Sistema de Recomendação Baseado em Localização para Ambiente Fechado

Antonio Jonas Gonçalves de Oliveira

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2024

Antonio Jonas Gonçalves de Oliveira

Sistema de Recomendação Baseado em Localização para Ambiente Fechado

Monografia apresentada ao curso Engenharia de Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em
Engenharia de Computação

Orientador: Dr. Alisson Vasconcelos de Brito

Outubro de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

O48s Oliveira, Antonio Jonas Goncalves de.
Sistema de recomendação baseado em localização para ambiente fechado / Antonio Jonas Goncalves de Oliveira. - João Pessoa, 2024.
48 f.

Orientação: Alisson Vasconcelos de Brito.
TCC (Graduação) - UFPB/CI.

1. Sistema de recomendação. 2. Serviços de localização. 3. Algoritmo collaborative filtering. I. Brito, Alisson Vasconcelos de. II. Título.

UFPB/CI CDU 004.421

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

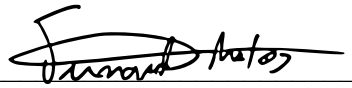
CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

No dia **22** do mês de **Outubro** de **2024**, às **14** horas, em sessão pública na sala **Remota** do Campus I da Universidade Federal da Paraíba, na presença da banca examinadora presidida pelo professor(a) orientador **Alisson Vasconcelos de Brito** e pelos professores **Fernando Menezes Matos** e **Verônica Maria Lima Silva**, o(a) aluno(a) **Antonio Jonas Gonçalves de Oliveira**, apresentou o trabalho de conclusão de curso intitulado: **Sistema de Recomendação Baseado em Localização para Ambiente Fechado** como requisito curricular indispensável para a integralização do Curso de **Engenharia de Computação**.

Após a exposição oral, o(a) candidato(a) foi arguido(a) pelos componentes da banca que reuniram-se reservadamente, e decidiram, **aprovada** a monografia, com nota **9**. Divulgando o resultado formalmente ao aluno e demais presentes, eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

Alisson Vasconcelos de Brito



Fernando Menezes Matos

Verônica Maria Lima Silva

Antonio Jonas Gonçalves de Oliveira

“Em algum lugar, há algo incrível esperando para ser descoberto.” – Carl Sagan

AGRADECIMENTOS

Quero agradecer primeiramente a minha mãe, Cleide, que sempre acreditou em mim e incentivou os meus estudos e meu apoiou em todos os momentos. A minha família, em especial minha tia Ana, por todo o suporte durante o curso, e aos meus queridos amigos e companheiros de curso: Aercio, Alexandre, Gabriel, Hananias, Jurani, Lucena, Luiz, Mateus, Nilbson, Rodrigo, Thales e Thiago por todo o apoio, parceria, e momentos difíceis superados juntos, não estaria aqui sem vocês. Agradeço também ao meu orientador Alisson, pela paciência e assistência durante esse trabalho e na faculdade, e a todos aqueles que participaram dessa jornada comigo.

RESUMO

Os Sistemas de recomendação e os Serviços Baseados em Localização são dois setores que cresceram imensamente nos últimos anos, possibilitando que soluções utilizando essas tecnologias sejam bastante relevantes e com retornos financeiros promissores. Nesse contexto, esse projeto visa desenvolver um sistema de recomendação que utiliza a localização do usuário em uma loja física para criar uma lista de recomendação de produtos personalizados e que também estejam próximos a ele, melhorando a experiência do usuário e a probabilidade de conversão da venda do produto. O algoritmo de recomendação *Collaborative Filtering*, um dos mais famosos e relevantes nessa área, foi utilizado como base e modificado para usar a abordagem da localização do usuário. Foi desenvolvido um protótipo para mostrar a jornada do usuário em uma aplicação mobile, e desenvolvido uma API na linguagem Python que criará uma recomendação dos dez produtos com maior pontuação. O resultado do protótipo conseguiu mostrar desde a tela inicial até a lista de recomendação, e o resultado do algoritmo modificado mostrou não só uma recomendação coerente com o contexto de uma loja física, como também uma evolução na média de pontuação de mais de 16% e um aumento ainda maior no terceiro quartil nos testes realizados, e a ainda houve uma mitigação do problema de *cold start* do algoritmo clássico.

Palavras-chave: Sistema de Recomendação, Serviços Baseados em Localização, Algoritmo Collaborative Filtering.

ABSTRACT

The Recommendation Systems and Localization Services are two sectors that have been growing immensely in the last few years, enabling solutions using those technologies to become very relevant and promising financial returns. In this context, this project aims to develop a recommendation system that uses the user localization in a physical store to create a list of recommended personalized products that are also close to him, improving the overall user experience and the probability of product sales conversion. The Collaborative Filtering recommendation algorithm, one of the most famous and relevant in the field, was used as base and modified to use the user location approach. A prototype was developed to show the user's journey in a mobile application and also an API in Python to create the recommendation of the ten highest scoring products. The prototype's result was able to show from the initial page until the recommendation list, and the result of the modified algorithm implementation showed that not only the generated recommendation was coherent in the physical store context, but also an increase in score of over 16% and a rise even bigger in the third quartile in the performed tests, and also a mitigation in the cold start problem from the classic algorithm.

Key-words: Recommendation Systems, Localization Services, Collaborative Filtering Algorithm.

LISTA DE FIGURAS

1 Modelo de tabela usando SQLAlchemy 18

2 API. 19

3 Grafo com pesos. 20

4 Algoritmo CF Baseado em Usuário e Item. 22

5 Fluxograma 25

6 Layout da Loja 28

7 Modelo do banco de dados MySQL 29

8 Arquitetura do Sistema 29

9 Grafo da Loja 31

10 Matriz de menor distância Dijkstra 31

11 Estrutura da API 34

12 Construção no Figma 36

13 Tela inicial do protótipo 37

14 Segunda tela do protótipo 38

15 Terceira tela do protótipo 39

16 Quarta tela do protótipo 40

17 Visualização dos Resultados no Mapa 43

18 Boxplot da Comparação 44

LISTA DE TABELAS

1 Colunas da base de dados 27
2 Recursos disponíveis na API 35
3 Retorno da API de uma recomendação sem localização 41
4 Retorno da API de uma recomendação com localização 42
5 Comparação Numérica 45

LISTA DE ABREVIATURAS

GPS – *Global Positioning System*

SBL – *Serviços Baseados em Localização*

SQL -*Structured Query Language*

API - *Application Programming Interface*

CF - *Collaborative Filtering*

HTTP - *Hypertext Transfer Protocol*

ORM - *Object Relational Mapper*

Sumário

1 Introdução **15**

1.1 Objetivo geral 15

1.2 Objetivos específicos 15

1.3 Estrutura da monografia 16

2 CONCEITOS GERAIS **17**

2.1 Python 17

2.1.1 Framework Flask 17

2.2 SQLALchemy 17

2.3 Banco de Dados 18

2.4 API Rest 19

2.5 MySQL 19

2.6 Cálculo da Distância Euclidiana 20

2.7 Algoritmo Dijkstra 20

2.8 Algoritmo Collaborative Filtering 21

2.9 Similaridade de Cosseno aplicada ao Collaborative Filtering 23

2.10 Trabalhos Relacionados 24

3 METODOLOGIA **25**

3.1 Fluxograma do Projeto 25

3.2 Pré-processamento de Dados 26

3.3 Modelagem do Banco de Dados 28

3.4 Arquitetura do Sistema 29

3.5 Usar Coordenadas do Usuário para localizar o Departamento 30

3.6 Calculo da Matriz de Distância Miníma Dijkstra 30

3.7 Algoritmo Collaborative Filtering com uso de Distância 32

3.8 API 33

3.9 Rotas da API 35

3.10 Protótipo da Aplicação Mobile 35

4 RESULTADOS E DISCUSSÕES **37**

4.1 Resultado do Protótipo 37

4.2 Resultado da recomendação com localização 40

4.3 Comparação dos Resultados com o Algoritmo Clássico 43

4.4 Mitigação do problema de *Cold Start* 45

5 CONCLUSÕES E TRABALHOS FUTUROS **46**

REFERÊNCIAS **46**

1 Introdução

Sistemas de recomendação possuem um papel crucial em diversos setores, como *e-commerce*, redes sociais e entretenimento no qual fornecer uma experiência personalizada é fundamental para o usuário. Além de possibilitar uma melhor experiência ao usuário sugerindo conteúdos ou produtos, em cenários de vendas, ele é fundamental para aumentar as taxas de conversão, pois sua recomendação personalizada para aquele usuário aumenta a probabilidade de um item ser vendido com base no histórico de produtos comprados anteriormente(RICCI et al., 2010).

Também existe um avanço crescente dos sistemas de geolocalização nos últimos anos e seu grande aumento de popularidade ao redor do mundo, como o Sistema de Posicionamento Global (GPS), soluções para Serviços Baseados em Localização (SBL) possuem um enorme potencial de lucro no mercado de tecnologia, e o interesse de grandes empresas para investir em sistemas que exploram a localização como um serviço com um valor agregado. De forma mais comum, os SBL foram inicialmente mais voltados para aplicações em ambientes externos, com diversas soluções robustas difundidas no meio tecnológico, enquanto a crescente procura para esses sistemas que irão ser usados em espaços fechados e menores é algo mais recente e o mercado está mais dinâmico para o aparecimento de soluções mais promissoras e performáticas(ZHANG et al., 2010).

Em um contexto de ambientes fechados, como uma loja de produtos, a junção dessas duas áreas para criar um sistema de recomendação utilizando a localização pode melhorar bastante a experiência do usuário e, principalmente, o ganho financeiro, pois com base no seu histórico de compras e onde ele está naquele momento na loja, a probabilidade de um produto ser comprado por estar mais próximo dele aumenta muito a força da recomendação, criando uma nova camada de personalização com uma abordagem de usar a distância dele ao produto.

