Lojas Americanas

2023.2_G5_ProjetoAmericanas
Software Architecture Document

Version 1.2

Revision History

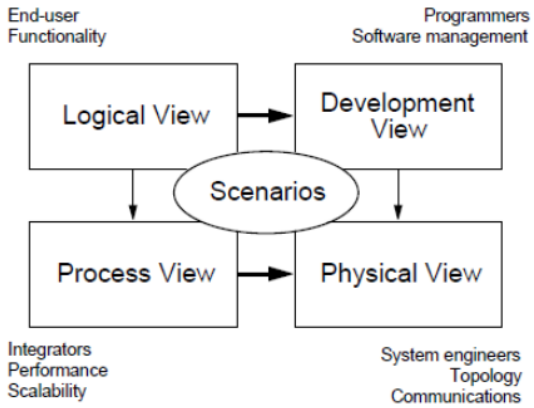
Date	Version	Description	Author
01/12/2023	1.0	Criação de documento	Pedro Vitor Augusto de Jesus Fellipe Pereira da Costa Silva
01/12/2023	1.1	Estruturação de introdução	Pedro Vitor Augusto de Jesus Fellipe Pereira da Costa Silva
01/12/2023	1.2	Inclui Architectural Representation, Architectural Goals and Constraints, Use-Case View, Use-Case Realizations, Logical View, Overview, Architecturally Significant Design Packages, Process View, Deployment View, Implementation View, Overview, Layers, Data View (optional), Quality.	Pedro Vitor Augusto de Jesus Fellipe Pereira da Costa Silva José Luís Ramos Teixeira Silas Neres de Souzar Pablo Christianno Silva Guedes Philipe de Sousa Barros

Table of Contents

Documento de Arquitetura de Software

1. Introdução

O Documento de Arquitetura de Software proporciona uma visão completa do panorama específico da Americanas, abrangendo seu propósito, escopo, definições, siglas, abreviações, referências e uma síntese abrangente do conteúdo contido nele. Essencialmente, busca oferecer uma compreensão global e detalhada da arquitetura do software, destacando as decisões cruciais tomadas e explorando várias perspectivas arquitetônicas para esclarecer os diversos elementos do sistema da Americanas em análise.



1.1 Purpose

Este documento oferece uma análise detalhada da arquitetura do sistema, utilizando múltiplas perspectivas arquitetônicas para abordar distintos aspectos do sistema. Seu propósito central é capturar e comunicar as decisões arquitetônicas de maior relevância adotadas no desenvolvimento do sistema.

1.2 Scope

Este Documento de Arquitetura de Software aborda a arquitetura do aplicativo de Avaliação de Produtos das Americanas. Nele estão contemplados a estrutura global do sistema, os principais componentes, interfaces, fluxos de informação e as interações entre os diversos módulos do software das Americanas.

1.3 Definições

*[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.4 Overview

Este documento será dividido em diferentes seções que abordam diversas visões que ajudam a compreender aspectos específicos do sistema.

Visões Lógicas:

Diagrama de Classes: Oferece uma representação visual das classes do sistema, seus atributos, métodos e relacionamentos, permitindo uma compreensão da estrutura estática do sistema.

Diagrama de Sequência: Mostra a interação entre objetos ao longo do tempo, destacando a ordem das mensagens trocadas entre eles em um cenário específico, útil para compreender o fluxo de execução.

Diagrama de Colaboração: Similar ao diagrama de sequência, porém com foco na estrutura de colaboração entre objetos, exibindo como os objetos se conectam e interagem entre si.

Diagrama de Pacotes: Apresenta a organização do sistema em módulos ou pacotes, mostrando a estrutura de agrupamento de classes e seus relacionamentos.

Visão de Implementação:

Diagrama de Componentes: Detalha os componentes físicos do sistema, como bibliotecas, módulos, frameworks, e como eles se relacionam para formar a estrutura do software.

Visão de Processo:

Diagrama de Sequência: Neste contexto, pode ser utilizado para representar a interação entre processos ou threads, mostrando a ordem das operações em um cenário de execução.

Diagrama de Atividades: Oferece uma visão de alto nível do fluxo de trabalho do sistema, mostrando as atividades e decisões realizadas ao longo do processo.

Visão de Implantação:

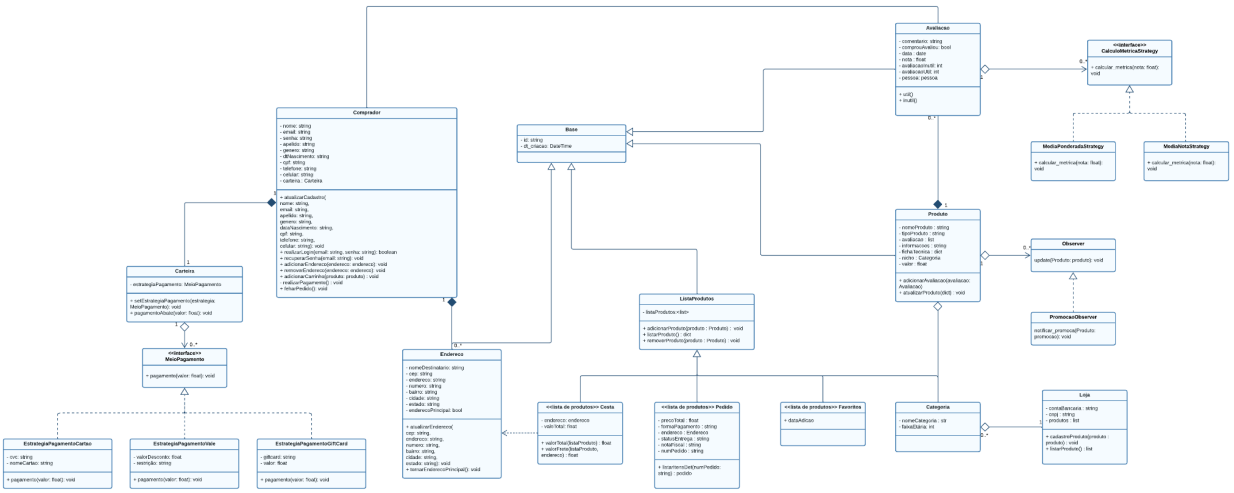
Diagrama de Implantação: Apresenta a disposição física dos elementos do sistema, incluindo hardware, software, nós de rede, mostrando como estes são implantados e interligados na infraestrutura.
Cada uma dessas visões fornece uma perspectiva única do sistema, permitindo que diferentes partes interessadas compreendam aspectos específicos do software, desde sua estrutura até sua implementação e execução no ambiente de produção.

2. Representação Arquitetural

Lista de artefatos construídos durantes o semestre:

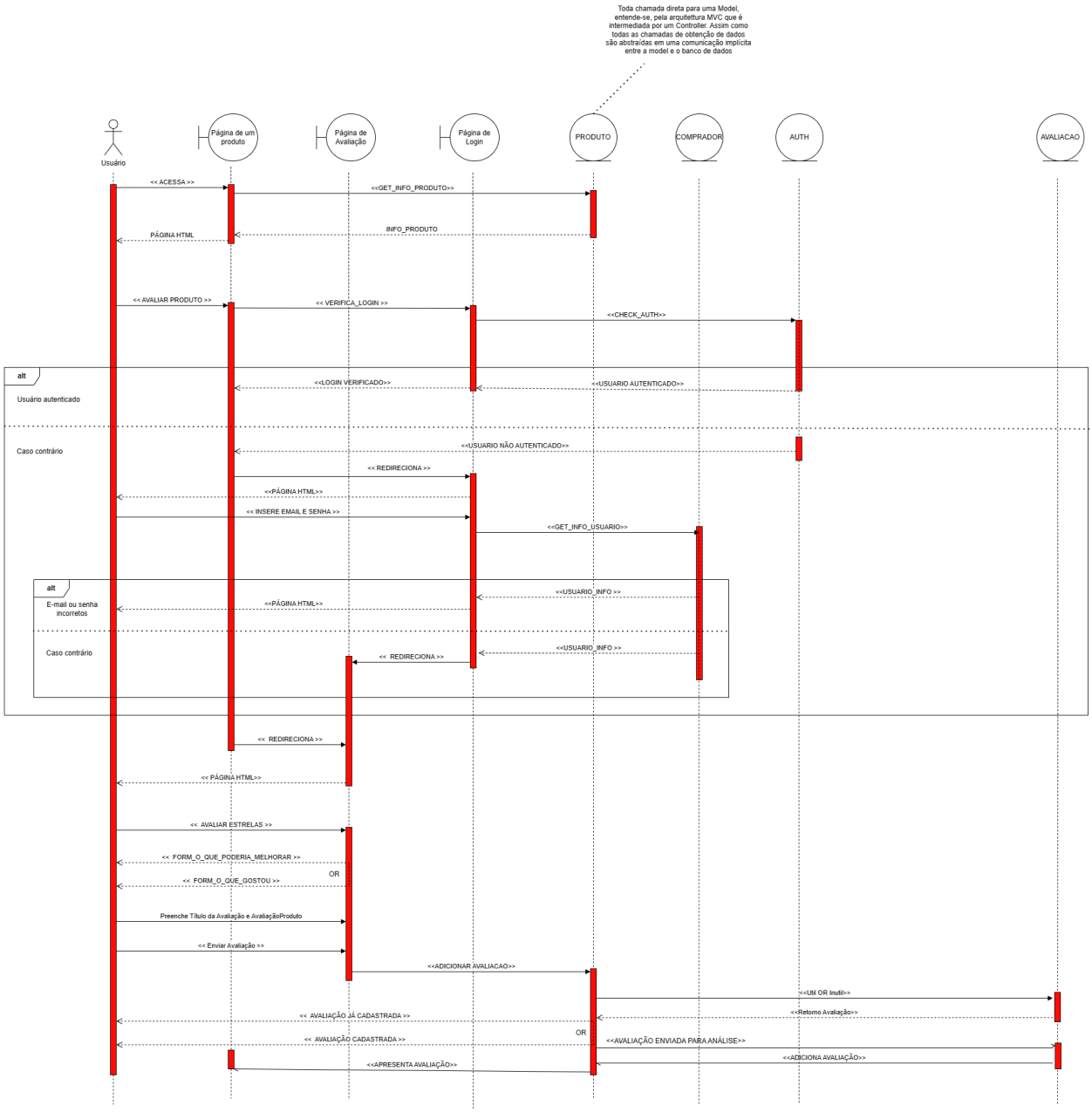
2.1 Diagrama de Classes

Um Diagrama de Pacotes é um diagrama estrutural definido pela UML que descreve os pacotes ou pedaços do sistema divididos em agrupamentos lógicos mostrando as dependências entre eles. Este diagrama é muito utilizado para ilustrar a arquitetura de um sistema mostrando o agrupamento de suas classes.



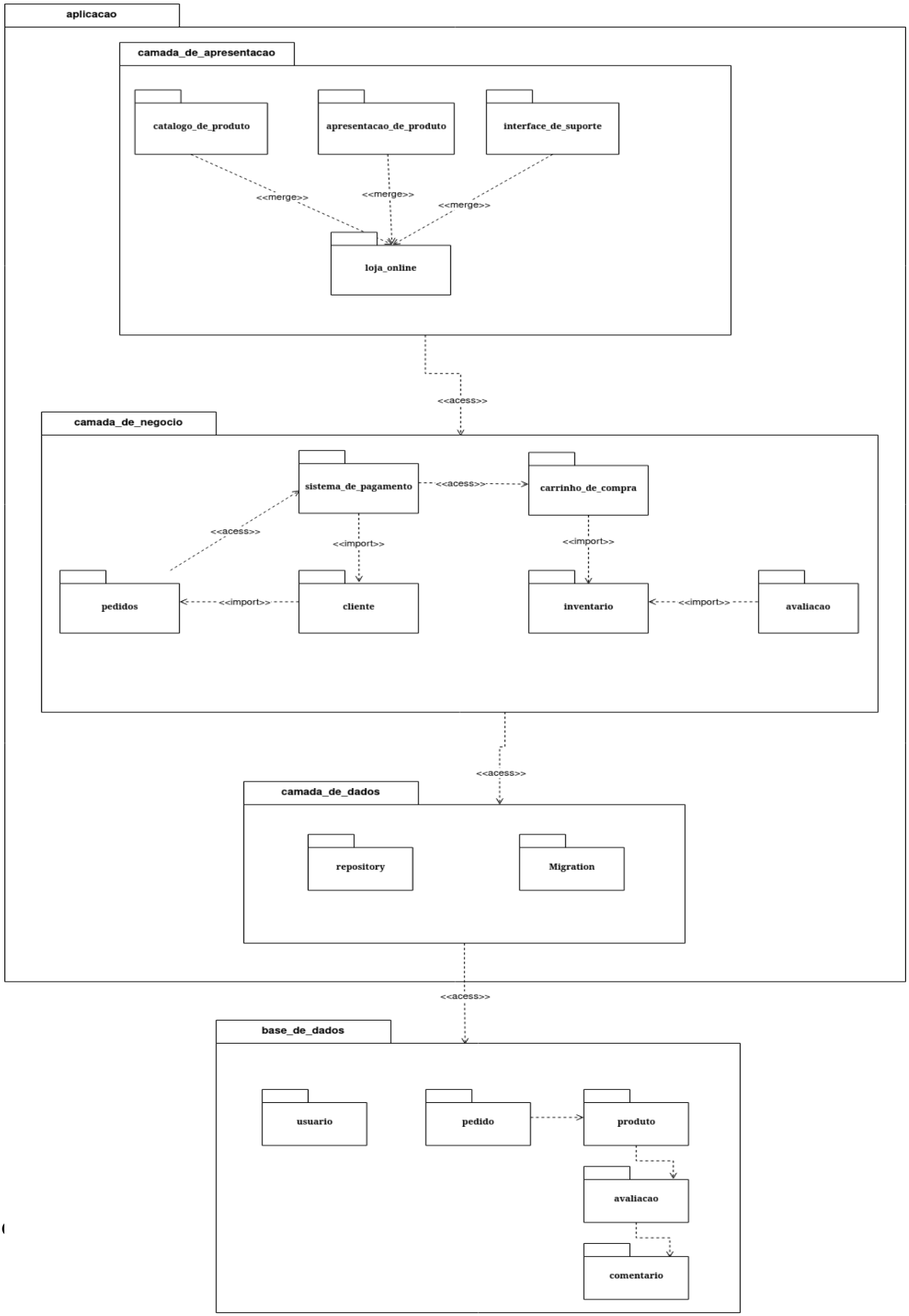
2.2 Diagrama de Sequência

Foi desenvolvido um diagrama de sequência que descreve o fluxo de avaliação de produtos no site das Americanas, utilizando a ferramenta Lucidchart. Isso envolveu a análise de requisitos, a identificação de atores como usuário e sistema, e a compreensão dos eventos relacionados à avaliação de produtos. Em seguida, foi criado o diagrama, mapeando visualmente os passos do processo no Lucidchart e definindo objetos como a API, banco de dados de produtos e usuário.



