Dilto’s App - Sistema de gerenciamento de lojas de conveniência

Sistema completo para gerenciamento de loja de conveniência com funcionalidades de controle de produtos, horários de funcionamento e reservas de mesas de sinuca.

1 - Visão Geral

Objetivos do Sistema

- Facilitar o gerenciamento de produtos e estoque
- Controlar horários de funcionamento (padrão e especiais)
- Gerenciar reservas de mesas de sinuca
- Separar funcionalidades entre administrador e cliente

Público-Alvo

- Administradores: Donos de loja que precisam gerenciar produtos, horários e reservas
- Clientes: Usuários que desejam visualizar produtos, horários e fazer reservas de mesas

Funcionalidades - Módulo Administrativo

Gerenciamento de Produtos

- Cadastrar novos produtos
- Listar todos os produtos
- Buscar produtos por ID, código, nome ou categoria
- Atualizar informações de produtos
- Deletar produtos
- Gerenciar estoque (entrada e saída)
- Alertas de estoque baixo

Gerenciamento de Horários

- Configurar horários padrão da semana
- Definir horários especiais (feriados, eventos)
- Visualizar status atual (aberto/fechado)
- Remover horários especiais

Gerenciamento de Mesas de Sinuca

- Visualizar mapa de reservas do dia
- Criar novas reservas (30 minutos, apenas para hoje)
- Remover reservas específicas
- Cancelar todas reservas de uma mesa (ex: mesa quebrada)
- Validar horário de funcionamento para reservas

Funcionalidades - Módulo Cliente

- Buscar produtos por nome ou categoria
- Visualizar horários de funcionamento
- Ver status atual da loja (aberto/fechado)
- Visualizar mapa de mesas de sinuca
- Criar reservas de mesa

2 - Arquitetura

Arquitetura em Camadas (Layered Architecture)

Justificativa: Escolhemos a arquitetura em camadas para garantir:

- Separação de Responsabilidades: Cada camada tem uma função específica
- Manutenibilidade: Mudanças em uma camada não afetam as outras
- Testabilidade: Camadas podem ser testadas independentemente
- Reusabilidade: Lógica de negócio pode ser reutilizada

