DOCUMENTO DE MODELAGEM UML

APLICATIVO MOBILE DE CUPCAKES GOURMET

Versão: 2.5

Data: 17/11/2025

Elaborado por: Vinicius Alves Leon Rodriguez

Metodologia: Scrum + TDD

Arquitetura: MVC + Microservices

Ferramentas: Draw.io, Lucidchart, Figma, MindMeister

1. INTRODUÇÃO

1.1 Objetivo

Este documento descreve a modelagem UML (Unified Modeling Language) do projeto **Aplicativo Mobile de Cupcakes Gourmet**, que visa representar visualmente a estrutura, comportamento e interação dos componentes do sistema.

A modelagem fornece uma visão clara das funcionalidades, fluxos de dados e integrações, servindo como base técnica para o desenvolvimento backend (Python + Flask/FastAPI) e frontend (Ionic + Capacitor).

1.2 Finalidade do Documento

- Estabelecer visão unificada da arquitetura do sistema
- Documentar requisitos funcionais por meio de diagramas
- Facilitar comunicação entre stakeholders técnicos e não-técnicos
- Servir como referência para implementação e testes

2. ESCOPO DO SISTEMA

O aplicativo mobile permitirá:

- Exibir catálogo de produtos (≥40 sabores de cupcakes e frozen yogurts)
- Realizar pedidos e pagamentos online (Pix/cartão)
- Notificar o usuário sobre o status da entrega
- Permitir ao administrador gerenciar estoque, produtos e relatórios de vendas
- Garantir segurança e acessibilidade (modo escuro, contraste e login seguro)

2.1 Atores do Sistema

Ator	Descrição
Cliente	Usuário final que realiza pedidos através do aplicativo
Administrador	Gerencia produtos, estoque, usuários e visualiza relatórios
Entregador	Responsável pela entrega física dos pedidos
Sistema de Pagamento	API externa (Mercado Pago) para processamento de transações
Sistema de Entregas	Serviço externo para rastreamento logístico

Sistema de Notificações	Serviço de push notifications

3. DIAGRAMAS UML ELABORADOS

A modelagem contempla os seguintes diagramas:

1. **Diagrama de Caso de Uso Geral** - Identifica atores e funcionalidades principais
2. **Casos de Uso Expandidos** - Detalham três processos críticos do sistema
3. **Diagrama de Classes** - Representa estruturas de dados e relacionamentos
4. **Diagrama de Sequência** - Demonstra interação temporal entre componentes

4. DIAGRAMA DE CASO DE USO GERAL

4.1 Atores Principais

- **Cliente**: Cadastro, login, consulta ao catálogo, realização de pedidos, pagamentos e acompanhamento de entregas
- **Administrador**: Gerenciamento de produtos, estoque, usuários, relatórios e segurança
- **Entregador**: Visualização de pedidos atribuídos, aceite/recusa de entregas e atualização de status
- **API Mercado Pago**: Processamento de transações financeiras
- **Sistema de Entrega**: Atualização de status logístico
- **Sistema de Notificações**: Envio de notificações push

4.2 Casos de Uso Identificados

Funcionalidades do Cliente

- **UC01** - Cadastrar-se
- **UC02** - Autenticar-se / Fazer Login
- **UC03** - Visualizar Catálogo
- **UC04** - Efetuar Pedido
- **UC05** - Processar Pagamento *(relacionamento include)*
- **UC06** - Acompanhar Pedido
- **UC07** - Receber Notificações Push

Funcionalidades do Administrador

- **UC08** - Atribuir Pedido ao Entregador
- **UC09** - Gerenciar Catálogo de Produtos
- **UC10** - Gerenciar Estoque e Custos
- **UC11** - Gerar Relatórios e Dashboards
- **UC12** - Gerenciar Usuários
- **UC13** - Gerenciar Segurança e Logs

Funcionalidades do Entregador

- **UC14** - Visualizar Pedidos Atribuídos
- **UC15** - Aceitar / Recusar Entrega
- **UC16** - Atualizar Status da Entrega
- **UC17** - Confirmar Entrega no Destino