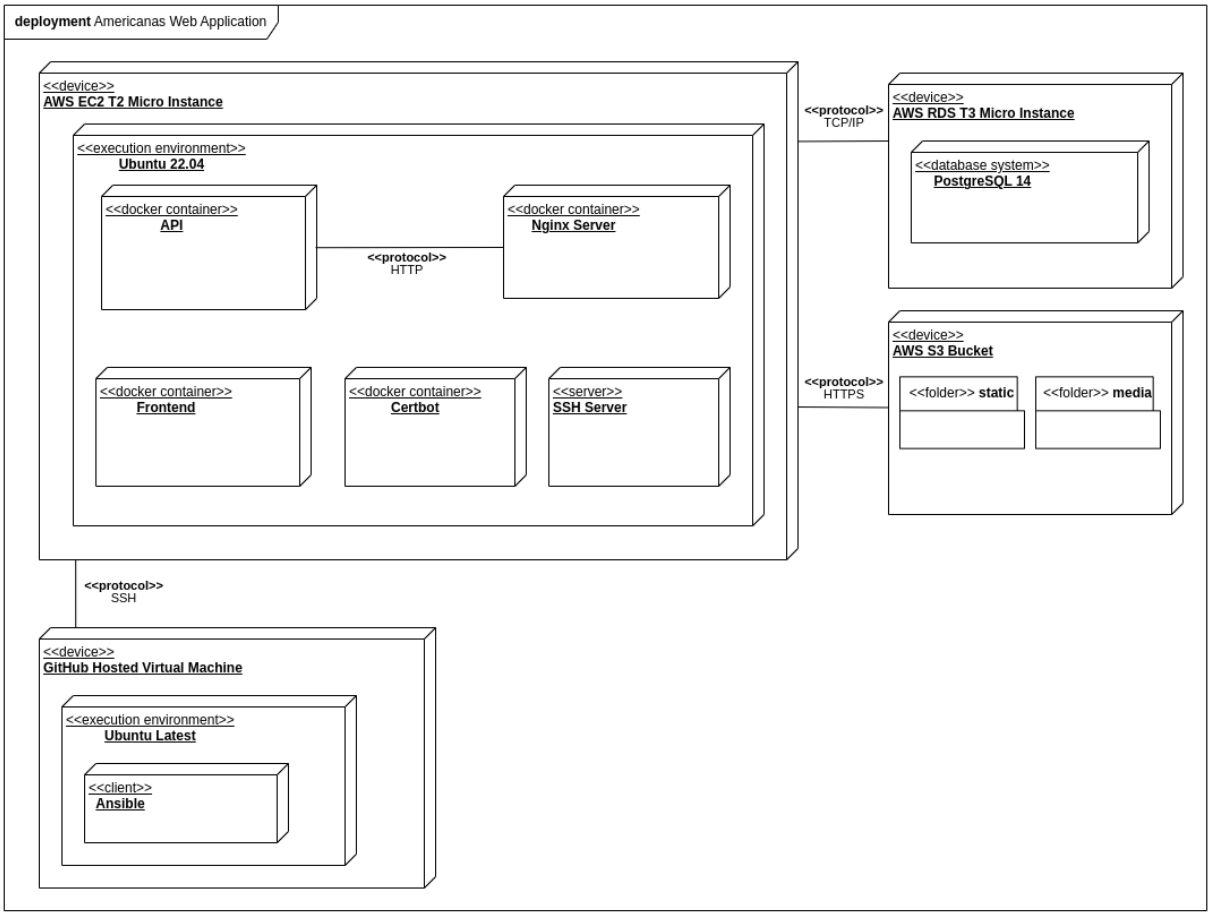
2.3 Diagrama de Pacotes

No processo de desenvolvimento do sistema para a Americanas, foi utilizado a ferramenta Draw.io para criar um diagrama de pacotes para prover uma visualização em escala mais alta do sistema, assim como a estrutura arquitetural e as dependências de cada camada. Os responsáveis pela criação do diagrama primeiramente modelaram individualmente e depois analisaram as duas modelagens e escolheram a que acharam que melhor representava o sistema a nível de pacote. A seguir foram feitos debates sobre a modelagem e aprimoramentos que deveriam ser feitos. A Partir deste diagrama podemos compreender melhor a arquitetura do sistema e o relacionamento de alguns de seus componentes em um nível mais amplo que o visto no diagrama de classes**.**


aplicacao
camada_de_apresentacao
catalogo_de_produto
apresentacao_de_produto
interface_de_suporte
<<merge>>
<<merge>>
<<merge>>
loja_online
<<acess>>
camada_de_negocio
sistema_de_pagamento
<<acess>>
carrinho_de_compra
<<acess>>
<<import>>
<<import>>
pedidos
<<import>>
cliente
inventario
<<import>>
avaliacao
<<acess>>
camada_de_dados
repository
Migration
<<acess>>
base_de_dados
usuario
pedido
produto
avaliacao
comentario

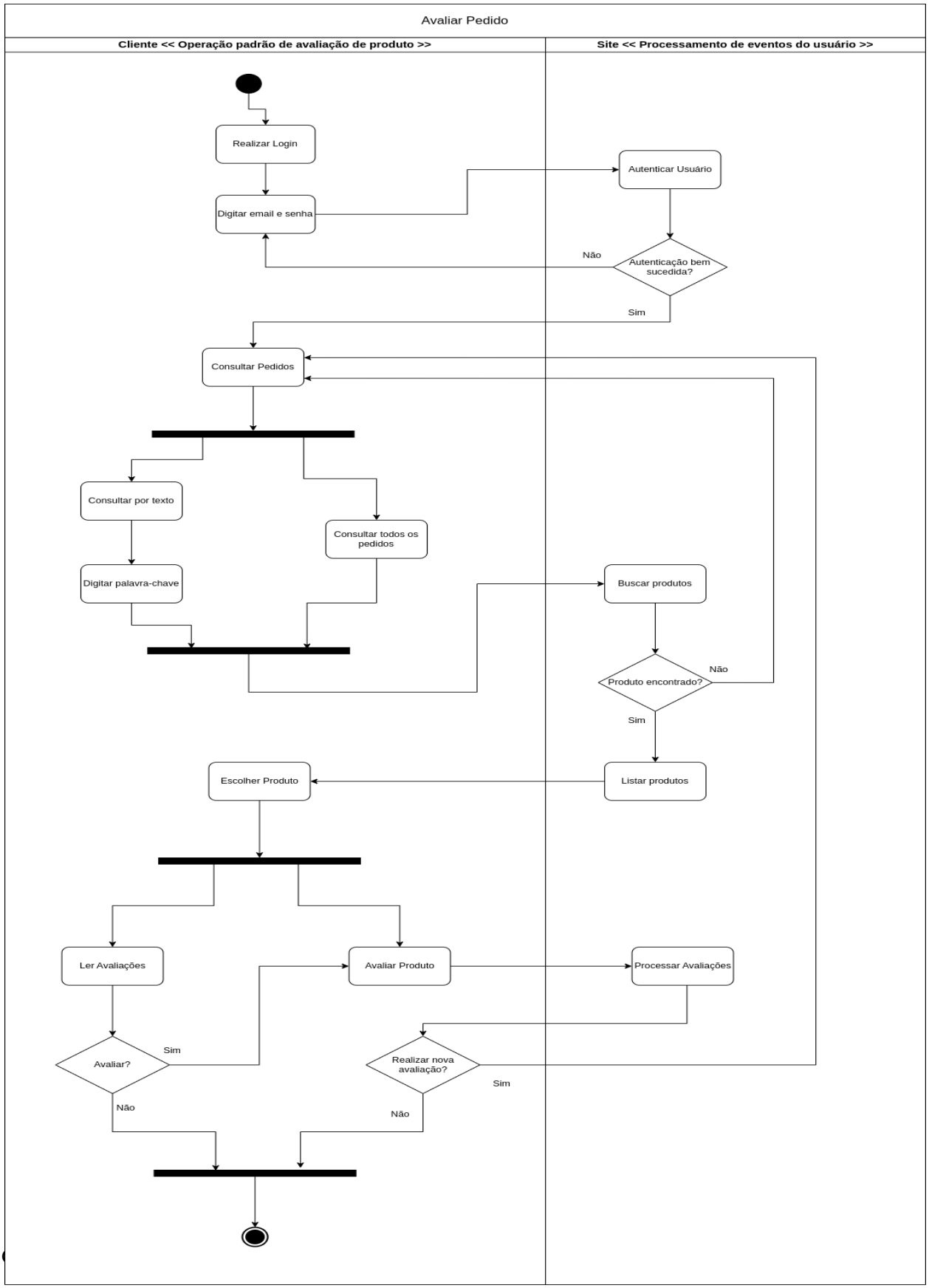
2.4 Diagrama de Componentes

O seguinte diagrama de implantação foi utilizado para fazer o design do sistema deste projeto. Tal design foi feito de acordo com os recursos financeiros disponíveis e com os frameworks escolhidos durante reunião com as pessoas desenvolvedoras. Para o frontend, foi escolhido o React. Para o backend, Django. Para o banco de dados, PostgreSQL. Para a hospedagem, AWS (EC2, RDS, S3). Para a pipeline CI/CD, GitHub Actions e Ansible. Para o proxy reverso, Nginx. Para provisão e renovação de certificados SSL, LetsEncrypt e Certbot. Para virtualizar a aplicação, foi escolhido utilizar containers Docker.