1.1 Objetivo geral

Desse modo, esta monografia busca como seu objetivo principal desenvolver um sistema de recomendação baseado em localização em ambientes fechados, procurando prever uma preferência para um cliente dentro de uma loja com base onde ele está localizado naquele momento e em seu histórico como usuário daquele lugar.

1.2 Objetivos específicos

- Escolher um algoritmo de recomendação da literatura que possa sustentar um sistema de recomendação.

- Utilizar esse algoritmo de recomendação como base para criar um algoritmo modificado com uma abordagem de localização para uma loja física.
- O desenvolvimento de uma API que possa gerar essa recomendação.

1.3 Estrutura da monografia

O capitulo um possui a introdução, objetivo geral e objetivos específicos deste trabalho de conclusão de curso. No capítulo dois é apresentado os conceitos gerais que são precisos para melhor entender o trabalho proposto, trazendo as principais tecnologias usadas que possibilitaram a construção da solução final. O capítulo três possui a metodologia do trabalho, onde foi definido o algoritmo utilizado e o desenvolvimento do sistema. A seguir, no capítulo quatro é apresentado os resultados obtidos neste projeto após o desenvolvimento do sistema e seus testes. Por fim, o capítulo cinco traz as conclusões e mostra os trabalhos futuros com o intuito de prover uma melhora na solução final da monografia.

2 CONCEITOS GERAIS

A seguir são apresentados os conceitos que servem de base para o sistema proposto, com as principais tecnologias que foram utilizadas.

2.1 Python

Pode ser descrita como uma linguagem de programação orientada a objeto, interpretada, de alto nível e com uma semântica com característica mais dinâmica. Uma de suas grandes vantagens é ser simples, fácil de aprender e uma sintaxe atrativa para os programadores e o que possibilita uma redução no custo de manutenção de código(PYTHON, 2021).

O Python também é uma linguagem que tem frameworks, que pode ser definido como uma estrutura de software que facilita a criação de aplicações web, garantindo padronização, ferramentas e bibliotecas que possibilitam o aceleramento no desenvolvimento de aplicações, minimizando a complexidade, provendo uma fundação robusta para o projeto desenvolvido(PYTHON, 2021). Os frameworks mais conhecidos do Python são Flask, Django, FastAPI e Tornado(GRINBERG, 2018). Também possui uma biblioteca de interação direta com banco de dados SQL, que é a SQLALchemy, que oferece uma abordagem de ORM(Object Relational Mapping) como entenderemos melhor em seguida.

2.1.1 Framework Flask

O Flask é um mini framework síncrono poderoso para a criação de web aplicativos poderosos e foi escolhido por ter uma vantagem natural em relação a boa parte dos frameworks disponíveis na literatura. Ele possibilita ao desenvolvedor um controle criativo de suas aplicações, como por exemplo possui suporte a todos os bancos de dados relacionais e não-relacionais também, pois ele garante uma flexibilidade na escolha de ferramentas e componentes a serem usados, dos mais comuns até componentes desenvolvidos pessoalmente(GRINBERG, 2018).

2.2 SQLALchemy

O SQLAlchemy é uma popular biblioteca da linguagem Python que possibilita a interação usando ORM, que é a prática que permite um mapeamento de classes usando Python diretamente para uma tabela em um banco de dados SQL, abstraindo o acesso ao banco utilizando uma interface para possibilitar uma forma mais intuitiva e eficiente de manipulação de consultas SQL(SQLALCHEMY, 2024).

Uma das principais vantagens é a possibilidade de criação de tabelas e consultas usando a sintaxe Python, que é uma forma bem mais atrativa que o uso da linguagem SQL, que é mais verbosa. A biblioteca é responsável pela tradução de uma sintaxe para a outra.(FLASK-SQLALCHEMY, 2024).

Figura 1: Modelo de tabela usando SQLAlchemy

```python
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, ForeignKey, DateTime, func
from sqlalchemy.orm import relationship

Base = declarative_base()


class PurchaseHistory(Base):  3 usages  jonas *
    __tablename__ = 'purchase_history'

    id = Column(Integer, primary_key=True, autoincrement=True)
    imei_number = Column(String(20), unique=True, nullable=False)
    price = Column(String(255), nullable=False)
    coordinates = Column(String(255), nullable=False)
    created_at = Column(DateTime(timezone=True), server_default=func.now())

    # Relacionamento
    uniq_id = Column(String(255), ForeignKey('inventory.unique_id'), nullable=False)
    inventory = relationship( argument: "Inventory", back_populates="purchase_history")

```

Fonte: Autoria Própria

Na Figura 1 podemos visualizar um exemplo de modelagem usando o SQLAlchemy em linguagem Python de forma direta para definir os elementos e as relações da tabela de dados.

2.3 Banco de Dados

O banco de dados atualmente é um elemento essencial de um software e pode ser definido como uma coleção de dados persistentes e é usado por alguma aplicação, onde pode ser acessada, gerenciada e deletada de forma intuitiva. Eles possuem tipos diferentes, como o mais comum, o banco de dados relacional, até um banco de dados nuvem, ou banco de dados não-relacionais. (KOLONKO, 2018).

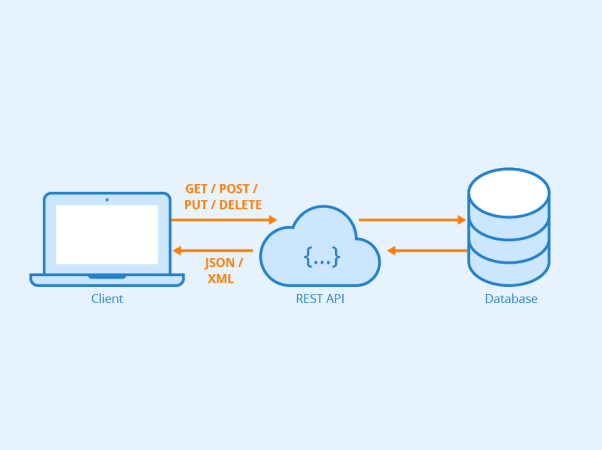
O banco de dados relacional pode ser descrito como um armazenamento em tabelas compostas por linhas e colunas, onde existe uma padronização para representar ou consultar os dados, assim eles podem ser usados de maneiras diferentes. As tabelas possuem pelo menos uma categoria de dados em cada coluna e as linhas possuem as instâncias de dados para as categorias definidas. Outra grande força desse modelo com o tempo foi a linguagem de consulta estruturada (SQL), sendo ela normalmente usada como interface padrão para manipular um banco de dados relacional. (KOLONKO, 2018).

O banco de dados não relacional, como o nome sugere, não utiliza tabelas de linhas e colunas, como é usado nos sistemas mais tradicionais. Ele utiliza um modelo que busca a otimização que atende determinados requisitos que os dados armazenados necessitam. Tem como característica possuir os tipos de dados mais bem definidos, como JSON, pares de chave de valores simples ou gráficos. (KOLONKO, 2018).

2.4 API Rest

Uma *Application Programming Interface*(API) pode ser descrita como um conjunto de regras permitindo a comunicação entre os diferentes programas. Definindo a melhor forma que um desenvolvedor de software crie um programa em um servidor que vai ter uma comunicação com vários aplicativos clientes. A API Rest se define como uma API que segue os padrões Rest, que são um conjunto de restrições de arquitetura, e pode ser implementada de formas variadas. A Figura 2 apresenta um exemplo da arquitetura da API Rest, onde o usuário faz uma solicitação para API que se comunicará com uma base de dados, processará a requisição e devolverá uma resposta para o cliente.(NAEEM, 2020)

Figura 2: API.



Fonte: (NAEEM, 2020)

2.5 MySQL

O MySQL é considerado o maior banco de dados de código aberto do mundo, e é usado em diversas plataformas mundiais famosas, como Netflix, Facebook, Youtube entre outras. É um banco de dados relacional, ou seja, o armazenamento de dados acontece de forma que são usadas tabelas separadas, objetos como colunas, linhas permitem uma melhor organização e flexibilidade na construção do ambiente que está sendo armazenado os dados, com regras que ditam o comportamento das relações entre tabelas e colunas, garantindo uma base consistente e com dados bem estruturados.(ORACLE, 2024)

Por ser um banco de código aberto, qualquer pessoa consegue usar o seu software, o que permite seu vasto uso e suporte. Também é compatível com várias linguagens, sendo algumas dela como Java, Go, C++ e Python, usada nesse trabalho, além de trazer benefícios como a facilidade de sua utilização, a confiabilidade, que por definição se trata da capacidade de um sistema continuar operando corretamente durante um longo período, a escalabilidade, onde se refere a capacidade de manter um alto desempenho mesmo que exista um aumento na carga de dados ou usuários, e por fim seu desempenho, que é garantido por diversos testes de qualidade realizados com indicadores como benchmarks para assegurar o funcionamento da ferramenta.(MYSQL, 2024)

2.6 Cálculo da Distância Euclidiana

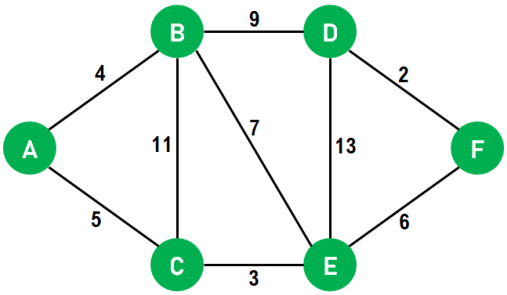
Uma medida muito difundida quando se fala de cálculo entre dois pontos em um espaço cartesiano, é o mais comum dos tipos usados na geometria e tem importante aplicação na área computacional, como visão computacional, aprendizado de máquina, análise de dados entre outros. Abaixo podemos ver a fórmula que é utilizada para realizar esse cálculo(STRANG, 1993).

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2.7 Algoritmo Dijkstra

Um algoritmo bastante usado na teoria de grafos e ciência computacional, desenvolvido em 1959 pelo holandês Edsger W. Dijkstra, é usado para encontrar o menor caminho entro nós num grafo com pesos.

Figura 3: Grafo com pesos.



