Faculdade

BOOTCAMP ARQUITETURA DE DADOS
DESAFIO FINAL

DOCUMENTO ARQUITETURAL

AUTOR: LUCAS VIEIRA MARTINS

Agosto de 2025

CONTEÚDO

DESCRIÇÃO DO SISTEMA

CONTEXTO

A loja Amazonas, conhecida por sua ampla variedade de produtos, que vão desde eletrônicos até vestuário, utensílios domésticos e livros, está expandindo sua atuação para o mercado digital com o objetivo estratégico de se tornar o maior e-commerce do Brasil, atendendo “de A a Z”.

Para isso, a empresa necessita de um e-commerce robusto, capaz de lidar com o rápido crescimento exponencial de clientes e transações, garantindo alta performance, disponibilidade contínua e escalabilidade, e se possível, recursos de elasticidade no uso de hardware, ajustando-se dinamicamente à demanda do mercado.

A solução proposta está fundamentada em uma arquitetura de dados não-relacional, escolhida por sua flexibilidade e capacidade de escalar horizontalmente, eliminando gargalos comuns em arquiteturas relacionais. Essa abordagem possibilita maior velocidade no processamento de grandes volumes de dados, redução da necessidade de joins complexos e facilidade na modelagem de coleções otimizadas para operações de leitura e escrita em larga escala.

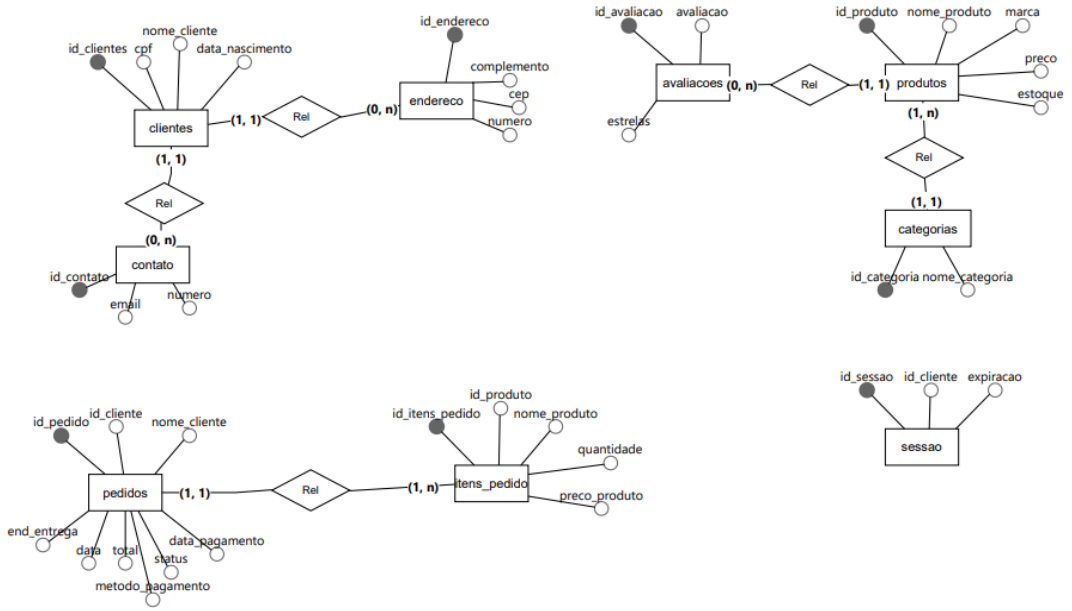
OBJETIVOS PRINCIPAIS

O objetivo deste projeto é desenvolver a modelagem de dados não-relacional para o e-commerce da loja Amazonas, garantindo uma estrutura robusta, escalável, flexível e tolerante a falhas, capaz de suportar o crescimento exponencial de clientes e transações. Para isso, serão projetadas coleções no estilo de bancos NoSQL que representem os principais aspectos do sistema, como clientes, produtos, pedidos, carrinhos e avaliações, utilizando práticas de desnormalização que evitem o uso de joins aproveitando o máximo de desempenho do NoSQL e promovam alta performance. Além disso, serão elaborados modelos de dados, contemplando chaves, campos relevantes, coleções e, se preciso, relacionamentos, de forma a apoiar a construção de uma aplicação moderna, eficiente e preparada para as demandas do mercado digital.

