**Sistema de Análise de Dados de Varejo Multimodelo**

**Objetivo:**

Devemos fazer um Sistema de Análise de Dados de Varejo Multimodelo(utilizando vários tipos de banco de dados). O sistema deve permitir análises históricas, tendências e previsões de vendas.

**Datasources:**

ObjectDB (BDOO) - Gerenciar produtos e suas características.  
Postgres (BDR) - Implementar um Data Warehouse com esquema estrela/floco de neve para histórico de vendas.  
MongoDB(NoSql) - Armazenar dados não estruturados (comentários de clientes, imagens de produtos).  
Postgres (BDR) - Implementar aspectos temporais para análise histórica de preços e estoque.

**Descrição do Domínio do Negócio:**

Domínio: Locadora de Filmes

O sistema gerencia o catálogo de filmes, o histórico de locações, a base de clientes, os comentários dos usuários, a disponibilidade dos filmes em estoque e a análise de desempenho das locações ao longo do tempo.

Entidades principais: Cliente, Filme, Locação, Estoque, Comentário/Avaliação

**Arquitetura:**

Estilo de arquitetura: Cliente-Servidor(Client-Server):

* **Frontend** (cliente) - a aplicação web em JavaScript (HTML/CSS/JS) – consumirá serviços fornecidos por um servidor, usando Fetch API.
* **Backend** (servidor) - a API REST em Java com Spring Boot – fornecerá dados, persistência e lógica de negócio, será hospedada em alguma plataforma com plano grátis.

Comunicação: REST (Representional State Transfer)

* A API segue o padrão REST, usando HTTP (GET, POST, PUT, DELETE) para manipular recursos.
* O frontend usa fetch() para chamar endpoints

Separação de responsabilidades:

* O frontend cuida da interface com o usuário, tratando da apresentação do Dashboard para visualização de KPIs(Key Performance Indicators) de vendas, Relatórios analíticos baseados em diferentes dimensões e Interface para exploração de dados via consultas OLAP (drill-down, roll-up).
* O backend cuida da lógica, persistência e regras de negócio, fazendo a integração entre os bancos de dados e realizando a análise dos dados(OLAP, algoritmos de data mining e consultas complexas) enquanto fornece os resultados através de endpoints.
* Comunicação acontece via JSON, sem acoplamento direto.

**Escopo Detalhado:**

### Funcionalidades do Sistema:

1. **Gerenciamento de Produtos e suas Características (ObjectDB)**

* Cadastro e manutenção de filmes, com seus atributos (título, descrição, ano de lançamento, duração, categoria, etc.).
* Gerenciamento de clientes (nome, e-mail, telefone, endereco).
* Registro de locações (cliente, filmes alugados, datas de locação e devolução, valor).

2. **Armazenamento de Dados Não Estruturados (MongoDB)**

* Armazenamento de avaliações e comentários feitos por clientes sobre os filmes alugados.
* Armazenamento de imagens associadas aos filmes.
* Estrutura de documentos com campos como: cliente, filme, nota, comentário, data.

3. **Data Warehouse (PostgreSQL - Modelo Estrela ou Floco de Neve)**

* Construção de um **Data Warehouse** para análise histórica de locações.
* Modelagem em estrela com tabela fato (FatoLocacao) e dimensões (DimFilme, DimCliente, DimTempo, DimLoja).
* Suporte a consultas analíticas com agregações por tempo, gênero, popularidade de filmes, atraso médio, entre outros.

4. **Aspectos Temporais (PostgreSQL)**

* Armazenamento e consulta do histórico de **preços de locação** e **estoque** por período de tempo.
* Implementação de entidades com validade temporal (ex: HistoricoPreco, HistoricoEstoque) contendo data\_inicio e data\_fim.
* Suporte a consultas como: "Qual era o preço da locação do filme X em março de 2023?".