Fonte: (VENKIS, 2024)

A Figura 3 representa um grafo com pesos onde aplicando o algoritmo de Dijkstra é possível resolver o problema de caminho mais curto de uma única fonte, ou seja, um vértice é escolhido para iniciar e o algoritmo vai achar a menor distância entre os

vértices vizinhos. Abaixo é representado o pseudocódigo que representa como o algoritmo funciona.(CORMEN T. H., 2009)

Listing 1: Pseudocódigo Algoritmo Dijkstra

```
1. Inicialize as distancias de todos os vertices:
   PARA cada vertice v in V[G]
       dist[v] = INFINITO
       prev[v] = NULO
   dist[s] = 0

2. Crie uma fila de prioridade Q com todos os vertices de G:
   Q = V[G]

3. Enquanto Q nao estiver vazia:
   ENQUANTO Q != Vazio
      u = EXTRAIR-MIN(Q)

      PARA cada vizinho v de u
         SE v de Q ENTAO
            alt = dist[u] + w(u, v)
            SE alt < dist[v] ENTAO
               dist[v] = alt
               prev[v] = u

4. Retorne dist[] e prev[]
```

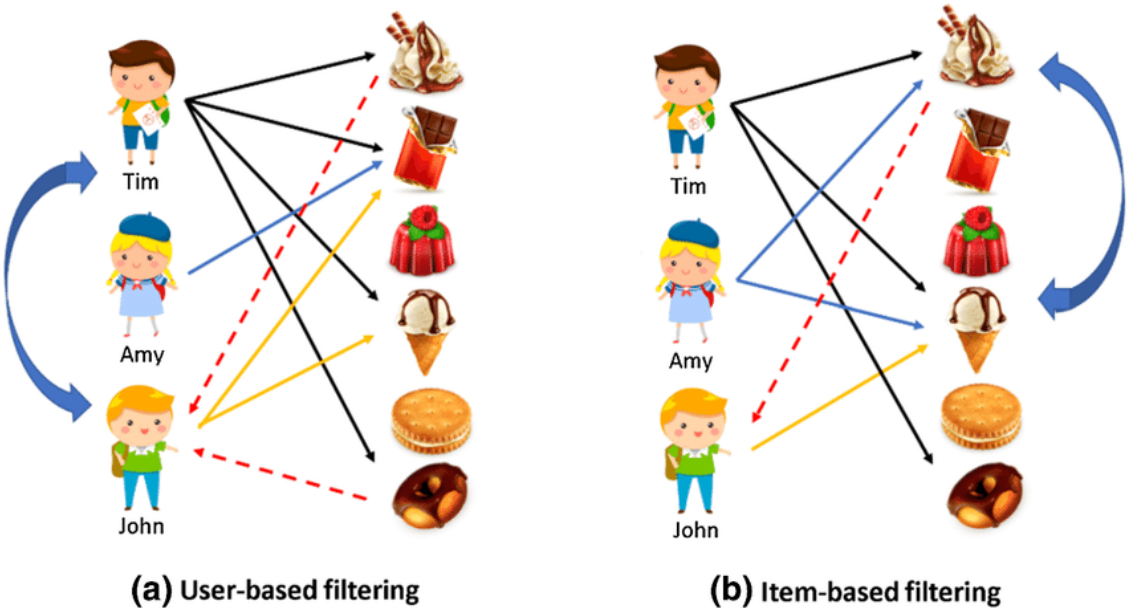
2.8 Algoritmo Collaborative Filtering

Esse algoritmo é uma das mais notórias técnicas que existe quando se fala de sistema de recomendação, ele prevê as preferências de um usuário através das informações que são obtidas de uma coleção de outros usuários e parte do princípio onde esse usuários que compartilham de interesses, preferências ou comportamentos no passado devem continuar a ter interesses similares no futuro(RESNICK et al., 1994).

O *Collaborative Filtering* usa uma matriz para mapear o comportamento do usuário e dos itens no sistema, e então valores são definidos a partir dela para gerar os pontos de dados em um espaço vetorial. Para calcular a distância entre esses pontos, as métricas mais famosas são: a similaridade de cosseno e o coeficiente de correlação de Pearson(RICCI et al., 2010).

O sistema de recomendação CF pode ser divido em dois tipos primários, sendo eles baseado em memória e baseado em modelo. Pode-se entender o primeiro tipo primário como uma extensão do algoritmo de KNN. Isso se deve ao fato de ocorrer a tentativa de predição do comportamento de um usuário alvo em relação ao um item baseado em um conjunto de itens e usuários. Por outro lado, o tipo primário baseado em modelo utiliza técnicas de ciência de dados e aprendizado de máquina(IBM, 2024).

Figura 4: Algoritmo CF Baseado em Usuário e Item.



Fonte: (CHEN YC., 2021)

Neste projeto foi utilizado o tipo baseado em memória, que pode ser dividido em dois subgrupos, como observados na Figura 4, sendo eles, filtragem baseada em usuário e filtragem baseada em item. Enquanto o primeiro baseia a recomendação na semelhança entre os usuários, o segundo é feito na semelhança entre os itens que o usuário já interagiu, ou seja, itens comprados ou avaliados.

O funcionamento do algoritmo de recomendação baseado em item se dá através do cálculo de uma pontuação atrelada a cada item que indica a força daquela recomendação e é retornado uma lista de itens juntos com suas respectivas pontuações ordenados com as melhores recomendações primeiro. Os passos principais que o algoritmo realiza são coletar o histórico de interações entre usuários e itens, sendo elas por exemplo, a quantidade de vezes que o item foi comprado, depois um dos passos mais importantes que seria o núcleo do CF, que é a construção de uma matriz de similaridade entre os itens, que é um cálculo de quão um item é similar a outro usando a medida de similaridade de cosseno. Abaixo podemos ver o pseudocódigo para o algoritmo baseado em item(RICCI et al., 2010).

Listing 2: Pseudocódigo Algoritmo de Collaborative Filtering

```
Entrada:
  - historico de compras de usuarios
  - matriz de similaridade entre os itens

Saida:
  - lista de itens recomendados para o usuario

PARA cada item i no historico de compras do usuario:
 PARA cada item j que o usuario ainda nao comprou:
    CALCULAR a pontuacao de recomendacao para o item j:
        - Pontuacao de recomendacao para
        j += similaridade(i, j) *
        avaliacao_do_usuario_para_item_i

ADICIONAR a pontuacao de j a lista de recomendacoes.

ORDERNAR os itens pela pontuacao de recomendacao em
ordem decrescente.

RETORNAR os itens com as maiores pontuacoes
como recomendacoes.
```

Por fim, com a matriz pronta é possível gerar a recomendação, com o sistema fazendo a análise dos itens que o usuário interagiu e recomenda outros similares através de pontuação, que é feita somando a similaridade dos itens comprados e ainda não comprados, ponderados pela quantidade de vezes que o usuário interagiu com os itens(RICCI et al., 2010).

2.9 Similaridade de Cosseno aplicada ao Collaborative Filtering

A similaridade do cosseno é uma métrica muito utilizada em sistemas de recomendação, especialmente o algoritmo CF, tanto o baseado em usuários quanto o em itens. Ela mede a similaridade entre dois vetores de características, calculando o cosseno do ângulo entre eles. A fórmula da similaridade entre dois vetores A e B pode ser vista a seguir(RICCI et al., 2010).

$$CosineSimilarity(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

No algoritmo de CF baseado em itens é medido a similaridade entre dois itens com base nas interações que o usuário teve com outros itens. A fórmula modificada para esse cenário pode ser vista a seguir.

$$Sim(i,j) = \frac{\sum_{u \in U} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U} r_{u,j}^2}}$$

Onde:

- Sim(i,j) é a similaridade entre o item i e o j.
- U é o conjunto de usuários que interagiram tanto com o item i como com o j.
- $r_{u,i}$ é a interação do usuário u para o item i.
- $r_{u,j}$ é a interação do usuário u para o item j.
- $\sum_{u \in U} r_{u,i} \cdot r_{u,j}$ é o produto escalar entre as interações dos itens i e j pelos usuários comuns.
- $\sqrt{\sum_{u \in U} r_{u,i}^2}$ e $\sqrt{\sum_{u \in U} r_{u,j}^2}$ são as normas dos vetores de avaliações dos itens i e j.

2.10 Trabalhos Relacionados

O (YELP, 2024) é um sistema de recomendação popular que utiliza avaliações e classificações de usuários para sugerir estabelecimentos locais, como restaurantes, lojas e serviços. No contexto de recomendação, ele adota um modelo híbrido, combinando filtro colaborativo, analisando as preferências de usuários semelhantes, e fatores baseados em conteúdo. E também incorpora localização geográfica, sugerindo estabelecimentos próximos ao usuário com base em sua localização atual.

Foi proposto por (LIANGXING; AIHUA, 2010) um sistema de recomendação híbrido para clientes de varejo de roupas, combinando métodos de filtro colaborativo e gráficos para capturar preferências pessoais e dinâmicas temporais. Ele utiliza dados implícitos de atividades do usuário para calcular classificações de preferências. A abordagem equilibra a personalização com mudanças nas tendências de moda ao longo do tempo, gerando recomendações que levam em consideração essas variações.

Já (YUCHEN; WU, 2019) discute um sistema de recomendação personalizado que integra experiências de compras offline e online, conhecido como O2O. A proposta do sistema é que os dados coletados durante a experiência de compra em lojas físicas, por meio de sensores, beacons ou apps móveis, sejam usados para oferecer recomendações personalizadas quando o cliente acessa a plataforma online, e vice-versa. O objetivo é criar uma experiência de compra contínua e eficiente, levando em conta o histórico de interações do cliente tanto online quanto offline.

3 METODOLOGIA

Nesse capítulo são mostrados os processos feitos para a criação do Sistema de recomendação utilizando a localização do usuário em uma loja física. O objetivo será entender as decisões tomadas e as etapas que fizeram parte do desenvolvimento do projeto.