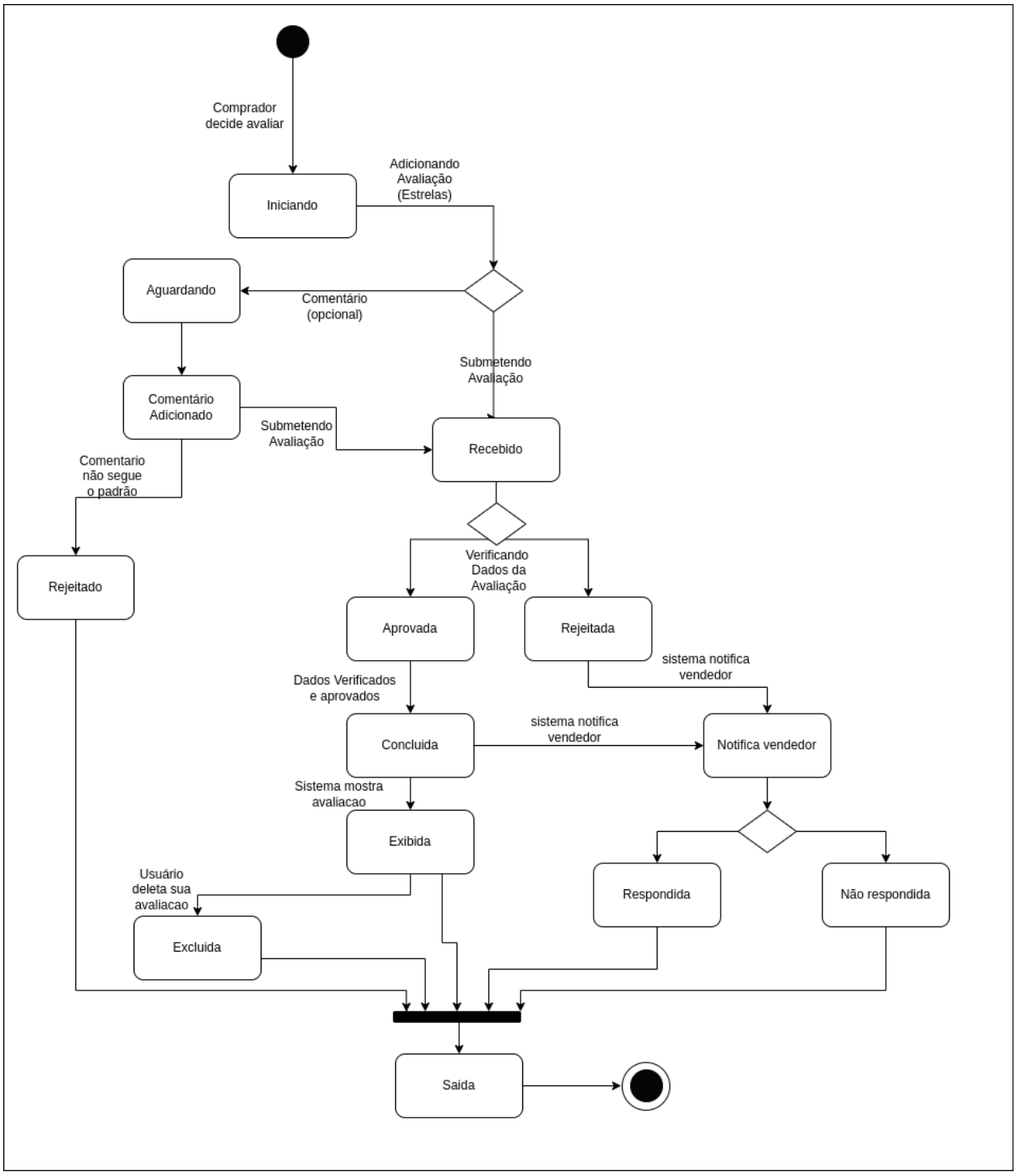
2.5 Diagrama de Atividade

Para a elaboração do diagrama de atividades para o fluxo de avaliação de pedidos da Americanas, foi utilizada a ferramenta draw.io em conjunto com o material disponibilizado na disciplina. O fluxo pode ser dividido em duas unidades organizacionais. A primeira é o cliente, que segue o seu fluxo de avaliação de pedidos. E a segunda é o site da americana, que é responsável pela listagem dos produtos e processamento da avaliação.


Avaliar Pedido
Cliente << Operação padrão de avaliação de produto >>
Site << Processamento de eventos do usuário >>
Realizar Login
Digitar email e senha
Autenticar Usuário
Autenticação bem sucedida?
Não
Sim
Consultar Pedidos
Consultar por texto
Digitar palavra-chave
Consultar todos os pedidos
Buscar produtos
Produto encontrado?
Não
Sim
Listar produtos
Escolher Produto
Ler Avaliações
Avaliar Produto
Processar Avaliações
Avaliar?
Sim
Não
Realizar nova avaliação?
Sim
Não

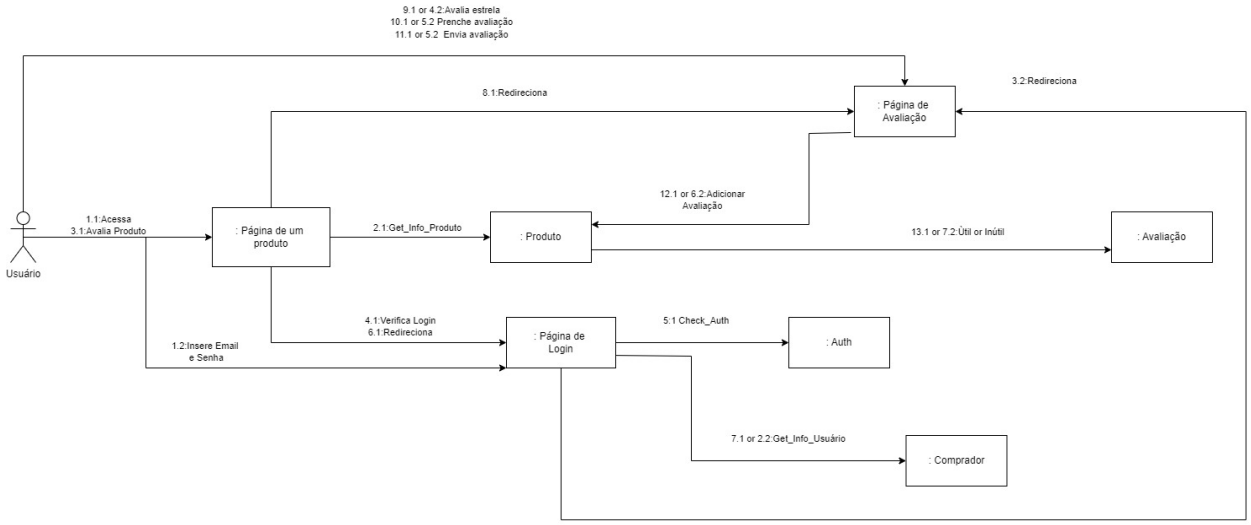
2.6 Diagrama de Estados

O diagrama de estados criado no Draw.io nos permitiu visualizar de forma clara como o Perfil de Comprador interage com a Plataforma e como os diferentes estados desse perfil se relacionam com as avaliações de produtos. Ele também ajudou a definir como o sistema responde a eventos específicos, como a submissão de uma avaliação ou a atualização de informações do perfil do comprador.



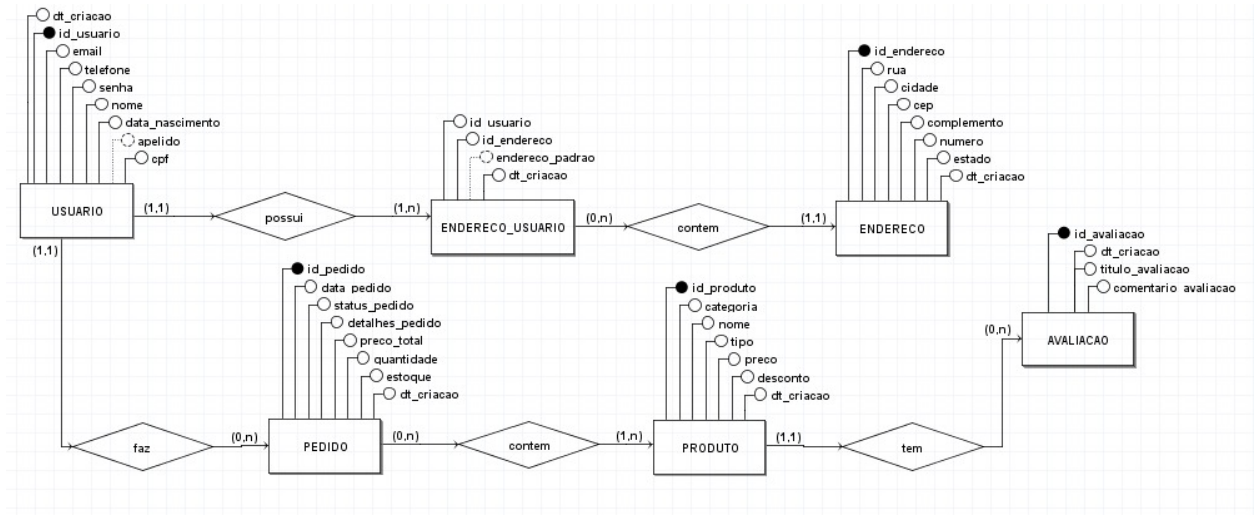
2.7 Diagrama de Comunicação

Foi tomado como base para criação desse diagrama o Diagrama de Sequência, que estava atualizado de acordo com o feedback da Professora. Esse diagrama serve muito bem como base, pois tem uma estrutura similar, sendo sua principal diferença para o Diagrama de Sequência, o enfoque na comunicação, no entanto, foi indicado a sequência de cada ação através de uma enumeração, para facilitar o entendimento e clareza do diagrama



