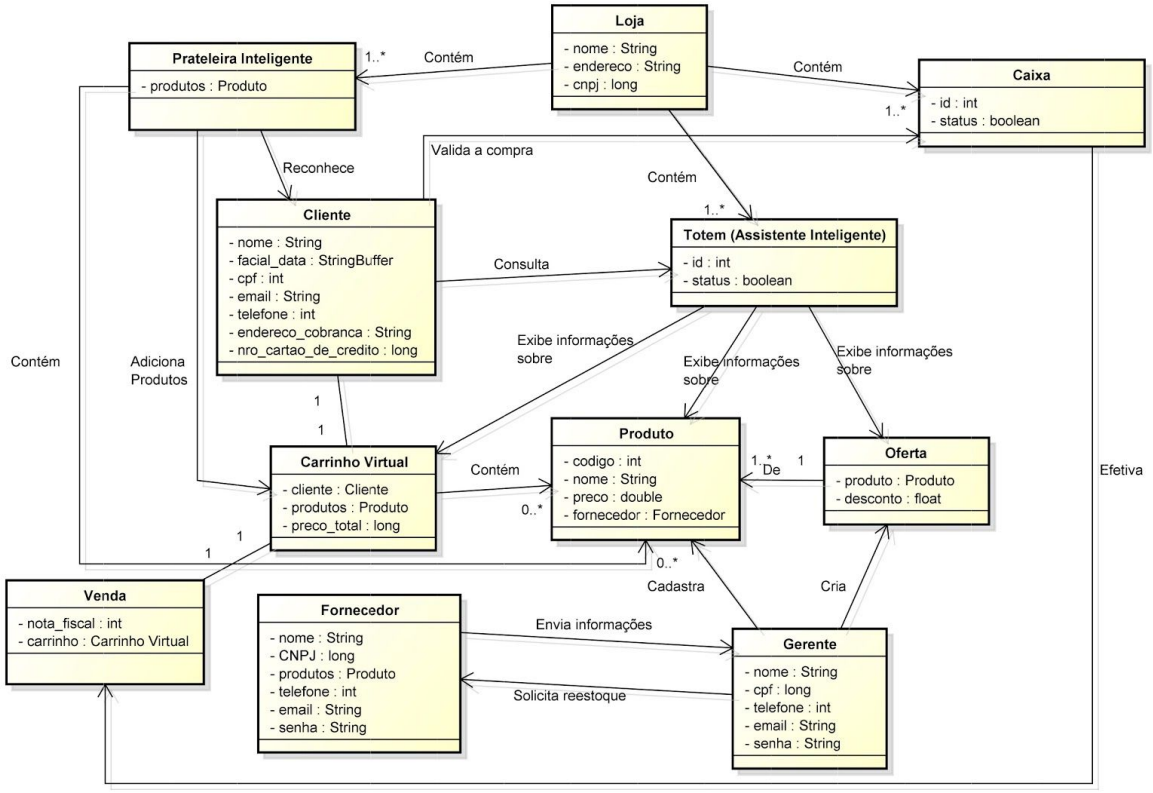
INF01127 – Engenharia de Software N – Trabalho Prático: Etapa 2 **Grupo 02**

Membros: Eduardo Spitzer Fischer <efischer@outlook.com.br>
Gabriel Lando <gabriel@lando.net.br>
Rodrigo Paranhos Bastos <ro.p.bastos@gmail.com>