A primeira parte visa aplicar um processamento nos dados de entrada para melhor representar o cenário real de uma loja física que possui vários clientes e itens que são vendidos, bem como um histórico de compras. O cliente usará uma aplicação em seu celular, fornecendo a sua localização, onde será possível determinar em que departamento da loja ele se encontra e assim fornecer mais um dado de entrada para o algoritmo. Com os dados bem definidos, a segunda parte do projeto é o desenvolvimento da API com o sistema de recomendação.

3.1 Fluxograma do Projeto

Figura 5: Fluxograma

Fonte: Autoria Própria

Na Figura 5 é mostrado um fluxograma de todo o projeto, descrevendo todas as ações que acontecem em cada etapa do processo, iniciando com o usuário utilizando um aplicativo até o retorno do resultado da recomendação para ele, realizando todas as etapas. Nas próximas seções será mostrado o passo a passo de cada uma dessas etapas de desenvolvimento.

3.2 Pré-processamento de Dados

O objetivo é criar um *software* que funcione em um cenário real, porém existe uma limitação física em relação à utilização de uma loja para obter os dados reais. Sendo assim, foi utilizada uma base de dados de uma loja real, a *Target*. Esse conjunto de dados foi encontrado na plataforma Kaggle(KAGGLE, 2024), uma das ferramentas mais renomadas com um vasto repertório de base dados para exploração e resolução de problemas. Nesse contexto, foi necessário aplicar um pré-processamento para adequar os dados de entrada de forma que melhor representa à realidade.

Na Tabela 1 podemos ver todos os dados da base escolhida, com todos os atributos que todos os produtos tem, principalmente a categoria do produto, preço, características e seu identificador único. Foi feito uma tratativa no conjunto de dados, pois nem todos esses atributos faziam sentido serem usados na aplicação ou estavam redundantes, então é importante entender o cenário que se precisa e remover essas colunas para trabalhar com um conjunto de dados mais claro e coeso. As colunas que possuem *raw* em seus nomes foram eliminadas por serem duplicadas e colunas como *sub category 2 e 3*, *sku*, *gtin13*, *available branch* também pois eram informações que não se encaixavam na tratativa.

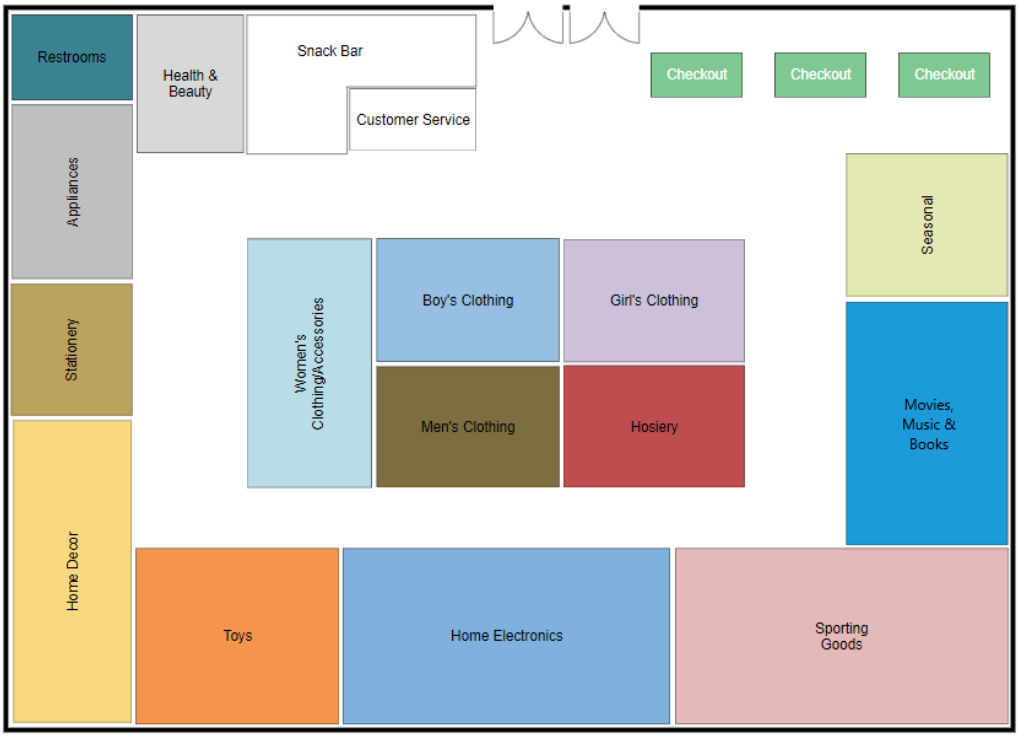
Tabela 1: Colunas da base de dados

Coluna	Descrição
title	Nome do produto
url	Link do produto
brand	Marca do produto
main image	Imagem principal
price	Preço do produto
availability	Disponibilidade daquele produto
primary category	Categoria principal
scraped at	Data da inserção do produto
highlights	Características do produto
uniq id	Identificador único
sku	Unidade de manutenção do estoque
raw description	descrição não formatada
sub category 1	categoria secundária 1
sub category 2	categoria secundária 2
sub category 3	categoria secundária 3
availableDelivery	Disponibilidade de entrega
description	Descrição do produto
currency	Moeda
specifications	Especificações do produto
images	Imagens
gtin13	Código de produtos
available branch	Disponibilidade de loja
raw specifications	Especificações não formatadas
raw highlights	características não formatadas

(KAGGLE, 2024)

Também foi preciso definir um *layout* para representar a loja física. Dessa forma foi escolhida uma que possui vários departamentos que existem na base de dados do inventário, para que a interação entre itens e loja ficasse contundente. Na Figura 6 está a representação da loja.

Figura 6: Layout da Loja



Fonte: (SMARTDRAW, 2024)

Foi convencionado que os cinco departamentos centralizados ficam em um primeiro andar e o restante em um térreo, e ao todo foram utilizados 11 departamentos para o sistema.

O *dataset* é um arquivo CSV que foi transformado em uma tabela chamada *inventory* no banco de dados *MySQL*. Era necessário ainda ter um novo atributo que representasse o departamento que aquele item estava, então foi adicionado a coluna *coordinate* e usando o atributo *primary category* foi possível fazer essa correlação e definir um departamento para cada item.

3.3 Modelagem do Banco de Dados

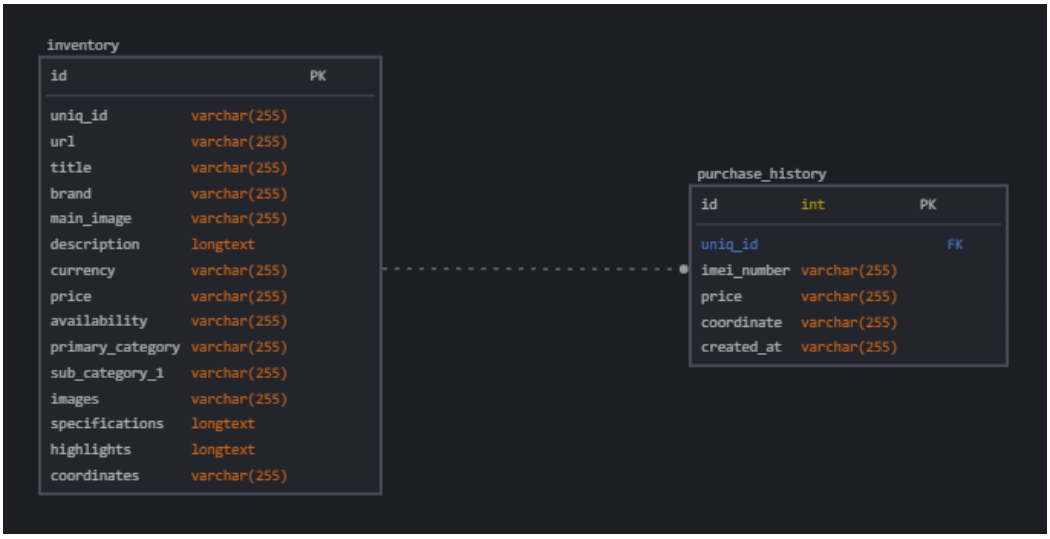
O banco de dados utilizado foi o MySQL para a criação de uma tabela para a base de dados *inventory*, que possui o inventário da loja, e uma tabela *purchase history* com dados de compra de itens por diversos usuários, pois o algoritmo precisa de um histórico de interação entre usuário-itens, então foi preciso gerar esses dados via código.

Na tabela *inventory*, é possível encontrar todos os atributos vistos anteriormente da base de dados, além da adição de uma nova coluna *coordinate* que possui o departamento de cada item. A coluna *uniq id* representa uma identificação única que cada item da base

possui, e também é uma chave estrangeira da tabela *purchase history*, ou seja, permite acessar os registros da outra tabela.

Já a tabela *purchase history* foi populada para representar um cenário real de um histórico de uma loja, onde o campo *imei number* representa a numeração única que cada dispositivo celular possui, e assim, um cliente único que realizou a compra na loja, sabemos quando a compra foi feita, seu valor e o identificador único do item. Na Figura 7 pode ser visualizado a modelagem das tabelas e seu relacionamento.