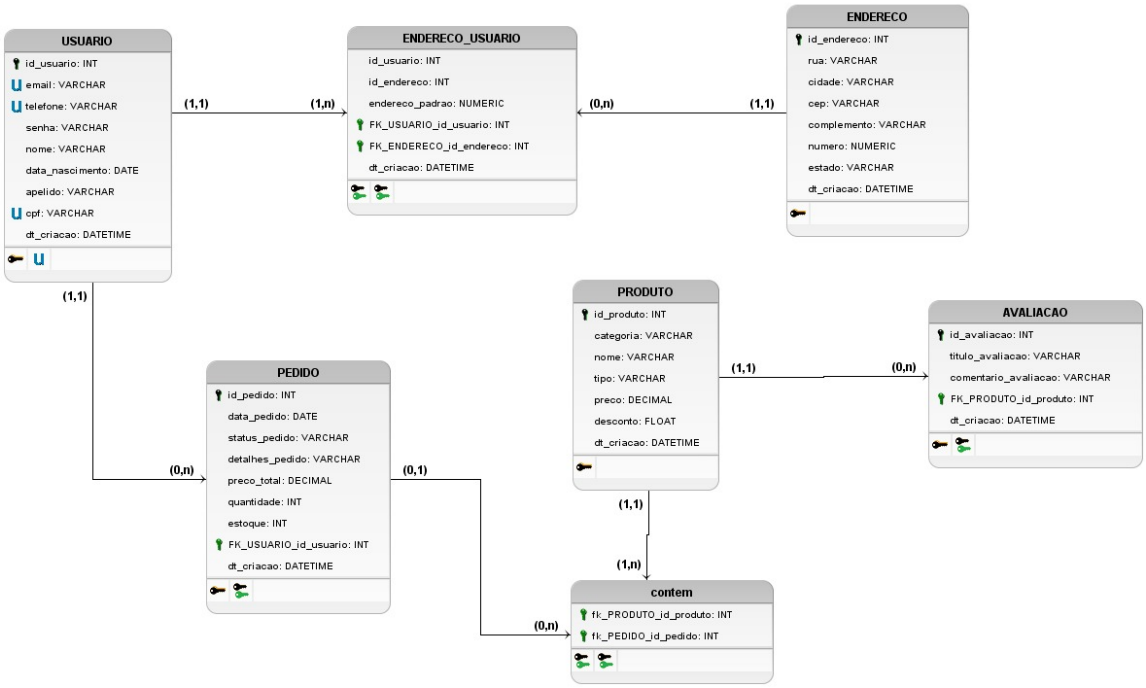
2.8 Diagrama de Entidade-Relacionamento

Um Diagrama Entidade-Relacionamento (DER ou ERD, do inglês Entity-Relationship Diagram) é uma representação visual que descreve as relações entre entidades em um sistema de banco de dados. Esses diagramas são amplamente utilizados durante a fase de modelagem de dados para projetar a estrutura de um banco de dados de forma clara e compreensível.



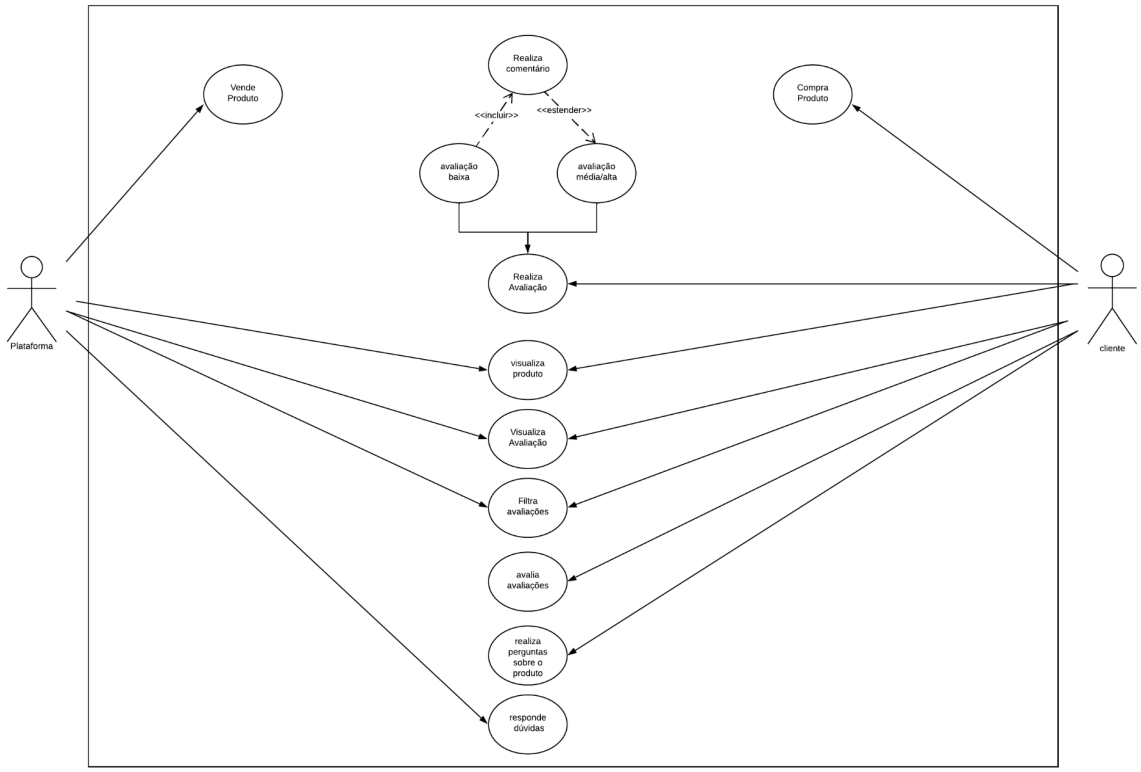
2.9 Diagrama Lógica de Dados

Um diagrama lógico de dados é uma representação visual que descreve como os dados estão organizados e relacionados em um sistema ou aplicação. Ele foca na estrutura lógica dos dados, independentemente da implementação física no banco de dados.



2.10 Diagrama de Caso de Uso

Um modelo de caso de uso é um modelo que descreve como diferentes tipos de usuários interagem com o sistema para resolver um problema. Como tal, ele descreve as metas dos usuários, as interações entre os usuários e o sistema, bem como o comportamento necessário do sistema para satisfazer estas metas. Um modelo de caso de uso consiste em um conjunto de elementos de modelo. Os elementos de modelo mais importantes são: casos de uso, atores e as relações entre eles.



3. Objetivos e Restrições

Os objetivos arquitetônicos delineiam as metas que a estrutura do software deve atingir, como escalabilidade, flexibilidade, modularidade, desempenho, entre outros critérios fundamentais. Estes objetivos estabelecem a direção e os resultados almejados para a arquitetura, servindo como alicerce para as

decisões arquitetônicas. A seguir, encontrará uma breve explicação e possíveis soluções para cada uma das metas mencionadas anteriormente.

Modularidade:

Para assegurar a modularidade do sistema, é crucial adotar práticas e abordagens que favoreçam a divisão de responsabilidades e a estruturação em componentes independentes. Algumas práticas recomendadas incluem o baixo acoplamento: Minimizar as interdependências entre os módulos, evitando fortes vínculos entre eles. A adoção de padrões de projeto, como aqueles empregados em nosso projeto, contribui significativamente para alcançar esse objetivo.

Flexibilidade:

Para aprimorar a flexibilidade do sistema, ou seja, sua capacidade de se adaptar e evoluir com maior facilidade, é fundamental adotar práticas específicas, tais como:

Modularidade: Fragmentar o sistema em módulos independentes e coesos, atribuindo a cada um uma responsabilidade claramente definida. Essa abordagem possibilita realizar alterações em módulos isolados, sem impactar o funcionamento global do sistema.