Diagrama de Arquitetura

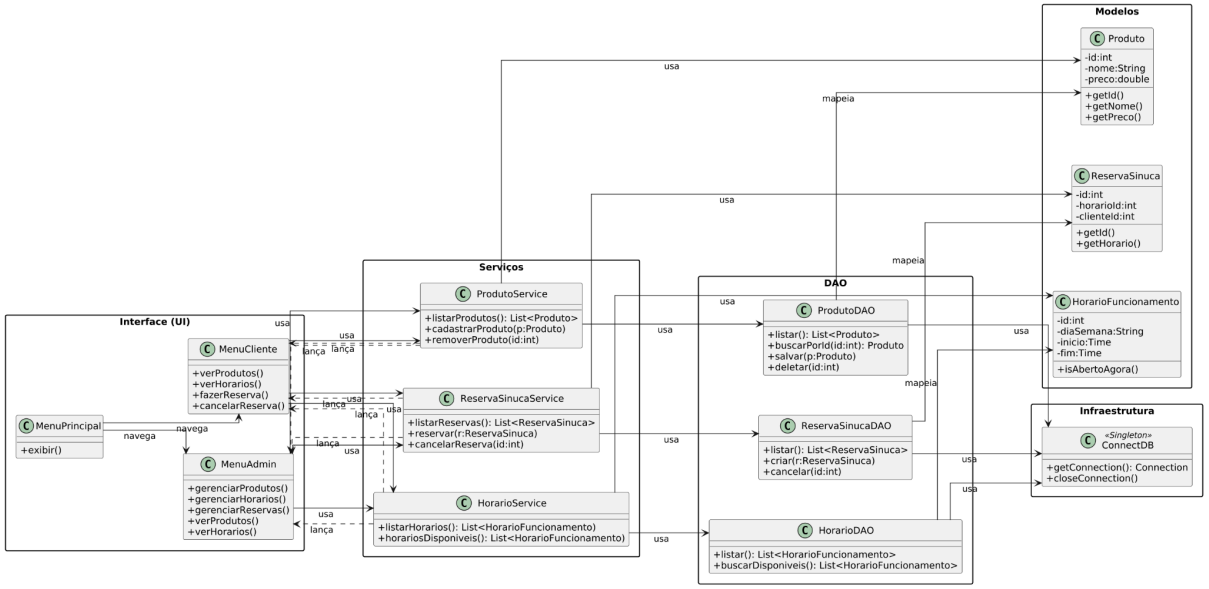

Diagrama ampliado disponível no github.

3 - Padrões de Projeto

DAO (Data Access Object)

Implementação: Classes ProdutoDAO, HorarioDAO, ReservaSinucaDAO

Justificativa:

- Encapsulamento: Lógica de acesso ao banco separada da lógica de negócio
- Manutenibilidade: Mudanças no banco não afetam outras camadas
- Testabilidade: Mock de DAOs para testes unitários
- Flexibilidade: Fácil migração para outro banco de dados

Diagrama:

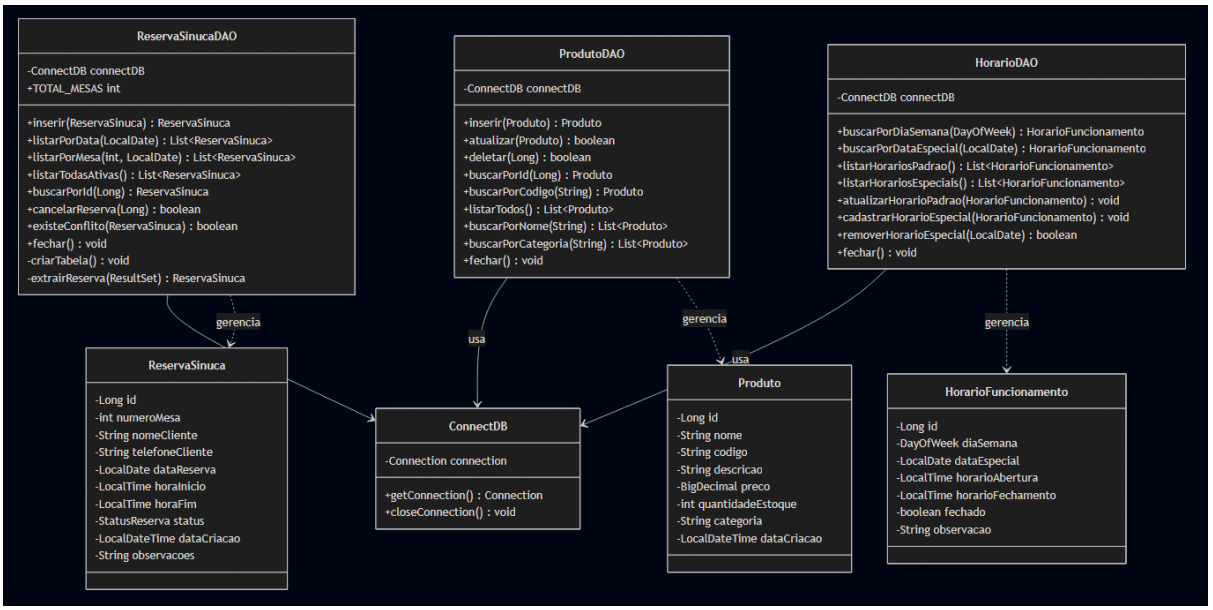

Diagrama ampliado disponível no github.