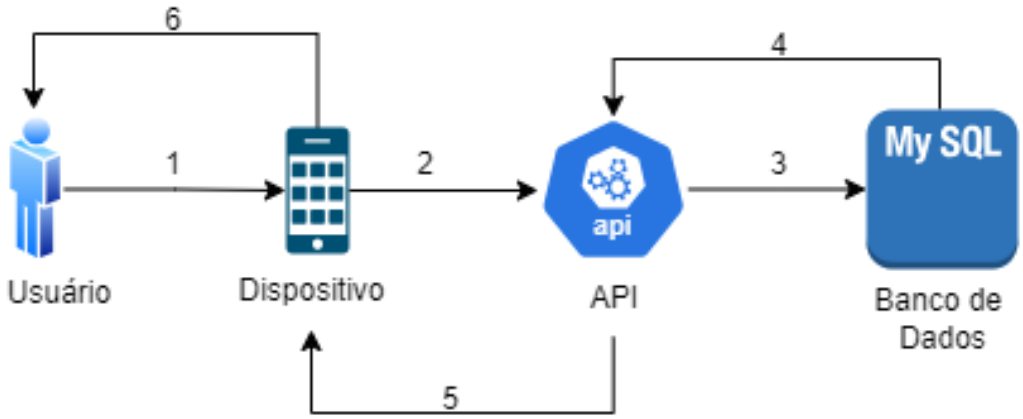
Figura 7: Modelo do banco de dados MySQL



Fonte: Autoria Própria

3.4 Arquitetura do Sistema

Figura 8: Arquitetura do Sistema



Fonte: Autoria Própria

Na Figura 8 podemos observar todo o fluxo da aplicação, iniciando na etapa onde o usuário acessa à aplicação do seu dispositivo, permite o uso de sua localização, assim, essa informação junto com o *imei number* é enviado a API, que usará o identificador único do dispositivo para localizar o histórico de compras na loja caso o usuário já tenha algum registro e com a localização será feito um processo para determinar em que departamento da loja ele se encontra, como será visto na próxima seção.

Após ter todas as informações de entrada, o sistema faz o cálculo da lista de recomendação organizada em ordem de pontuação de cada item e exibe na tela do dispositivo essa lista de produtos com as suas respectivas informações.

3.5 Usar Coordenadas do Usuário para localizar o Departamento

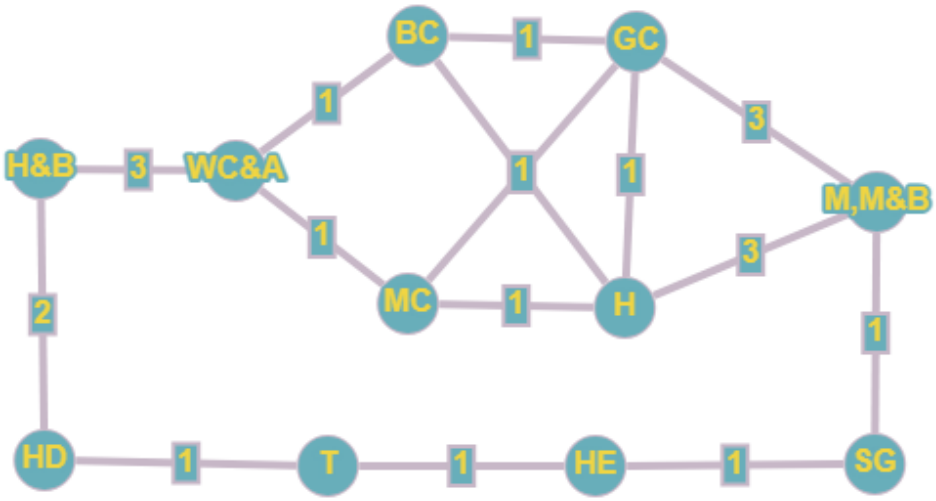
Uma etapa do projeto é o tratamento da localização, que é um dado de entrada do usuário que vai receber a recomendação. Essa localização está representada em uma coordenada geográfica, latitude e longitude, que estão no mesmo sistema de coordenadas da loja. Então foi definido que cada departamento possui uma área delimitada por um conjunto de coordenadas, ou seja, um polígono. Assim, com a coordenada do usuário, o sistema definirá em qual departamento da loja o cliente se encontra, caso esteja em algum.

3.6 Calculo da Matriz de Distância Miníma Dijkstra

É importante ressaltar que, por se tratar de uma loja física, não é possível o uso da distância euclidiana para obter a medida do usuário para o produto por existir barreiras físicas como paredes e pavimentos entre os pontos. Dessa forma, para solucionar essa situação foi utilizado o algoritmo de dijkstra para calcular a distância corretamente.

A Figura 9 representa um grafo com pesos que foi criado a partir do *layout* da loja descrito na Figura 6. A definição do peso dos grafos foi feita de forma intuitiva pela distância dos departamentos de forma direta, sendo menor de um departamento ao lado do outro e peso maior se estão em andares diferentes. A partir desse grafo é aplicado o algoritmo de dijsktra e gerar uma matriz de menor distância. Nessa matriz as linhas representam as menores distâncias, e ela é usada no sistema para ajustar as pontuações de recomendação do item, com base na proximidade do usuário ao departamento.

Figura 9: Grafo da Loja



Fonte: Autoria Própria

Na Figura 10 podemos visualizar como ficou a matriz gerada pelo algoritmo.

Figura 10: Matriz de menor distância Dijkstra

```
      "HB", "HD", "T", "HE", "SG", "MMB", "H", "GC", "MC", "BC", "WCA"
"HB", [0,    2,    3,    4,    5,    6,    5,    5,    4,    4,    3]
"HD", [2,    0,    1,    2,    3,    4,    7,    7,    6,    6,    5]
"T",  [3,    1,    0,    1,    2,    3,    6,    6,    7,    7,    6]
"HE", [4,    2,    1,    0,    1,    2,    5,    5,    6,    6,    7]
"SG", [5,    3,    2,    1,    0,    1,    4,    4,    5,    5,    6]
"MMB",[6,    4,    3,    2,    1,    0,    3,    3,    4,    4,    5]
"H",  [5,    7,    6,    5,    4,    3,    0,    1,    1,    1,    2]
"GC", [5,    7,    6,    5,    4,    3,    1,    0,    1,    1,    2]
"MC", [4,    6,    7,    6,    5,    4,    1,    1,    0,    2,    1]
"BC", [4,    6,    7,    6,    5,    4,    1,    1,    2,    0,    1]
"WCA",[3,    5,    6,    7,    6,    5,    2,    2,    1,    1,    0]
```

Fonte: Autoria Própria

3.7 Algoritmo Collaborative Filtering com uso de Distância

A próxima etapa é o núcleo do projeto, que é a utilização do algoritmo *Collaborative Filtering* com a abordagem de usar a localização do usuário para gerar uma matriz de menor distância para os produtos da loja. O uso desse novo parâmetro de entrada adiciona uma nova dimensão ao algoritmo, tornando mais relevante em um ambiente fechado, como o de uma loja, e influenciando diretamente na probabilidade de compra de um produto pelo usuário.

O algoritmo clássico não leva em conta o contexto físico de onde o usuário está, o que muda usando essa abordagem. Um dos problemas mais conhecidos do CF é o *cold start*, que seria a entrada de um novo usuário no sistema, onde sem registros de interação anteriores, impossibilita uma recomendação personalizada que faça sentido para ele. Com essa modificação no algoritmo, esse problema é mitigado, pois o sistema terá agora um ponto de partida, que é a informação da localização do usuário que será convertida em uma recomendação mais especializada(IBM, 2024). A fórmula ajustada pode ser vista a seguir.

$$P(i,j) = \frac{\sum_{j \in N(i)} \text{sim}(i,j) \cdot r_{u,j} \cdot f(d_{u,i})}{\sum_{j \in N(i)} |\text{sim}(i,j)|}$$

Um ponto importante na modificação do algoritmo é o desafio de não deixar a inclusão da distância influenciar muito no cálculo da recomendação, então é adicionado um fator multiplicativo que ajusta essa influência e precisa ser calibrado corretamente para não ofuscar a similaridade do item, e se for muito baixo não impactar da forma desejada no algoritmo.

Assim, foram feitos testes utilizando entre a casa de 20 a 40% de aumento na pontuação para o mesmo departamento e departamentos próximos, e também uma queda da pontuação, em cenário de um produto muito longe.

Esse equilíbrio entre similaridade e distância é crucial, pois um item que esteja em um departamento mais distante pode ser bastante similar, assim, ainda sendo uma recomendação relevante, ou também um item muito próximo que é irrelevante para aquele usuário.

Listing 3: Pseudocódigo Algoritmo de Collaborative Filtering com Distancia

```
Entrada:
  - historico de compras de usuarios
  - matriz de similaridade entre os itens
  - matriz de distancia entre os departamentos dos itens

Saida:
  - lista de itens recomendados
  para o usuario

1. PARA cada item i no historico de compras do usuario:
 2. PARA cada item j que o usuario ainda nao comprou:
   3. CALCULAR a pontuacao de recomendacao para o item j:
    - Pontuacao base = similaridade(i, j) *
      avaliacao_do_usuario_para_item_i
    - Ajuste de distancia = funcao de penalizacao/diminuicao
      com base na distancia(i, j)
      - Pontuacao final = Pontuacao base *
        Ajuste de distancia
 4. ADICIONAR a pontuacao ajustada de j a lista
   de recomendacoes.

5. ORDENAR os itens pela pontuacao de recomendacao ajustada
em ordem decrescente.
6. RETORNAR os itens com as maiores pontuacoes como
recomendacoes.
```

3.8 API

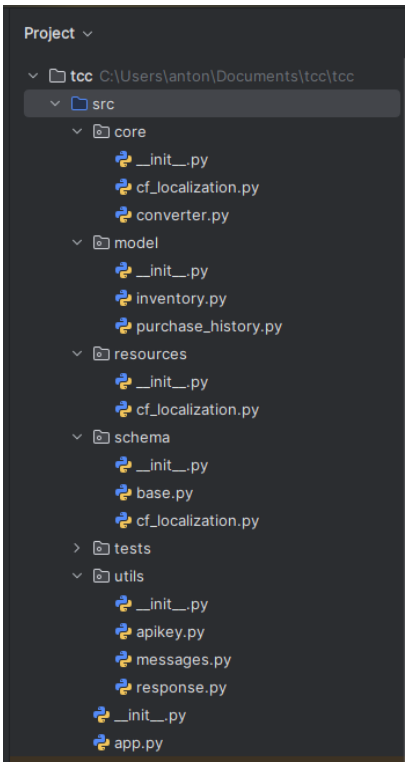
A API que foi implementada é onde todas as etapas de processamento do projeto são realizadas. Os principais endpoints são o de conversão da localização em departamento da loja, cálculo da matriz dijkstra de menor distância, encontrar o departamento através de um item recomendado e, principalmente, a recomendação usando o algoritmo CF com e sem a influência da distância.