Padrões de Projeto: A utilização de padrões reconhecidos, como Injeção de Dependência, Strategy, Decorator e Adapter, empregados em nosso projeto, pode potencializar a flexibilidade do sistema. Esses padrões permitem ajustar o comportamento do sistema sem a necessidade de modificar seu núcleo, facilitando sua evolução e manutenção.

4. Visão de Casos de Uso

Um modelo de caso de uso é um modelo que descreve como diferentes tipos de usuários interagem com o sistema para resolver um problema. Como tal, ele descreve as metas dos usuários, as interações entre os usuários e o sistema, bem como o comportamento necessário do sistema para satisfazer estas metas. Um modelo de caso de uso consiste em um conjunto de elementos de modelo. Os elementos de modelo mais importantes são: casos de uso, atores e as relações entre eles.

Ao criar uma visão arquitetural de casos de uso, os arquitetos de software podem visualizar e comunicar de forma clara como o sistema irá atender às necessidades dos usuários, facilitando a tomada de decisões arquiteturais e o alinhamento entre os requisitos e o design do sistema.

4.1 Use-Case Realizations

Atores:

Número	Ator	Descrição
1	Comprador	Usuário comprador de produtos e avaliador no site da Americanas
2	plataforma	Usuário vendedor de produtos, sendo a própria plataforma

Tabela de itens:

Código	Descrição
US01	Vende produto
US02	Compra produto
US03	Visualiza produto
US04	Realiza Avaliação
US05	avaliação baixa
US06	avaliação alta
US07	realiza comentário
US08	visualiza avaliação
US09	filtra avaliações
US10	avalia avaliações
US11	realiza perguntas
US12	responde perguntas

Tabela de dependências:

Relação	Tipo
US05xUS07	INCLUD
US05xUS07	EXTEND

Diagrama de casos de uso

O diagrama está presente no tópico 2.10 desse documento.

5. Logical View

A visão lógica de um documento arquitetural de software descreve a estrutura interna do sistema e como seus diferentes componentes se relacionam entre si de forma lógica. Essa visão tem como objetivo principal fornecer uma representação abstrata do sistema, destacando os principais elementos funcionais e suas interações. É essencial para que os desenvolvedores, arquitetos de software e outros stakeholders compreendam a estrutura e o

funcionamento interno do sistema, facilitando o desenvolvimento, a manutenção e a evolução do software ao longo do tempo.

5.1 Overview

A visão lógica na arquitetura de software se refere a uma perspectiva específica utilizada para compreender a estrutura e o funcionamento de um sistema de software. Essa visão concentra-se nas regras de negócio, nos processos e na lógica subjacente que orientam o comportamento do sistema.

5.2 Architecturally Significant Design Packages

Para o nosso projeto utilizamos a visão lógica nos diagramas.

Diagrama de Classe

No processo de desenvolvimento do sistema para a Americanas, foi utilizando a ferramenta Lucidchart para criar um diagrama de classe UML que representa os principais componentes do sistema, focando no Perfil de Comprador, na Plataforma e nos fluxos relacionados às avaliações de produtos, Por esse diagrama é capaz de e perceber os relacionamentos de cada classe, permitindo uma melhor compreensão de funcionamento interno do sistema, tendo em vista como principais classes, comprador, loja e produtos.

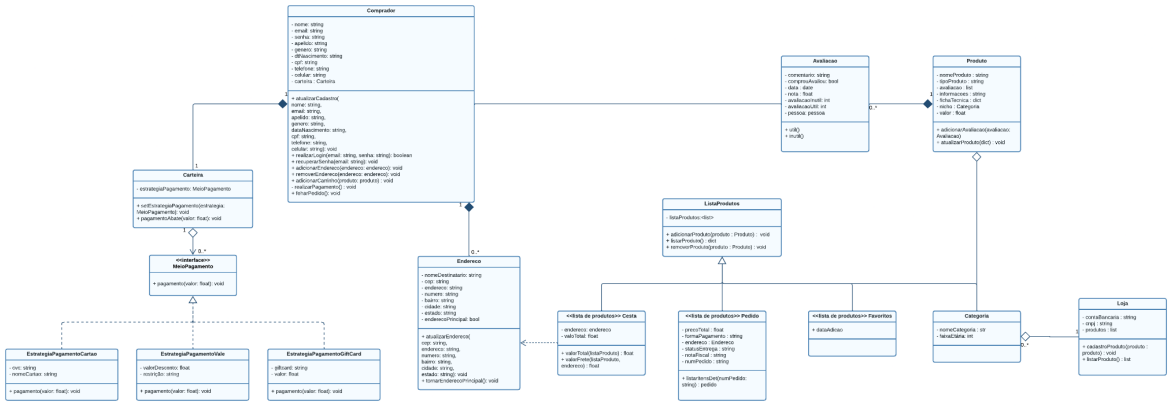


Diagrama de Pacotes

No processo de desenvolvimento do sistema para a Americanas, foi utilizado a ferramenta Draw.io para criar um diagrama de pacotes para prover uma visualização em escala mais alta do sistema, assim como a estrutura arquitetural e as dependências de cada camada. Os responsáveis pela criação do diagrama primeiramente modelam individualmente e depois analisam as duas modelagens e escolheram a que achavam que melhor representava o sistema a nível de pacote. A seguir foram feitos debates sobre a modelagem e aprimoramentos que deveriam ser feitos. A partir deste diagrama podemos compreender melhor a arquitetura.