Service
Implementação: Classes HorarioService, ProdutoService, ReservaSinucaService
Justificativa:

- Separação de responsabilidades: Cada camada tem responsabilidade clara, o que facilita a manutenção e evolução.
- Centralização da lógica de negócio: Regras, validações, orquestração de operações ficam num único lugar
- Melhor testabilidade: Services podem ser testados isoladamente.
- Controle de requisições e consistência: A camada de serviço é o local natural para definir frlimites entre requisições

Diagrama:

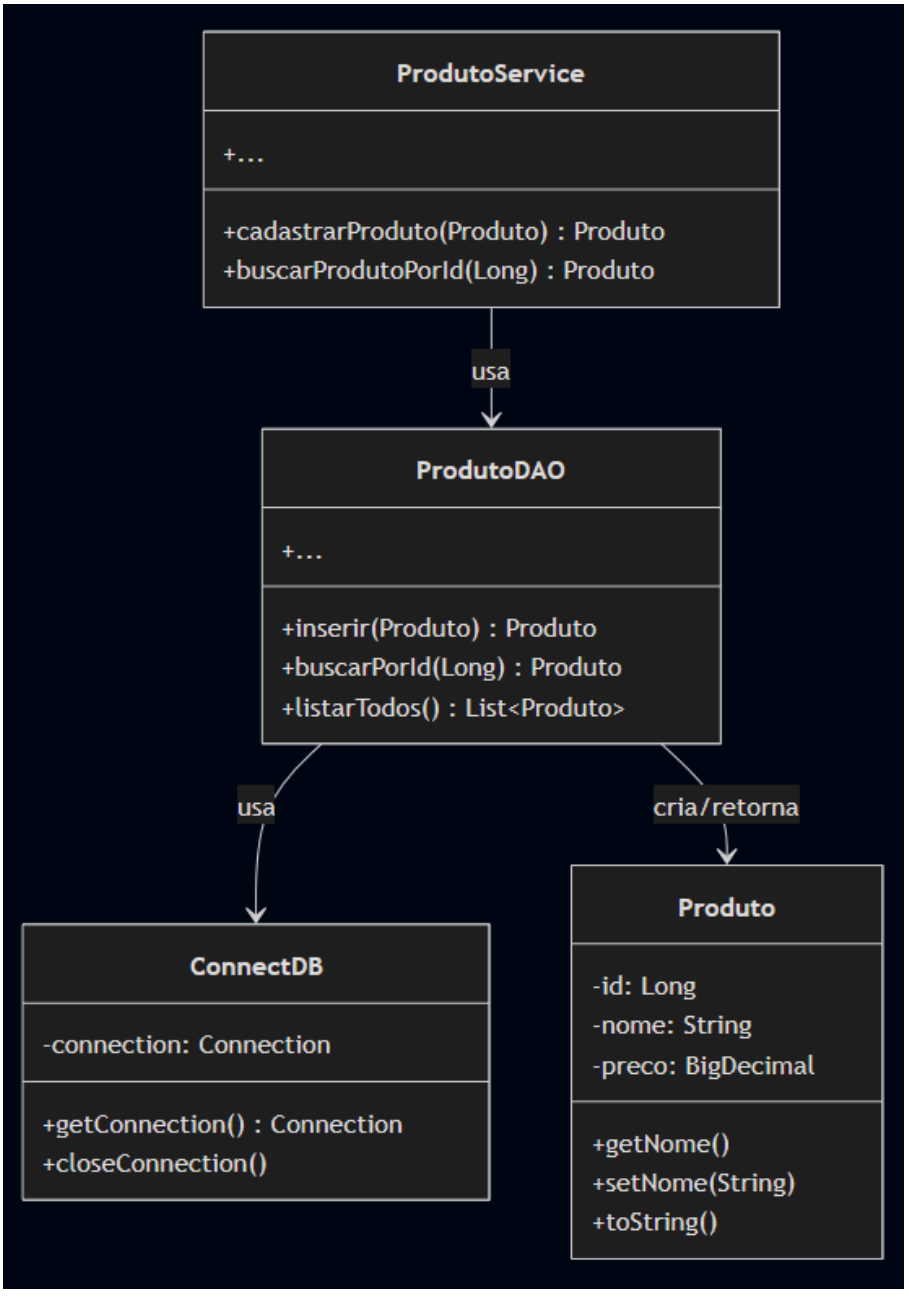

Diagrama ampliado disponível no github.