Foi utilizado a linguagem *Python* na implementação da API, devido a todas as vantagens de rapidez, versatilidade e disponibilidade de frameworks como o *Flask*, que é excelente para a criação de API's e integração com ORM's como o *SQLAlchemy* que é utilizado nesse projeto.

A API inicialmente recebe dois dados de entrada, o *imei number* e as coordenadas de localização atual do usuário. Dessa forma, através do endpoint departament e das informações passadas como entrada, obtêm-se o departamento em que o usuário se encontra. Assim, é possível calcular a matriz dijkstra de menor distância do departamento atual para os outros departamentos.

E o principal endpoint da implementação é o que faz a criação da recomendação usando a matriz de distância dijkstra do departamento que o usuário se encontra, bem como o histórico de compra que se encontra no banco de dados e é efeito o cálculo da pontuação retornando os dez produtos com maiores pontuações.

Figura 11: Estrutura da API



Fonte: Autoria Própria

Na Figura 11 a API está estruturada em três principais módulos, sendo eles *resources*, *model* e *core*. Essa abordagem foi utilizada pois ela garante uma separação de responsabilidades, facilitando uma maior organização do projeto, bem como manutenção e escalabilidade, facilitando quando é preciso reutilizar funções do projeto ou adicionar novas.

Na camada *resources* está estruturada as rotas da aplicação, e sua função é permitir a interação entre o usuário e o sistema, utilizando as outras camadas para executar ações e processar respostas. Na camada *model* é definida a estrutura de dados, representando as

entidades do sistema. Nessa aplicação é usado o banco de dados relacional *MySQL* junto a biblioteca SQLAlchemy para realizar os mapeamentos via ORM, validando a integridade dos dados, a lógica de persistência e todas as entidades e suas relações. Na camada *core* está toda a lógica central que foi implementada para aplicação. Incluindo as lógicas de negócio, serviços, cálculos e manipulação de dados.

3.9 Rotas da API

No contexto de API visto anteriormente, uma rota pode ser entendida como o caminho ou *URL* que um usuário utiliza para acessar um determinado recurso que é fornecido pela aquela API. As rotas carregam uma função muito importante definindo como a interação entre sistema-usuário acontece através de solicitações HTTP que define a ação a ser executada pela rota. As rotas funcionam no contexto onde uma requisição é feita, processada pela API e é devolvida uma resposta, a requisição pode enviar uma carga de dados, chamada de *payload*, que será utilizada no processamento da ação solicitada.

Tabela 2: Recursos disponíveis na API

Tipo de Método	**Rota**	**Descrição da Operação**
GET	/list	Recupera todos os produtos do inventário.
GET	/departament	Traz o departamento que o item se encontra.
POST	/calcule-dijkstra	Cria a matriz de menor distância dijkstra.
POST	/recommendation	Cria uma lista de recomendações com 10 produtos por ordem de pontuação.

Fonte: Autoria Própria

A Tabela 2 apresenta os recursos que podem ser acessados através da API, bem como o método que deve ser utilizado e uma descrição da operação a ser realizada.

3.10 Protótipo da Aplicação Mobile

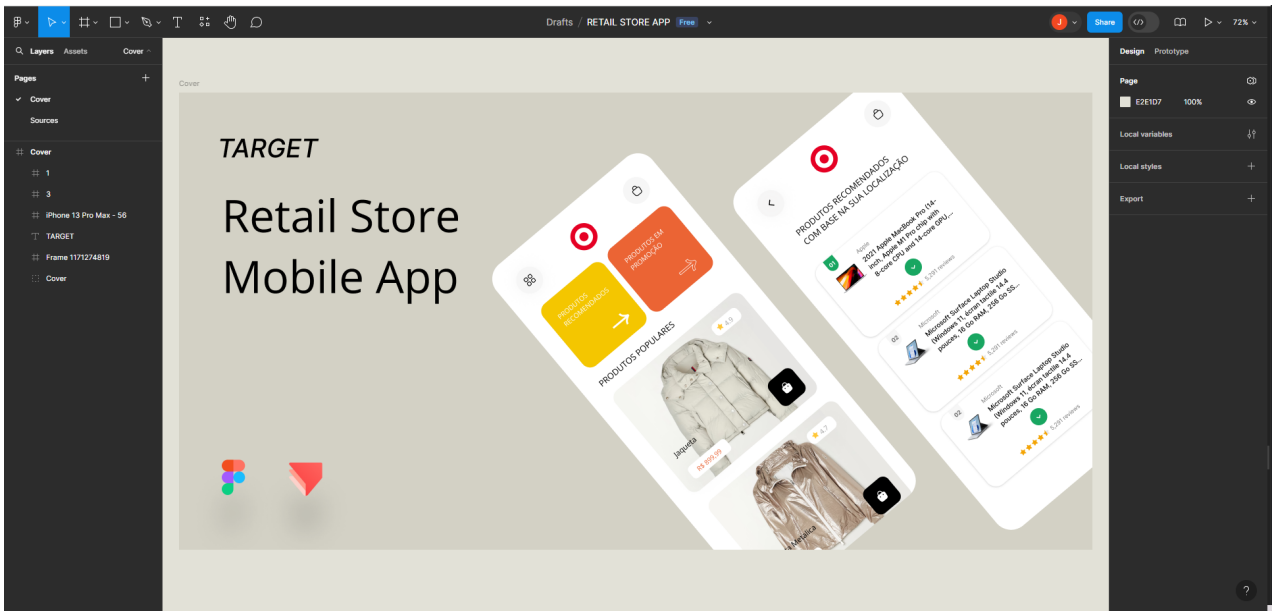
Para melhor representar o fluxo que um cliente vai utilizar usando uma aplicação *mobile* em seu dispositivo, foi desenvolvido um protótipo de um aplicativo, usando a

ferramenta *Figma*.

O Figma é uma ferramenta poderosa para criação de projetos de design web, onde uma comunidade pode interagir e colaborar na criação de protótipos, muitos desse sem nenhum custo, sendo criação de telas, manipulações totais de texto e imagens e todo tipo de ferramenta de edição para possibilitar ao desenvolvedor uma experiência completa(FIGMA, 2024).

O usuário ao acessar a aplicação terá a opção de visualizar recomendação de produtos personalizados para ele baseado em sua localização atual na loja. Quando ele permite a utilização de sua localização, a tela mostrará uma lista de produtos trazidos pela recomendação com uma pontuação especifica e ao clicar no produto além de conseguir visualizar detalhes sobre aquele produto, ele também consegue saber em que departamento ele se encontra, e assim pode ir de encontro ao produto, deixando a probabilidade de uma recomendação bem sucedida muito maior. Na Figura 12 podemos visualizar a ferramenta em uso para criar o protótipo proposto.

Figura 12: Construção no Figma



Fonte: Autoria Própria

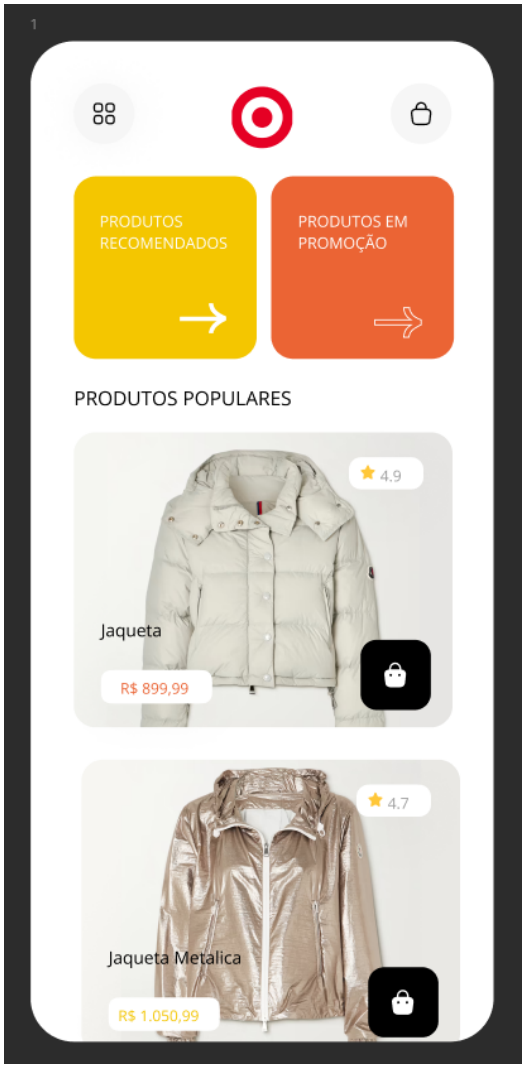
4 RESULTADOS E DISCUSSÕES

Nesse capítulo serão mostrados os resultados obtidos no projeto utilizando a metodologia vista anteriormente. No primeiro momento será mostrado como ficou o resultado do protótipo da aplicação mobile e depois o resultado do sistema de recomendação com localização que foi desenvolvido e a comparação de resultados usando o algoritmo com a abordagem da localização e o clássico.

4.1 Resultado do Protótipo

Na Figura 13 está a tela inicial do protótipo, onde é apresentado algumas funcionalidades como os produtos populares, um botão para os em promoção e, principalmente, os produtos recomendados.

Figura 13: Tela inicial do protótipo



Fonte: Autoria Própria

Na segunda tela, após o botão para os produtos recomendados ser pressionado, o usuário vai se deparar com uma mensagem solicitando o acesso a sua localização, que é um comportamento padrão visto em aplicações *mobile*, com a justificativa que receberá produtos recomendados, descontos e promoções como maneira de incentivar o cliente a acessar essa funcionalidade.

Essa etapa é muito importante, pois para os produtos serem recomendados de forma personalizada visando o contexto da loja física, é necessário que a localização seja um dos dados de entrada. Após a confirmação do uso da localização, o endpoint de criar as recomendações será chamado para trazer a lista ordenada das melhores recomendações.