Produto de Software: Smart Retail
Product Owner: G02

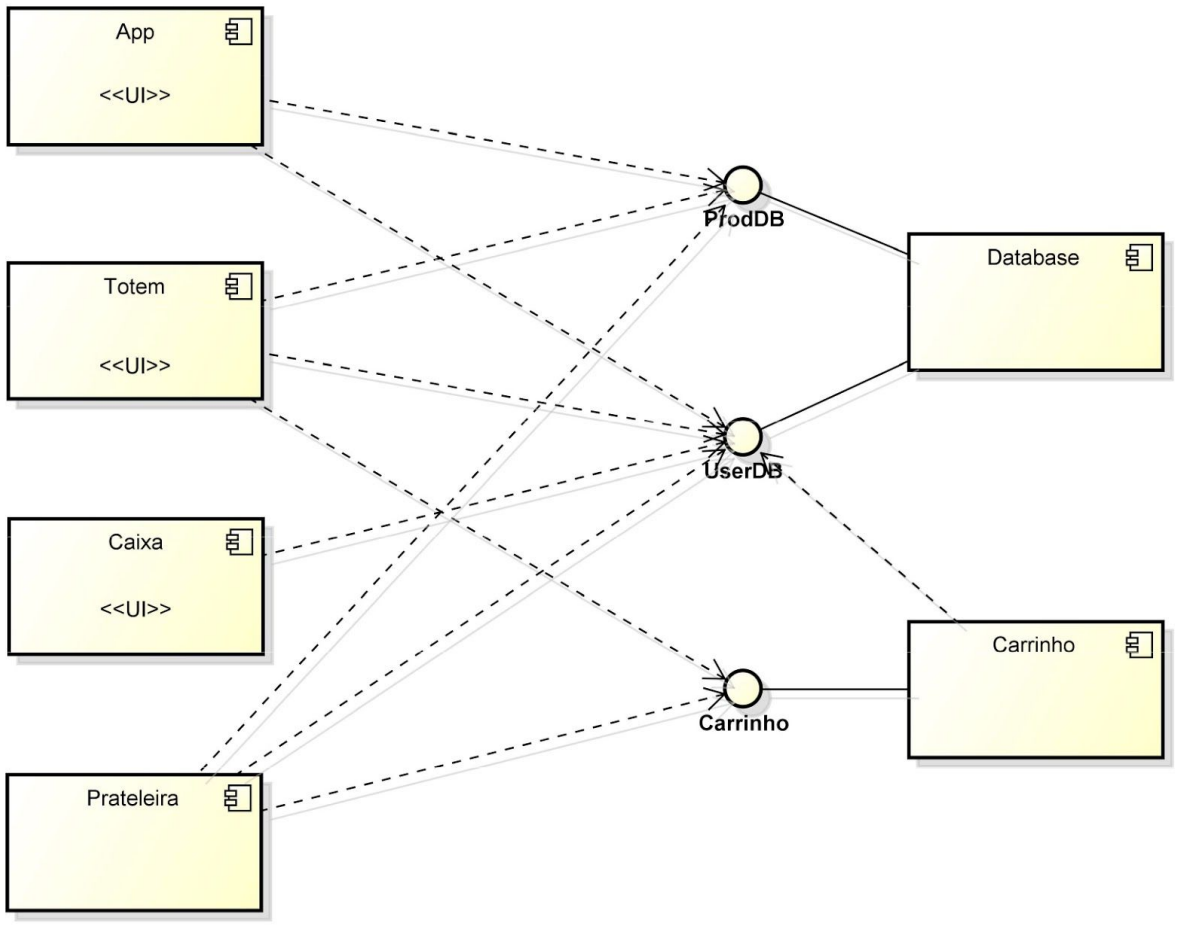
1. Modelo de Análise



Modelo Conceitual

2. Modelo Arquitetural

2.1 Arquitetura do Software



App: Aplicativo para smartphone responsável por cadastrar produtos e estoque. É usado somente pelo gerente do sistema. O login é feito através de um endereço de e-mail e senha previamente cadastrados.

Totem: Dispositivos distribuídos pelos corredores das lojas. Neles, os clientes podem consultar os itens adicionados no carrinho, valor total dos itens, entre outras informações. Ao cliente se aproximar de um totem, ele deve reconhecer o cliente através de um sistema de reconhecimento facial.