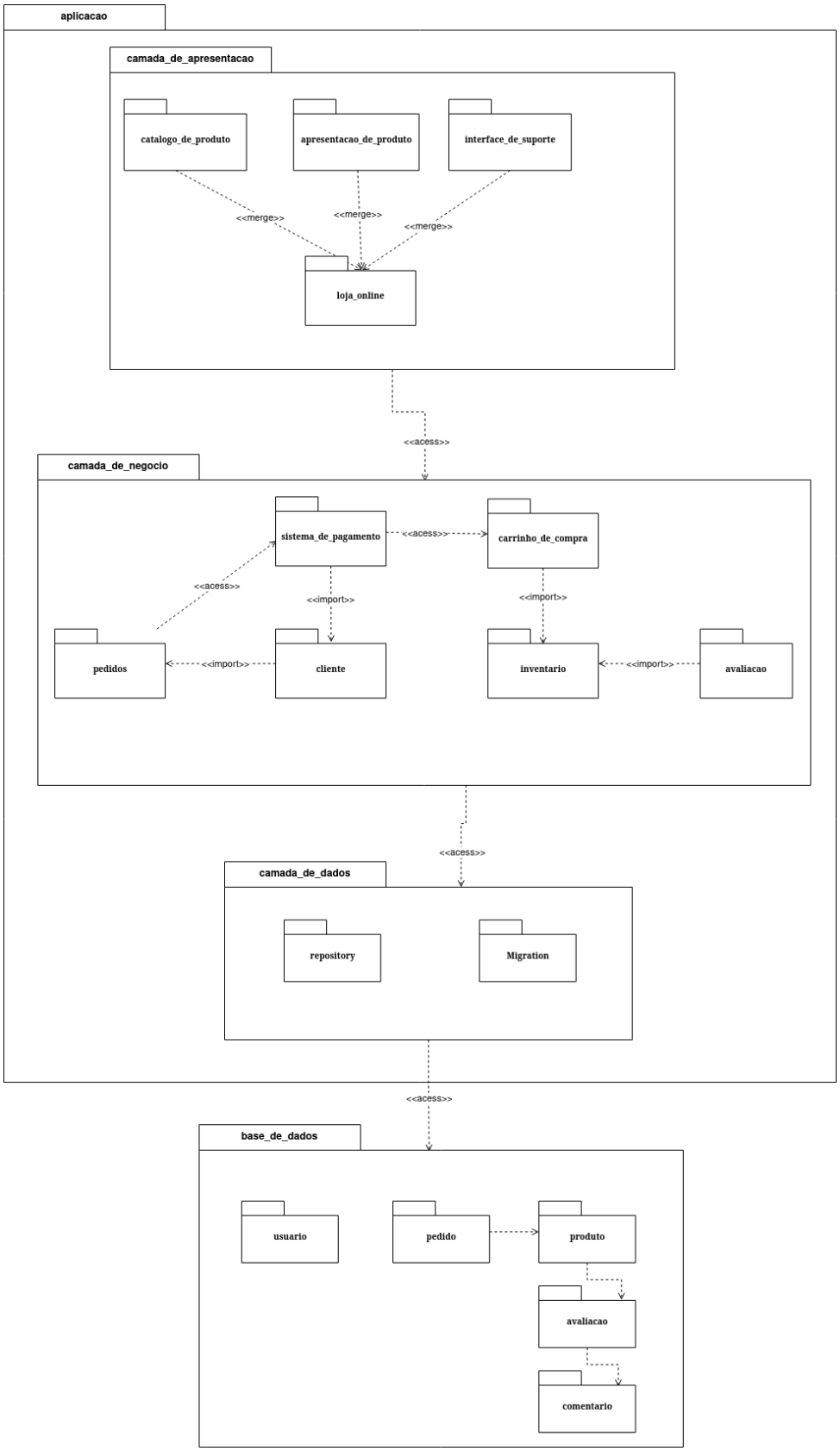
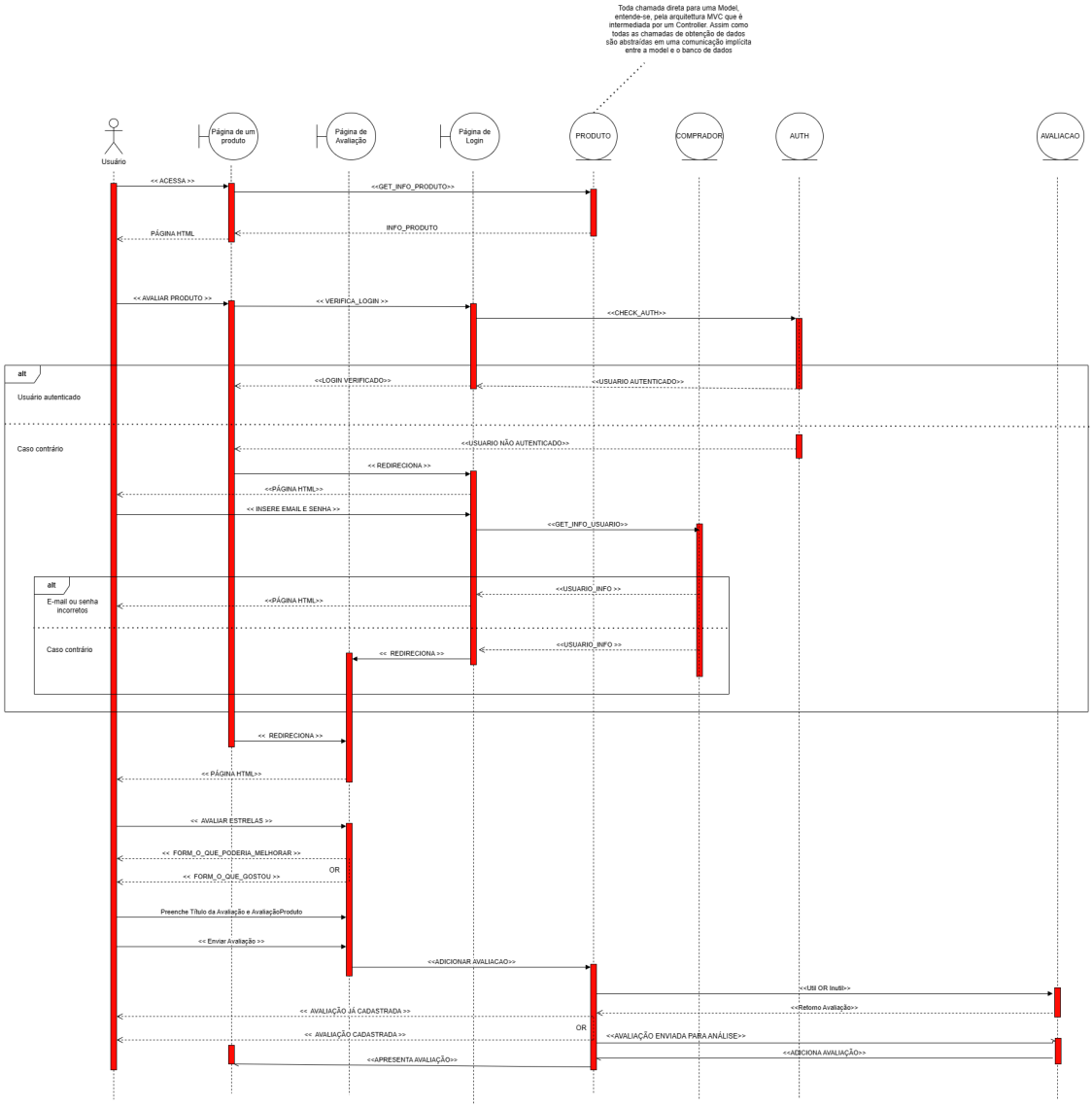

aplicacao
camada_de_apresentacao
catalogo_de_produto
apresentacao_de_produto
interface_de_suporte
<<merge>>
<<merge>>
<<merge>>
loja_online
<<acess>>
camada_de_negocio
sistema_de_pagamento
<<acess>>
carrinho_de_compra
<<acess>>
<<import>>
<<import>>
pedidos
<<import>>
cliente
inventario
<<import>>
avaliacao
<<acess>>
camada_de_dados
repository
Migration
<<acess>>
base_de_dados
usuario
pedido
produto
avaliacao
comentario

Diagrama de Sequência

Foi desenvolvido um diagrama de sequência que descreve o fluxo de avaliação de produtos no site das Americanas, utilizando a ferramenta Lucidchart. Isso envolveu a análise de requisitos, a identificação de atores como usuário e sistema, e a compreensão dos eventos relacionados à avaliação de produtos. Em seguida, foi criado o diagrama, mapeando visualmente os passos do processo no Lucidchart é definido objetos como a API, bancos de dados de produtos e usuário.



6. Visão de Processo

A visão de processos na arquitetura de sistemas de software oferece um entendimento detalhado de como os processos e tarefas são estruturados e interagem. Essa perspectiva visa identificar os

processos-chave do sistema, compreendendo sua comunicação, coordenação e compartilhamento de recursos. É fundamental para modelar a interação entre componentes e módulos em direção aos objetivos do sistema.

Essa abordagem permite mapear as entradas, saídas e fluxos de informação no software, delineando as responsabilidades de cada componente. Ao ser incorporada no Documento de Arquitetura de Software (DAS), essa visão facilita a análise de desempenho, a identificação de gargalos e a otimização do fluxo de trabalho. Essa análise aprofundada garante eficiência, escalabilidade e facilidade de manutenção do software, fortalecendo sua robustez e adaptabilidade às necessidades do negócio.

7. Deployment View

Este segmento descreve uma configuração física de rede na qual o software é implantado e executado para a Americanas. Ele oferece uma visão detalhada do Modelo de Implantação, identificando os elementos físicos e suas interconexões que suportam a execução do software.

Configuração de Hardware:

Servidor Principal: Este nó físico é responsável por executar os principais processos do software. Possui uma arquitetura de servidor de alta capacidade, com múltiplos CPUs para lidar com as demandas intensivas de processamento. Está conectado à rede local (LAN) para comunicação com outros nós.

Servidores de Banco de Dados: Constituídos por um cluster de servidores dedicados para gerenciar o armazenamento e acesso aos dados do sistema. A interconexão desses servidores é estabelecida por meio de uma rede local de alta velocidade (LAN).

Interconexões:

Todos os servidores estão interligados por uma infraestrutura de rede local (LAN) de alta velocidade para facilitar a comunicação eficiente entre os nós.

Mapeamento de Processos:

Os processos do Modelo de Processo são distribuídos da seguinte forma:

Os processos principais estão alocados no Servidor Principal, onde são executados os processos críticos para o funcionamento do sistema.

Os processos de gerenciamento e acesso aos dados estão distribuídos entre os Servidores de Banco de Dados, garantindo um desempenho otimizado no armazenamento e recuperação de informações.

8. Implementation View

Esta visão oferece uma perspectiva abrangente da implantação prevista para o sistema deste projeto. Além de destacar os objetivos, ela também delimita as restrições e limitações associadas à implantação. A abordagem adotada aqui é fornecer uma única visão de implantação por meio do diagrama UML de implantação. Este diagrama ilustra a interação entre as máquinas físicas, máquinas virtuais e contêineres,

oferecendo uma representação visual clara do ambiente de implantação proposto.

Visão Geral (Overview):

A visão de implementação detalha a disposição física e lógica dos componentes do sistema, oferecendo uma representação visual das interações entre os diferentes elementos. Isso permite uma compreensão clara do ambiente no qual o sistema será executado.