5. **API REST (Spring Boot)**

* Fornecimento de endpoints REST para manipulação e consulta de dados filtrados, processados ou consultas temporais.
* Integração com os diferentes bancos de dados de acordo com o tipo de dado.
* Implementação de lógica de negócio para análise de KPIs (filmes mais alugados, média de atraso, avaliações médias etc.).
* Módulo OLAP: análise multidimensional de 3 dimensões (ex: Tempo, Localização, Categoria de Produto) e com funcionalidades como drill-down, roll-up, slice e dice.
* Data Mining (provavelmente usando biblioteca Weka ou Smile): Utilização de 2 técnicas de Data Mining (ex: clustering de clientes, previsão de vendas)

6. **Frontend Web (HTML/CSS/JavaScript)**

* Interface web para:
  + Visualização de KPIs (indicadores de desempenho).
  + Navegação por relatórios de locação, clientes e filmes.
  + Exploração de dados analíticos com funcionalidades OLAP (drill-down, roll-up, slice, dice).
* Consumo da API REST usando fetch().

### Tecnologias Envolvidas:

* **Java 17 + Spring Boot** (API Backend)
* **ObjectDB 2.9.3** (modelo objeto)
* **PostgreSQL** (data warehouse e histórico temporal)
* **MongoDB** (dados não estruturados)
* **HTML, CSS, JS puro (Fetch API)** (frontend)
* **Maven** para gerenciamento de dependências
* **Plataforma gratuita para deploy** da API (Render, Railway, Vercel etc.)

**Modelagem conceitual dos bancos de dados:**

1. **ObjectDB (Object-Relational)**

**Propósito:** Gerenciar produtos e suas características (neste caso, os filmes).

* **Filme**

id : Long

title : String

descrição : String

anoLançamento : int

categoria : String

duração : int

precoAtual : BigDecimal

1. **PostgresSQL (Data Warehouse + Dados Temporais)**

**Propósito:** Armazenar histórico de vendas e dados temporais.

**Esquema Estrela (Fato + Dimensões):**

* **Tabela Fato: fato\_locacao**

id : BIGINT (PK)

id\_tempo : FK -> dim\_tempo

id\_filme : FK -> dim\_filme

id\_cliente : FK -> dim\_cliente

id\_loja : FK -> dim\_loja

valor\_pago : NUMERIC

quantidade : INTEGER

* **Tabela Dimensão: dim\_tempo**

id : SERIAL

ano : INTEGER

mês : INTEGER

dia : INTEGER

data : DATE

* **Tabela Dimensão: dim\_filme**

id : SERIAL

titulo : TEXT

categoria : TEXT

duracao : INTEGER

* **Tabela Dimensão: dim\_cliente**

id : SERIAL

nome : TEXT

email: TEXT

cidade : TEXT

estado : TEXT

* **Tabela Dimensão: dim\_loja**

id : SERIAL

nome : TEXT

cidade : TEXT

estado : TEXT

**Tabelas Temporais:**

* **estoque\_temporal**

id : SERIAL

id\_filme : FK

quantidade : INTEGER

inicio\_vigencia : DATE

film\_vigencia : DATE

* **preco\_temporal**

id : SERIAL

id\_filme : FK

preco : NUMERIC

inicio\_vigencia : DATE

fim\_vigencia : DATE

1. **MongoBD (NoSQL – Dados Não Estruturados)**

**Propósito:** Armazenar comentários/avaliações e imagens de filmes.

* **Documento: comentário**

{

"\_id": ObjectId,

"filme\_id": "123",

"cliente\_id": "456",

"comentario": "Filme muito bom!",

"avaliacao": 4.5,

"data\_comentario": "2024-05-01T00:00:00Z"

}

* **Documento: imagem\_filme**

{

"\_id": ObjectId,

"filme\_id": "123",

"url\_imagem": "https://vitorsrepository/imagem123.jpg",

"descricao": "Capa oficial do filme"

}