Exception Handling Pattern
Implementação: Classes ProdutoException , ReservaSinucaException
Justificativa:

- Tratamento Específico: Exceções por domínio
- Mensagens Claras: Erros de negócio vs erros técnicos
- Rastreabilidade: Facilita debugging

Diagrama:

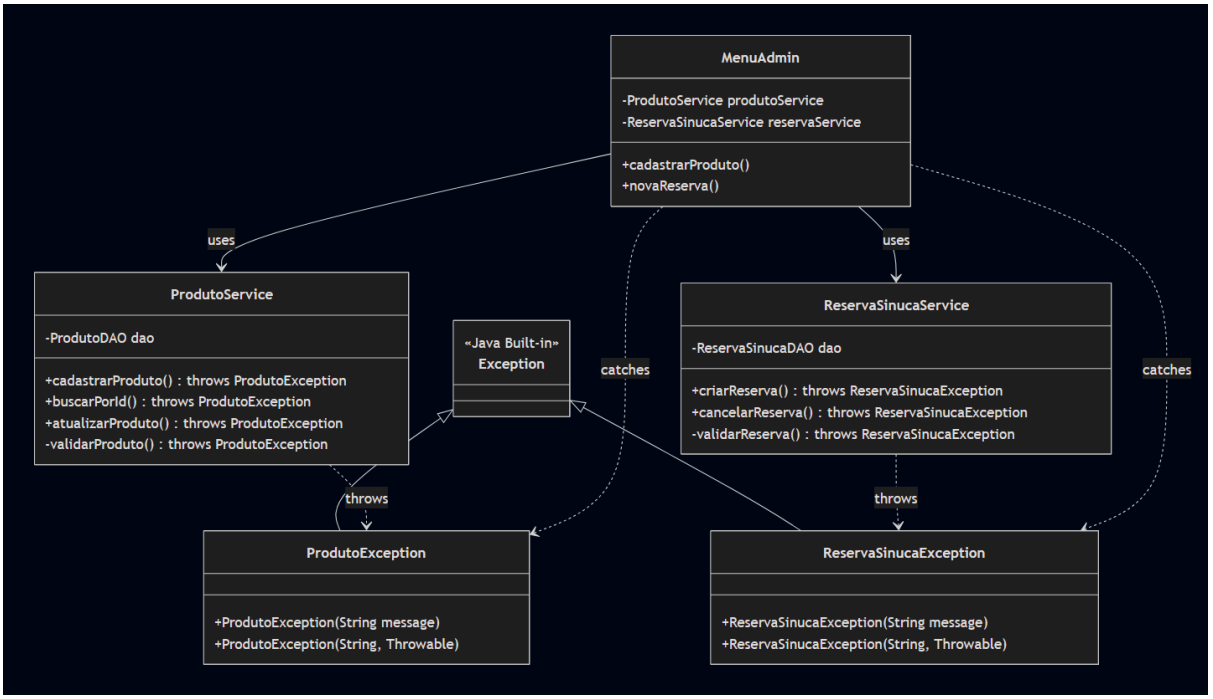

Diagrama ampliado disponível no github.

4 - Gerenciamento de Qualidade

A divisão de tarefas foi feita da seguinte forma: Após definir os atributos dos models, cujo desenvolvimento ficou sob responsabilidade do Thiago, simultaneamente, os outros dois membros trabalhavam nos menus de admin e cliente separadamente, além do menu principal para começar os testes. Após a conclusão dos bancos de dados, o Paulo começou o desenvolvimento dos services necessários, o João criou os DAOs, enquanto o Thiago fez as exceptions.

A linguagem utilizada foi Java, por meio da IDE IntellliJ. Foram tomadas precauções como impedir que produtos tenham preço negativo, unicidade de códigos de produto, padronização de horários de reserva (30min ou 1h), verificação de horário de funcionamento para reservas e verificação de conflitos de horário.

5 - Demonstração do App

Link: Demonstração.mp4