O diagrama de componentes pode ser visualizado neste documento no no tópico 2.9.

Camadas (Layers):

O diagrama de implantação representa as camadas de software e suas relações com os recursos físicos. Isso inclui a disposição hierárquica dos módulos e componentes do sistema em diferentes camadas, demonstrando como essas camadas são distribuídas entre as máquinas físicas, máquinas virtuais ou contêineres.

9. Data View (optional)

A perspectiva da Visão de Dados se focaliza na estrutura e organização dos dados persistentes empregados pelo sistema, delineando a forma como os dados são armazenados, acessados e manipulados, abrangendo entidades de dados, seus atributos e interconexões.

Modelo Entidade-Relacionamento(ME-R)

A abordagem do Modelo Entidade-Relacionamento (MER) constitui um tipo de modelo de dados aplicado para representar a estrutura e as relações entre as entidades de um sistema. Este modelo proporciona uma representação das entidades, seus atributos e as conexões entre elas.

No contexto do modelo Entidade-Relacionamento, as entidades são simbolizadas por retângulos, enquanto os relacionamentos são representados por linhas que conectam essas entidades. Cada entidade detém atributos que especificam suas características particulares. A seguir, apresentamos o nosso modelo, que foi elaborado utilizando a ferramenta brmodelo.

Entidades

- USUARIO
- ENDERECO_USUARIO
- PEDIDO
- PRODUTO
- ENDERECO

- AVALIACAO

Descrição das Entidades

- USUARIO (id_usuario, telefone ,name, email, apelido, cpf, data_nascimento, senha, dt_criacao)
- ENDERECO_USUARIO (id_usuario, id_endereco, endereco_padrao , dt_criacao)
- PEDIDO (id_pedido, reputation, data_pedido, status_pedido, detalhes_pedido, preco_total, quantidade, estoque, dt_criacao)
- PRODUTO (id_produto, categoria, nome, tipo, preco, desconto, dt_criacao)
- ENDERECO (id_produto, rua, cidade, cep, complemento, numero, estado, dt_criacao)
- AVALIACAO(id_avaliacao, id_avaliacao, titulo_avalicao, comentario_avaliacao, dt_criacao)

Descriação dos relaciomentos

- PRODUTO **- tem -** AVALIACAO
  Um PRODUTO pode ter nenhuma ou várias AVALIACAOs, e uma AVALIACAO pertence a um único PRODUTO.
  Cardinalidade: **(1:n)**
- USUARIO **- possui -** ENDERECO_USUARIO
  Um USUARIO possui um ou mais ENDERECO_USUARIOs, e um ENDERECO_USUARIO pertence a um único USUARIO.
  Cardinalidade: **(1:n)**
- USUARIO **- faz-** PEDIDO
  Um USUARIO pode fazer nenhum ou vários PEDIDOs, mas um PEDIDO pertence a um único USUARIO.
  Cardinalidade: **(1:n)**
- ENDERECO_USUARIO **- contem -** ENDERECO
  Um ENDERECO_USUARIO pode conter um ENDERECO, e um ENDERECO pertence a nenhum ou vários ENDERECO_USUARIO.
  Cardinalidade: **(n:1)**
- PEDIDO **- contem -** PRODUTO

  Um PEDIDO pode contem nenhum ou vários PRODUTOs, e um PRODUTO pode ser de um ou vários PEDIDOs.
  Cardinalidade: **(n:m)**

Diagrama entidade relacionamento (DE-R)

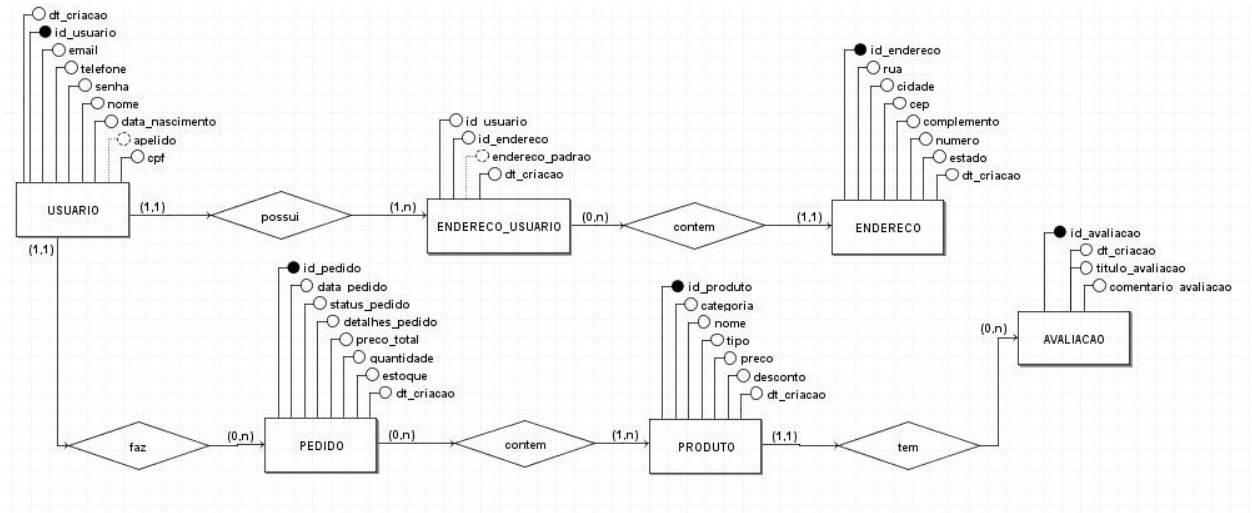


Diagrama Lógico de Dados(DDL)

Esse diagrama pode ser visualizado neste documento no no tópico 2.4.

10. Quality

Para falarmos de tamanho e performance não há como não associar ao termo Escalabilidade. Escalabilidade é a capacidade de um sistema ou processo lidar com uma grande quantidade de trabalho, e com performance. Existem alguns tipos que são relevantes para o Americanas.

A Escalabilidade de dados é a capacidade de um sistema lidar com um aumento no volume de dados sem comprometer seu desempenho. É um conceito-chave na qualidade de software, permitindo que um sistema cresça e se adapte às necessidades em constante mudança. Para alcançar a escalabilidade de dados, é necessário considerar o armazenamento eficiente, a capacidade de processamento e a distribuição de carga. Isso é essencial para garantir que o sistema possa lidar com grandes quantidades de dados sem afetar sua performance.

Escalabilidade horizontal onde aumentamos a quantidade de máquinas no sistema. Para o caso do Americanas imaginamos que é preciso ter um Serviço que permita o aumento da quantidade de dados armazenados então algumas soluções como o serviço da AWS Amazon S3, por exemplo, permite uma solução escalável e inclusive renovável já que serviços em cloud segundo uma pesquisa no data center LocaWeb, aponta algumas vantagens de utilizar serviços em cloud como menor consumo de energia, espaço físico e, uso racional dos recursos naturais (Pinto, Campos, Azevedo, 2021).

A escalabilidade vertical, é a capacidade de um sistema lidar com um aumento na demanda ou carga de trabalho do sistema, fornecendo mais recursos ao componente existente. Por exemplo, se um servidor estiver sobrecarregado e lento devido a um aumento no tráfego de usuários, a escalabilidade vertical envolveria a adição de mais CPU, RAM ou outros recursos para lidar com a carga adicional. Em um cenário on-premise, essa adição seria física, aumentando a capacidade do hardware. No cenário cloud, a escalabilidade vertical é mais dinâmica, uma vez que o poder computacional é virtualmente ilimitado. No ambiente cloud é também possível que essa escalabilidade seja automática para suprir a demanda do momento.

A Escalabilidade de rede é a capacidade de uma rede de computadores lidar com o aumento da demanda e do tráfego, mantendo um desempenho adequado e fornecendo serviços de forma eficiente. É a capacidade de expandir ou dimensionar uma rede para acomodar um maior número de dispositivos, usuários e carga de trabalho. Um exemplo é o Elastic Load Balancing da AWS que distribui o tráfego de rede para melhorar a escalabilidade da aplicação.

Elasticidade: refere-se a habilidade de um sistema escalar de forma variável de acordo com a demanda, Para o Americanas é muito importante lidar com essas picos e baixos de acesso de forma estável, então soluções como serviço da AWS Amazon S3, também possuem essa habilidade sendo a própria AWS o gerenciador do dimensionamento do serviço.

Para o projeto visamos utilizar o modelo de qualidade ISO/IEC 25010, que determina características e subcaracterísticas relevantes para o produto de software