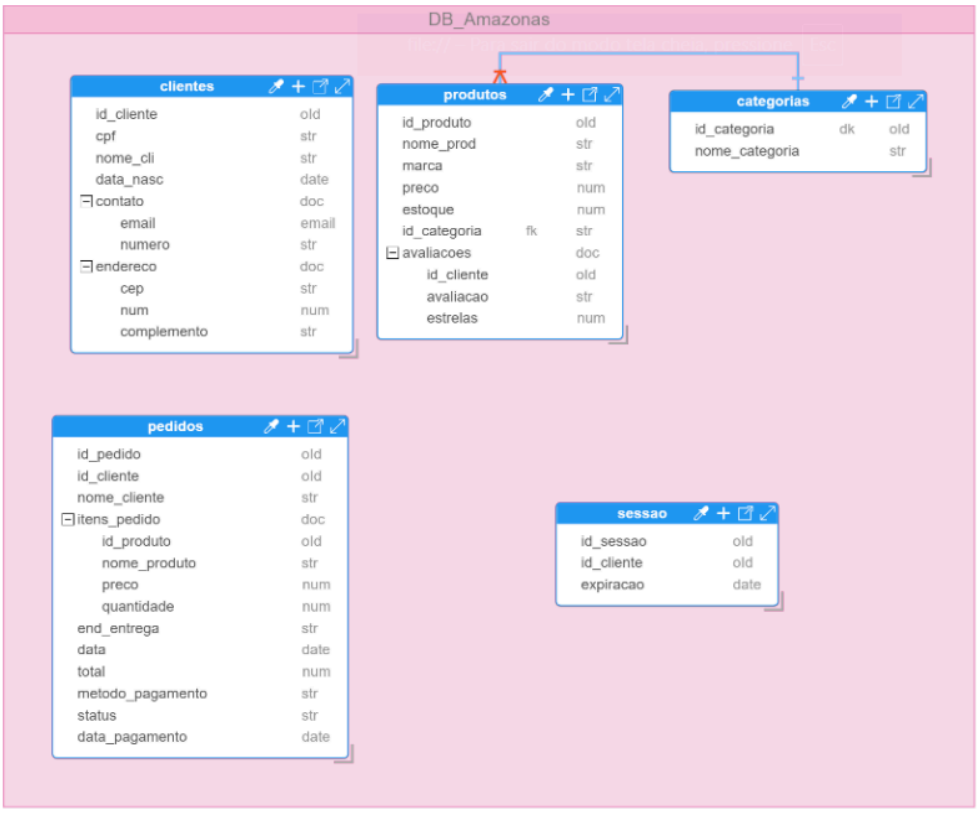
ESTRUTURA DE DADOS PROPOSTA

Os modelos conceitual e lógico foram feitos usando respectivamente as ferramentas BRModelo Web e Hackolade Community Edition.