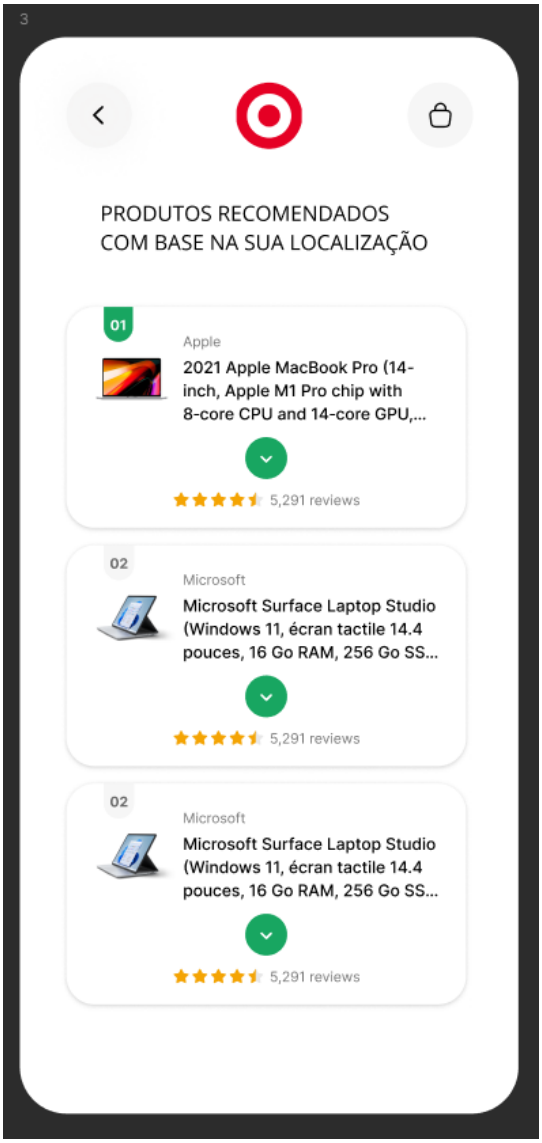
Figura 14: Segunda tela do protótipo

Fonte: Autoria Própria

Na terceira tela podemos ver no protótipo o retorno de uma lista de recomendações,

onde ele pode visualizar o nome completo do produto, uma imagem e a avaliação daquele item, como também um botão para ver detalhes daquele produto.

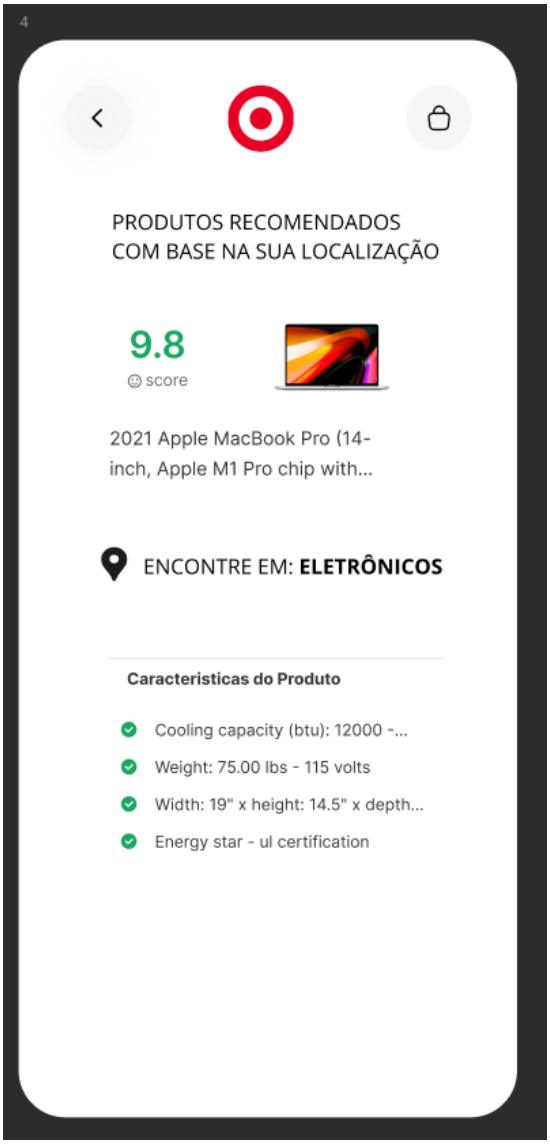
Figura 15: Terceira tela do protótipo



Fonte: Autoria Própria

Na quarta tela, tem uma descrição detalhada do produto, onde é possível ver a pontuação de recomendação normalizada entre 0 e 10 para mostrar ao usuário quão aquele item é forte para ele, e principalmente é possível saber a localização na loja daquele produto, assim aumentando a probabilidade de ir de encontro a loja, e também é visto algumas informações gerais do produto.

Figura 16: Quarta tela do protótipo



Fonte: Autoria Própria

4.2 Resultado da recomendação com localização

Nessa sessão são analisados os resultados referentes a implementação de um sistema de recomendação utilizando a localização do usuário em uma loja física de produtos onde ele já possui um histórico de compras.

Nas tabelas a seguir vemos o retorno da API, onde ele cria uma lista de dez recomendações, que é atrelado um identificador único do item junto a uma pontuação, essa lista está ordenada pelo valor dessa pontuação, sendo a maior a recomendação com mais força, ou seja, é um produto com a maior probabilidade de interesse para aquele usuário, levando em conta o cálculo do algoritmo de CF, que usa a matriz de similaridade

junto com a localização atual do usuário para gerar uma matriz de menor distância, que juntos retornam a recomendação mais personalizada possível para aquele cliente. A Tabela 3 mostra o retorno da API do algoritmo clássico sem a abordagem da localização.

Tabela 3: Retorno da API de uma recomendação sem localização

Identificador Único do Produto	**Pontuação**
15308b51-d3f3-57a6-b77a-1e71a1d27c80	213.08
1cafcd4e-e8b0-56fd-9a7d-c0b1b2a1a1e3	213.08
13030b5d-654b-5854-90d3-a7224c8294c7	162.63
0cb7c5df-d981-5ea1-8c05-df6b5617823f	131.79
17b73104-003c-5860-8334-692ad322761b	131.79
4cdeffac-a321-5530-b614-c67b6eb3d62f	121.87
55fb4af6-6bf2-5bca-8d2f-3db117c49d06	121.87
a8f069da-a30f-58b3-ab12-8429a2abe937	121.04
ff1752aa-6003-5cdb-aea9-eb74c08b6c98	111.02
dfb3bb0c-e562-5e38-8f68-147a456fa974	110.45

Fonte: Autoria Própria

Para verificar se houve e como foi a mudança do comportamento do algoritmo modificado para a versão clássica, foi utilizado os mesmos dados de entrada, mudando apenas o uso da localização, e foi percebido um aumento claro na pontuação observado da Tabela 3 para a Tabela 4, confirmando que o uso da localização promoveu uma melhora nesse recorte.

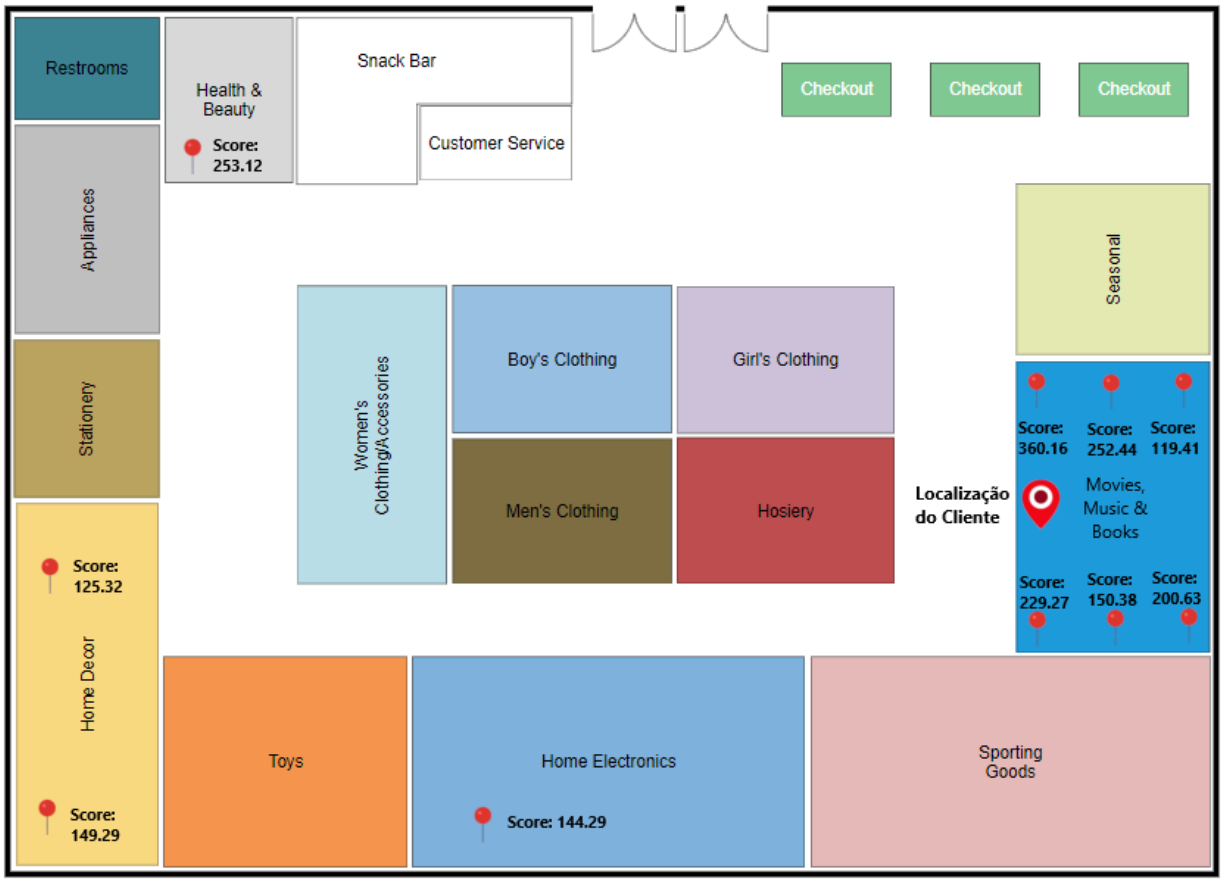
Tabela 4: Retorno da API de uma recomendação com localização