Funcionalidades dos Sistemas Externos

- **UC18** - Confirmar Pagamento *(Sistema de Pagamento)*
- **UC19** - Notificar Falha de Pagamento *(Sistema de Pagamento)*
- **UC20** - Receber Pedido para Entrega *(Sistema de Entregas)*
- **UC21** - Registrar Status da Entrega *(Sistema de Entregas)*

4.3 Relacionamentos Entre Casos de Uso

Include (Obrigatório)

- **UC04 (Efetuar Pedido)** → include → **UC05 (Processar Pagamento)**
- **UC06 (Acompanhar Pedido)** → include → **UC07 (Receber Notificações Push)**

Estes relacionamentos indicam que os casos de uso incluídos são executados automaticamente como parte do fluxo principal.

5. CASOS DE USO EXPANDIDOS

Três casos de uso críticos foram expandidos com detalhamento completo:

5.1 UC04 - Efetuar Pedido

Fluxo completo desde a seleção de produtos até o acionamento do pagamento, incluindo validação de estoque e tratamento de exceções.

5.2 UC05 - Processar Pagamento

Integração com gateway de pagamento externo, validação de dados e tratamento de aprovação/rejeição de transações.

5.3 UC16 - Atualizar Status da Entrega

Processo de atualização logística com validação de sequência de status e notificação automática ao cliente.

Detalhamento completo disponível no documento de Casos de Uso Expandidos.

6. DIAGRAMA DE CLASSES

6.1 Classes Identificadas

Usuário

- **Atributos:** id_usuario, nome, cpf, telefone, email, senha_hash, tipo_usuario
- **Métodos:** autenticar(), atualizarPerfil(), validarCPF()

Endereço

- **Atributos:** id_endereco, id_usuario, cidade, rua, numero, cep, complemento
- **Métodos:** validarCEP(), formatarEndereco()

Produto

- **Atributos:** id_produto, nome, descricao, preco, quantidade_estoque, imagem_url, categoria, ativo
- **Métodos:** atualizarEstoque(), validarDisponibilidade(), calcularPrecoTotal()

Pedido

- **Atributos:** id_pedido, id_usuario, id_endereco, data_pedido, valor_total, status, data_entrega_prevista
- **Métodos:** calcularTotal(), atualizarStatus(), validarPedido()

Item_Pedido

- **Atributos:** id_item, id_pedido, id_produto, quantidade, preco_unitario, subtotal
- **Métodos:** calcularSubtotal(), validarQuantidade()

Pagamento

- **Atributos:** id_pagamento, id_pedido, metodo_pagamento, valor, status_transacao, data_pagamento, codigo_transacao
- **Métodos:** processarPagamento(), validarTransacao(), gerarComprovante()

Cartão

- **Atributos:** id_cartao, id_usuario, numero_cartao_hash, tipo_cartao, cvv_hash, data_validade, bandeira
- **Métodos:** validarCartao(), verificarValidade(), criptografarDados()

6.2 Relacionamentos

- **Usuário** 1:N **Endereço** - Um usuário pode ter múltiplos endereços
- **Usuário** 1:N **Pedido** - Um usuário pode realizar múltiplos pedidos
- **Usuário** 1:N **Cartão** - Um usuário pode cadastrar múltiplos cartões
- **Pedido** 1:N **Item_Pedido** - Um pedido contém múltiplos itens
- **Produto** 1:N **Item_Pedido** - Um produto pode estar em múltiplos pedidos
- **Pedido** 1:1 **Pagamento** - Cada pedido possui um pagamento associado
- **Pedido** N:1 **Endereço** - Cada pedido é entregue em um endereço

7. DIAGRAMA DE SEQUÊNCIA

7.1 Fluxo: Realizar Pedido com Pagamento

Participantes:

- Cliente (Ator)
- Interface Mobile
- Controller Pedido
- Controller Pagamento
- API Mercado Pago
- Banco de Dados