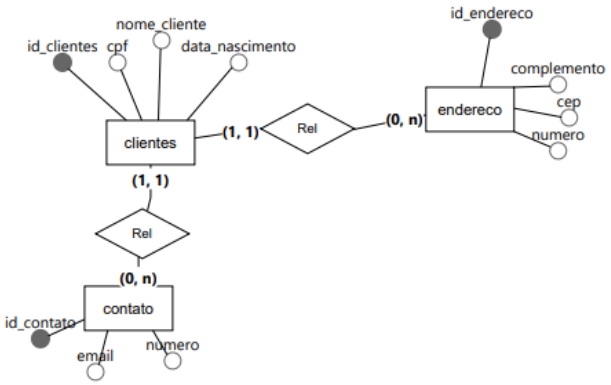
MODELO CONCEITUAL

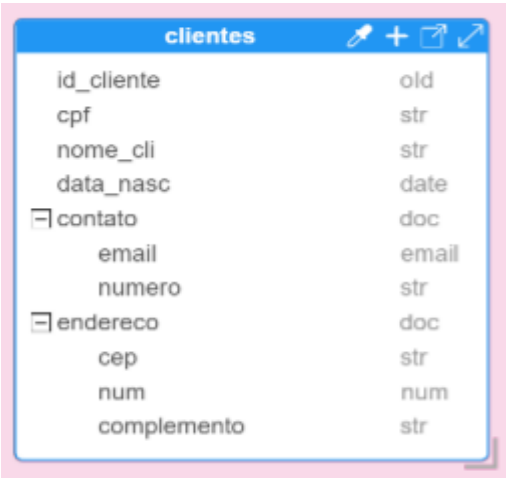


MODELO LÓGICO



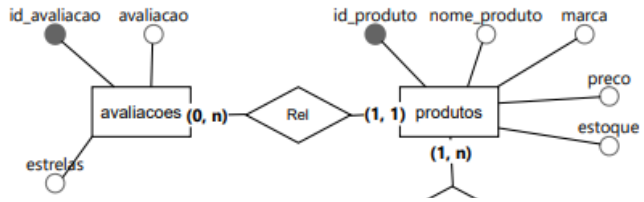
COLEÇÃO CLIENTES





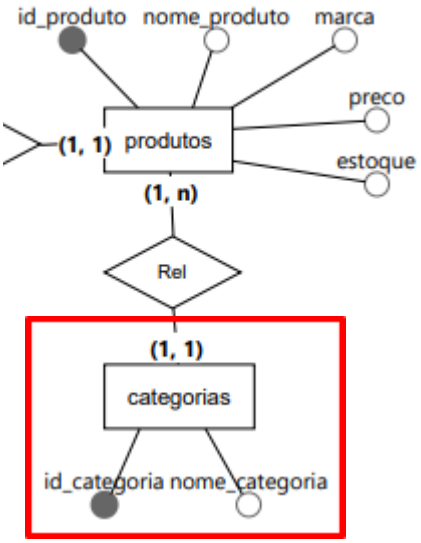
A coleção “clientes” possui um ObjectID como chave primária e o CPF como dado pessoal auxiliando na identificação do cliente. Dados como CPF e senha são dados que requerem um nível maior de segurança como criptografia. A coleção clientes também possui campos como contato, sendo um array que pode conter ao menos um email e um número de contato, mas podendo ter vários emails e números. O campo endereço também é um array de documentos, permitindo o cliente ter vários, ou pelo menos um endereço.

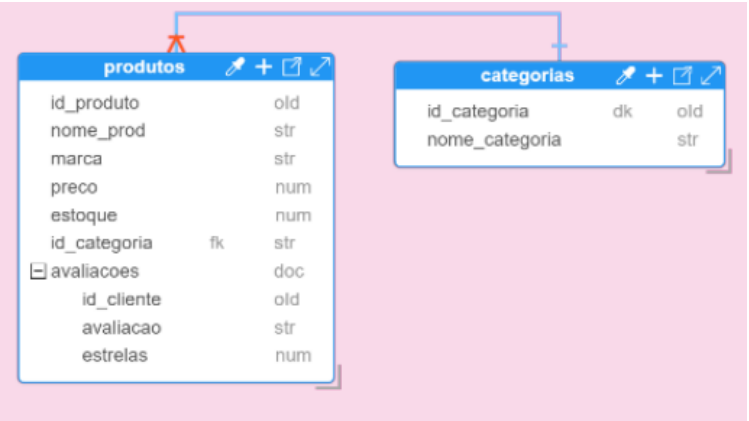
COLEÇÃO PRODUTOS



A coleção “produtos” possui um ObjectID como identificador, e os campos nome, marca, preço, estoque. Também possui o campo “id_categoria” para, se necessário, fazer referência à categoria (departamento) na qual o produto pertence. O produto possui um campo “avaliações”, que é um *array* de documentos, podendo ter nenhuma ou várias avaliações de pessoas que compraram o produto.

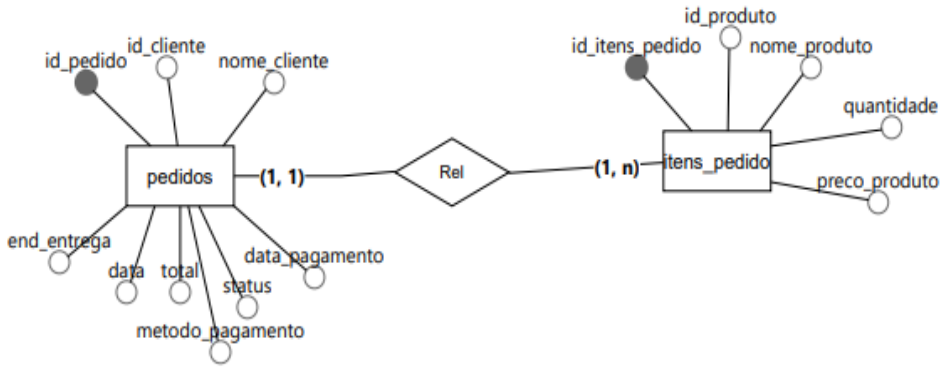
COLEÇÃO CATEGORIAS





A coleção categorias é referenciada em produtos com seu identificador apenas para consulta de qual categoria pertence tal produto.