Caixa: Dispositivos que estarão na entrada dos estabelecimentos. Eles devem reconhecer o cliente na entrada e saída do estabelecimento, através de reconhecimento facial. Quando o cliente for entrar, se ele detectar que é um cliente novo, deve exigir que ele faça um cadastro, fornecendo informações como nome, CPF, dados do cartão de crédito para cobrança, entre outros. Na saída, deve realizar, automaticamente, o pagamento dos itens do carrinho do mesmo.

Prateleira: Deve saber a quantidade de cada produto que está na sua localidade. Quando um cliente pega um produto e o adiciona em seu carrinho, a prateleira deve perceber essa ação, detectar qual cliente pegou o produto (através de reconhecimento facial) e adicionar este produto em seu carrinho virtual.

Database: Base de dados que deve armazenar todos os dados de usuários e produtos. Deve prover essas informações sempre que requisitado através de suas interfaces.

Carrinho: Carrinho virtual que deverá armazenar as informações dos produtos e quantidade que cada cliente retirou da prateleira.

2.2 Tecnologias

Serão usadas as seguintes tecnologias:

- Linguagens de programação: Java / C++
    - Comunicação com o Banco de Dados;
    - Processamento das transações;
    - Simulação dos dispositivos;

- IDE de desenvolvimento:
    - Visual Studio;
    - Eclipse;

- Interface Gráfica:
    - Java Swing;
    - Windows Form;

- Banco de dados:
    - MySQL;

- Firebase Authentication
    - Autenticação de clientes e usuários;

2.3 Interfaces

2.3.1 UserDB

- *registerClient(client_data) : bool success*

Recebe uma estrutura com dados de um novo cliente a ser cadastrado (nome, facial_data, CPF, telefone, email, endereco_cobranca, nro_cartao_de_credito). Se o cadastro for realizado com sucesso, retorna *True*, do contrário retorna *False*.

- *getClientCollection(filters) : client_collection*

Retorna um conjunto de clientes filtrado do database geral com base nos filtros configurados em filters. Os valores dos filtros são determinados pelos dispositivos inteligentes e consistem em altura, faixa etária e sexo estimados para os clientes filmados pelos dispositivos. Além disso, um dos filtros também marca se devem ser retornados clientes que estão dentro do estabelecimento ou não, para que dispositivos que usem apenas dados de clientes que estão dentro dos estabelecimentos não gastem recursos recuperando dados de clientes fora dos estabelecimentos.

- *clientInStore(client_id, bool) : bool success*

Aciona uma flag, informando se o cliente se encontra ou não dentro de uma das lojas. *True* significa que o cliente se encontra dentro e *False* significa que o cliente não se encontra em nenhuma loja. Será usado nos filtros acima descritos.

2.3.2 ProdDB

- *getProductData(product_id) : product_data*

Retorna as informações acerca de um produto: nome, preço, fornecedor, preço de reestoque.

- *registerProduct(product_data) : bool success*

Registra um novo produto com as características presentes em *product_data* (nome, preço, código). Retorna *True* se o cadastro tiver obtido sucesso, *False* do contrário.

2.3.3 Carrinho

- *addProduct(product_id, amount) : bool success*

Adiciona a quantidade *amount* do produto identificado por *product_id* no carrinho. Retorna *True* em caso de sucesso e *False* do contrário.

- *removeProduct(product_id) : bool success*

Remove um produto do tipo identificado por *product_id* do carrinho, retorna *False* se o produto não tiver sido encontrado ou se a remoção tiver falhado por qualquer outro motivo. Retorna *True* do contrário.

- *getContents(void) : cart_data*

retorna uma lista dos produtos do carrinho e suas quantidades.

- *getTotal(void) : float total*

Retorna o preço total dos produtos do carrinho.

3. Tarefas

1. Iniciar ambiente de desenvolvimento (Slot de tempo: 2 horas)
    a. *Requisitos:* Iniciar um novo projeto na IDE, para ser usado pelos componentes do grupo no desenvolvimento da aplicação.