Sequência de Interações:

1. **Cliente → Interface:** Solicita login
2. **Interface → Controller:** Valida credenciais
3. **Controller → BD:** Consulta usuário
4. **BD → Controller:** Retorna dados do usuário
5. **Controller → Interface:** Confirma autenticação
6. **Cliente → Interface:** Seleciona produtos do catálogo
7. **Interface → Controller:** Requisita dados dos produtos
8. **Controller → BD:** Consulta produtos disponíveis
9. **BD → Controller:** Retorna lista de produtos
10. **Cliente → Interface:** Adiciona produtos ao carrinho
11. **Interface:** Calcula valor total
12. **Cliente → Interface:** Confirma pedido
13. **Interface → Controller Pedido:** Cria novo pedido
14. **Controller Pedido → BD:** Insere pedido
15. **BD → Controller Pedido:** Confirma inserção
16. **Controller Pedido → Controller Pagamento:** Inicia processamento de pagamento
17. **Controller Pagamento → API Mercado Pago:** Envia dados da transação
18. **API Mercado Pago → Controller Pagamento:** Retorna status (aprovado/recusado)
19. **Controller Pagamento → BD:** Atualiza status do pagamento
20. **Controller Pagamento → Interface:** Notifica resultado
21. **Interface → Cliente:** Exibe confirmação ou erro

7.2 Cenários Alternativos

Pagamento Recusado:

- API retorna status de falha
- Sistema reverte criação do pedido

- Cliente é notificado e pode tentar novamente

Produto Indisponível:

- Validação de estoque identifica inconsistência
- Sistema remove item do carrinho
- Cliente é informado antes da finalização

8. MAPA DE AFINIDADE

O mapa de afinidade foi utilizado durante as sessões de brainstorming para organizar requisitos funcionais e não-funcionais em categorias:

Categorias Identificadas:

1. **Funcionalidades do Cliente** - Cadastro, login, catálogo, pedidos, pagamento
2. **Funcionalidades Administrativas** - Gestão de produtos, estoque, relatórios
3. **Integrações Externas** - Mercado Pago, sistema de entregas, notificações
4. **Requisitos Não-Funcionais** - Segurança, acessibilidade, performance
5. **Entrega e Logística** - Rastreamento, atualização de status

9. FERRAMENTAS UTILIZADAS

Ferramenta	Finalidade
Draw.io	Criação de diagramas UML (casos de uso, classes, sequência)
Lucidchart	Diagramação colaborativa e revisão em equipe
Figma	Prototipação de interfaces e design system
MindMeister	Mapa mental e organização de ideias
Trello/Jira	Gerenciamento de tarefas e sprints Scrum
GitHub	Controle de versão e documentação técnica
MySQL Workbench	Modelagem de banco de dados relacional
Pytest	Framework para Test-Driven Development (TDD)
Word/PDF	Documentação final e entrega formal

10. CONCLUSÃO

A modelagem UML do **Aplicativo Mobile de Cupcakes Gourmet** fornece uma visão clara e integrada de todos os componentes do sistema, facilitando:

- **Desenvolvimento:** Base técnica sólida para implementação backend e frontend
- **Comunicação:** Linguagem visual comum entre desenvolvedores, gestores e stakeholders
- **Escalabilidade:** Arquitetura preparada para crescimento e novas funcionalidades
- **Segurança:** Fluxos bem definidos para proteção de dados sensíveis
- **Manutenibilidade:** Documentação que facilita manutenção e evolução do sistema

A documentação está alinhada com as melhores práticas de Engenharia de Software, metodologia Scrum e padrões UML, garantindo qualidade técnica e conformidade com os requisitos do projeto.

Histórico de Revisões

Versão	Data	Autor	Alterações
1.0	11/11/2025	Time Dev	Versão inicial
2.0	15/11/2025	PO	Refinamento backlog
2.5	17/11/2025	Scrum Master	Ajustes finais