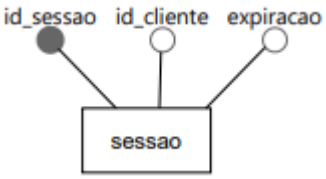
COLEÇÃO PEDIDOS

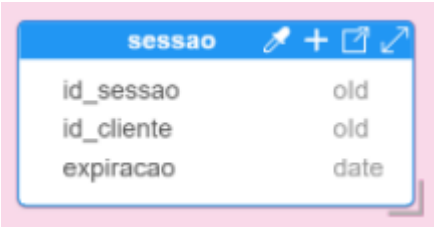


A coleção pedidos possui o seu identificador, um ObjectID, e também possui o identificador e o nome do cliente para identificar com poucas informações quem fez o pedido. A coleção tem um array de documentos, “itens_pedido” contendo as principais informações, assim como do cliente, dos itens que estão sendo comprados, como o *id* do produto, o nome, o preço e a quantidade. A coleção ainda apresenta os campos para endereço de entrega, data da compra, total (sendo um campo derivado do cálculo da soma do preço de cada produto multiplicado pelas suas quantidades), o método de

pagamento, o status da compra (podendo ser uma compra em andamento (carrinho), aguardando pagamento, em processo de entrega ou já finalizada (histórico de compras), podendo ter uma coleção só para o histórico contendo todas as compras já finalizadas) e por último a data da confirmação do pagamento.

COLEÇÃO SESSÃO





A coleção sessão guarda informações sobre a sessão do cliente, ela possui os campos de id da sessão, id do cliente e o tempo para expiração da sessão.

PLANO DE ESCALABILIDADE: PERFORMANCE E DISPONIBILIDADE

Como plano de escalabilidade, o sistema será feito usando o banco de dados MongoDB, usando a plataforma em nuvem da própria Mongo, a MongoDB Atlas.

Será utilizado um *cluster* com provisionamento na infraestrutura de nuvem da AWS, na região de São Paulo, com *auto-scale* e *cloud backup*. O *auto-scale* aumenta o cluster quando 90% do uso de disco é alcançado e para ter elasticidade, também analisa o uso de CPU e memória para fazer o *scale up* ou *scale down*. Já o *cloud backup* mantém backup contínuo automático do banco para restaurá-lo a qualquer ponto que for preciso.

Para o sistema ter tolerância a falhas, garantindo a alta disponibilidade, cada coleção terá réplicas em três nós diferentes, com localização física em *datacenters* diferentes, todas as coleções terão réplicas.

Algumas coleções ainda teriam também particionamento (*shards*) para alta performance, em três *shards* diferentes.

Seriam essas coleções:

- **Clientes**: A coleção de clientes tende a crescer de forma massiva, abrangendo usuários de diferentes regiões. O particionamento em *shards* permite distribuir esses dados geograficamente, equilibrando a carga de leitura e escrita e reduzindo a latência de acesso em consultas.

- **Produtos**: O catálogo da loja Amazonas é bastante amplo e dinâmico, com vários itens de diferentes categorias. O uso de *sharding* garante que o sistema suporte buscas rápidas e filtragens em grande escala, além de permitir que atualizações e inserções de novos produtos sejam distribuídas de forma eficiente entre os nós.

- **Pedidos**: A coleção de pedidos é a mais crítica, pois apresenta crescimento exponencial e altíssima taxa de escrita devido às constantes transações. Particionar em *shards* garante escalabilidade horizontal, possibilitando lidar com picos de acessos durante promoções, liquidações e datas sazonais (ex.: Black Friday), sem comprometer a performance.

As coleções restantes não necessitam de particionamento, por isso estão sem o esquema de *shards*.

DIAGRAMA DA INFRAESTRUTURA DE RÉPLICA E SHARDING