2. Classe Prateleira Inteligente (Slot de tempo: 1,5 horas)
    a. *Requisitos:* Implementar a classe Prateleira Inteligente, juntamente com seus atributos.
    b. Componentes: prateleira

3. Sistema de retirada/reposição da Prateleira (Slot de tempo: 2 horas)
    a. *Requisitos:* Implementar a classe Prateleira Inteligente, juntamente com seus atributos e métodos capazes de controlar a detecção de um cliente, a

remoção de um produto da prateleira, a devolução do produto à prateleira e a reposição de um produto na prateleira.
    b. *Componentes:* prateleira.take_product, prateleira.return_product, prateleira.fill

4. Reconhecimento Facial da Prateleira (Slot de tempo: 2,5 horas)
    a. *Requisitos:* Identificar um cliente assim que ele se aproxima da prateleira
    b. *Componentes:* prateleira.recognize_client

5. Classe Totem (Slot de tempo: 1 hora)
    a. *Requisitos:* Implementar a classe Totem, juntamente com seus atributos.
    b. *Componentes:* totem

6. Reconhecimento Facial Totem (Slot de tempo: 1 hora)
    a. *Requisitos:* Identificar um cliente assim que ele se aproxima do totem
    b. *Componentes:* totem.recognize_client

7. Função carrinho de compras no Totem (Slot de tempo: 1,5 horas)
    a. *Requisitos:* Mostrar o carrinho de compras quando o cliente se aproxima do totem.
    b. *Componentes:* totem.show_cart

8. Recomendações de produtos no Totem (Slot de tempo: 2 horas)
    a. *Requisitos:* Totem exibe recomendações personalizadas de produtos para cada cliente.
    b. *Componentes:* totem.show_recomendations

9. Classe Caixa (Slot de tempo: 1 hora)
    a. *Requisitos:* Implementar a classe Caixa, juntamente com seus atributos.
    b. *Componentes:* caixa

10. Reconhecimento Facial Caixa (Slot de tempo: 1 hora)
    a. *Requisitos:* Identificar um cliente assim que ele se aproxima do caixa
    b. *Componentes:* caixa.recognize_client

11. Aprovação da compra (Slot de tempo: 1,5 horas)
    a. *Requisitos:* Ao se dirigir ao caixa, o usuário visualiza o seu carrinho de compras juntamente com o valor de cada produto e o valor total, para efetuar a compra o cliente deve aprovar a finalização.
    b. *Componentes:* caixa.validate_cart

12. Efetuação da compra (Slot de tempo: 1,5 horas)
    a. *Requisitos:* A compra é efetuada e o valor é cobrado no cartão de crédito do cliente.
    b. *Componentes:* totem.purchase

13. Implementação Carrinho Virtual (Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Carrinho Virtual, juntamente com seus atributos e métodos para adicionar/remover produtos, finalizar a compra, calcular valor total e limpar carrinho.

    b. *Componentes:* carrinho, carrinho.add_product, carrinho.remove_product, carrinho.checkout, carrinho.get_total, carrinho.clear_items

14. Implementação Loja (Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Loja juntamente com seus atributos e métodos.
    b. *Componentes:* loja

15. Implementação Produto(Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Produto juntamente com seus atributos e métodos.
    b. *Componentes:* produto

16. Implementação Oferta(Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Loja juntamente com seus atributos e métodos para criar oferta, adicionar/remover produtos e seus respectivos descontos e finalizar oferta.
    b. *Componentes:* oferta, oferta.set_product, oferta.set_discount, oferta.end

17. Implementação Venda(Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Venda juntamente com seus atributos e métodos.
    b. *Componentes:* venda

18. Implementação Fornecedor (Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Fornecedor juntamente com seus atributos e métodos.
    b. *Componentes:* fornecedor

19. Implementação Gerente(Slot de tempo: 1,5 horas)
    a. *Requisitos:* Criar a classe Gerente juntamente com seus atributos e métodos.
    b. *Componentes:* gerente