Identificador Único do Produto	**Pontuação**
15308b51-d3f3-57a6-b77a-1e71a1d27c80	360.16
1cafcd4e-e8b0-56fd-9a7d-c0b1b2a1a1e3	253.12
13030b5d-654b-5854-90d3-a7224c8294c7	252.44
ff1752aa-6003-5cdb-aea9-eb74c08b6c98	229.27
a8f069da-a30f-58b3-ab12-8429a2abe937	200.63
0cb7c5df-d981-5ea1-8c05-df6b5617823f	150.38
4cdeffac-a321-5530-b614-c67b6eb3d62f	149.29
dfb3bb0c-e562-5e38-8f68-147a456fa974	144.29
17b73104-003c-5860-8334-692ad322761b	125.32
fc1704c5-f1f9-5601-a08b-6cdc69701338	119.41

Fonte: Autoria Própria

Para melhorar visualizar o resultado da recomendação usando a localização do cliente, a Figura 17 mostra a representação dos produtos no mapa da loja, junto com a pontuação daquele item e em qual departamento o cliente está. Pode-se perceber que a recomendação mais forte vem de um item do mesmo departamento do cliente, o que mostra que a localização influência de forma direta aumentando a pontuação de um item similar ao histórico de compras, mas também não influência em todos os produtos da recomendação, como é visto que a segunda maior pontuação vem de um produto em departamento diferente, dessa forma, fica claro que o algoritmo apresenta um balanceamento entre a similaridade e a localização.

Figura 17: Visualização dos Resultados no Mapa



Fonte: Autoria Própria

Na próxima sessão será feito uma análise comparando os dois algoritmos com uma amostragem maior e mais ampla da base de dados para entender se esse comportamento visto acima se mantém em outros usuários e com outros produtos.

4.3 Comparação dos Resultados com o Algoritmo Clássico

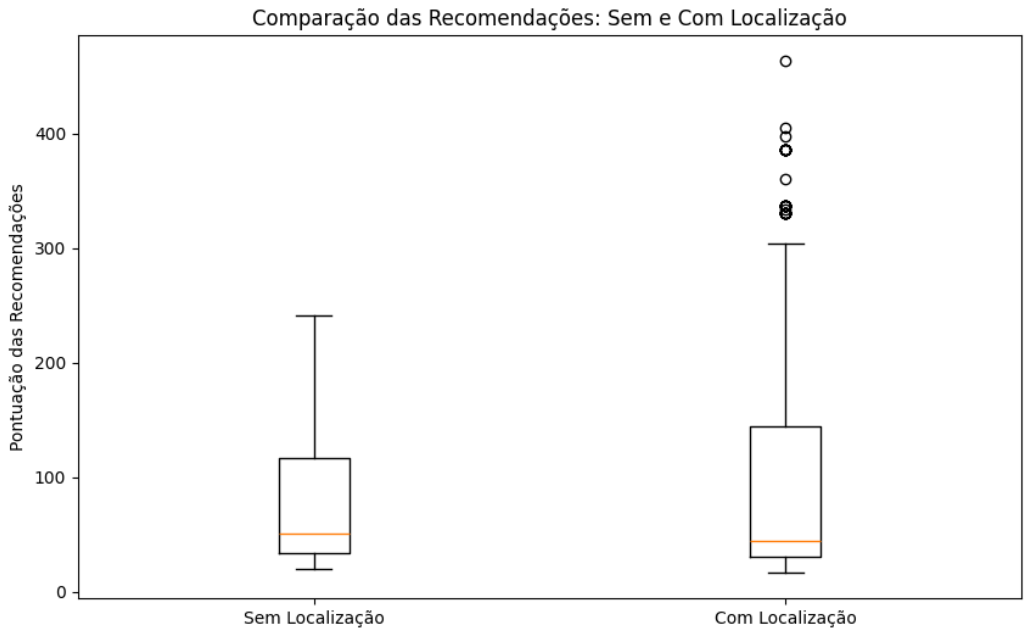
Nessa seção será mostrado uma comparação entre os resultados usando o algoritmo clássico do CF ao lado do algoritmo proposto nesse projeto com o uso da localização do usuário e entender esses resultados e se estão coerentes.

Primeiramente foi pensando na melhor forma de criar esse conjunto de testes, então foi definido que seriam utilizados os dez usuários com maiores registros na tabela do histórico de compras para se ter uma maior amostragem, bem como uma recomendação seria gerada para cada um dos 11 departamentos usados no projeto. Assim, foi criado um registro de mais de mil pontuações para os dois algoritmos.

Para melhor representar visualmente, foi criado um boxplot, que é uma representação gráfica de um conjunto de dados em cinco principais pontos, sendo o mínimo, o primeiro quartil, mediana, terceiro quartil e o máximo. As extremidades da caixa representam os quartis, que divide em quatro partes iguais os dados, já as linhas fora da caixa mostram os valores máximos e mínimos, além da presença de outliers, que são valores atípicos que se encontram fora dessas linhas(MCGILL; LARSEN, 1978).

O objetivo dessa comparação era visualizar se os dados criados pelo algoritmo modificado tinham o mesmo comportamento do clássico e se existiu alguma melhora em um mesmo cenário quando a abordagem da localização era inserida.

Figura 18: Boxplot da Comparação



Fonte: Autoria Própria

Analisando a Figura 18, vemos que o fator localização aumentou a pontuação de recomendação, principalmente a partir do terceiro quartil até seu valor máximo, além do aumento da média de forma geral, aparecendo inclusive outliers, que nesse cenário são explicados pois alguns itens além de apresentarem uma grande similaridade para o usuário, também estão muito próximo a sua localização, como por exemplo, no seu mesmo departamento, o que aumenta bastante a pontuação da recomendação, criando um item com bastante força, ou seja, esse produto é bastante personalizado e com grande probabilidade de compra do cliente.

Tabela 5: Comparação Numérica

	Algoritmo Clássico	**Algoritmo CF com Localização**
Maior Valor	241.8	463.8
Média	76.0	88.2
Menor Valor	19.4	16.6

Fonte: Autoria Própria

Os seguintes dados da Tabela 5 nos possibilita observar uma melhora factível no uso da localização para gerar recomendações, visto um crescimento de 16% na média geral da pontuação, valor que fica ainda maior na amostragem a partir do terceiro quartil. O menor valor do CF modificado estar mais baixo que o clássico é explicado que existe um impacto para um produto que além de ser pouco similar também está muito longe do usuário, o que deixa a pontuação mais baixa, o que é coerente, pois um produto longe fica menos atrativo para ele.

Assim, vemos um aumento significante ao valor máximo, pois entra no cenário explicado anterior dos resultados outliers, que a pessoa se encontra numa situação muito propícia, que é um histórico de compras que vai gerar uma grande similaridade aliado ao fato que a localização do produto coincide com a dele, gerando essa alta pontuação. O resultado mais interessante é observar que a localização está impactando positivamente a experiência do usuário, criando recomendações personalizadas para o contexto de uma loja física de produtos, além de gerar uma maior taxa de conversão de venda.

4.4 Mitigação do problema de *Cold Start*

Um das colaborações positivas observadas é que no cenário de um cliente que não possui registros na base de dados, a abordagem de usar a localização gera uma mitigação inicial do problema de *cold start*, um dos mais famosos do algoritmo clássico, visto que o sistema conseguirá prover uma lista de recomendação levando em conta pelo menos um aspecto conhecido do usuário que é a sua localização, ou seja, os produtos indicados vão ter uma personalização, mesmo sem histórico anterior de suas compras.

5 CONCLUSÕES E TRABALHOS FUTUROS

Esse trabalho foi feito com o objetivo de desenvolver um sistema de recomendação utilizando a localização de um usuário em um contexto de ambiente fechado de uma loja de produtos. Diante disso, os objetivos definidos foram alcançados, visto que a implementação teve um resultado positivo.

Foi criado com sucesso um protótipo de uma aplicação *mobile* para melhor representar a jornada do usuário em seu dispositivo até a lista de recomendação de produtos personalizados. Além de criar uma lista adequada de itens recomendados, também foi possível ver uma melhora na média geral da pontuação de recomendação e principalmente um aumento considerável em cenários que o usuário está muito próximo do produto, o que permite gera um maior grau de personalização para a experiência do cliente, bem como um aumento na probabilidade que um produto seja vendido, que era um dos desejos com o desenvolvimento desse projeto. Também é importante salientar que um dos problemas mais vistos do algoritmo *Collaborative Filtering* clássico é o cenário de *cold start*, que com o uso da abordagem de localização num contexto de ambiente fechado, mitigou esse problema, pois forneceu um dado de entrada sobre o usuário novo, permitindo a criação de recomendações mais personalizadas.

Para trabalhos futuros e melhorias do projeto, seria interessante aplicar o que foi desenvolvido em um cenário real de uma loja, utilizando seu espaço físico bem como sua base de dados, para conseguir ter uma visualização mais ampla do cenário, onde seria possível fazer ajustes e melhorar ainda mais os resultados, pois os dados seriam mais completos. Também seria interessante aplicar modelos e ferramentas de inteligência artificial no algoritmo, como árvore de decisão e redes neurais, e realizar treinamento dos dados, possibilitando criar um modelo preditivo de *Machine Learning*, que deixaria o algoritmo ainda mais inteligente, possibilitando ainda melhores resultados de recomendação.

REFERÊNCIAS

CHEN YC., H. L. . T. T. A collaborative filtering recommendation system with dynamic time decay. 2021.

CORMEN T. H., L. C. E. R. R. L. . S. C. *Introduction to Algorithms*. 3nd edition. ed. [S.l.]: MIT Press, 2009. ISBN 9780262033848.

FIGMA. *About Us*. 2024. Disponível em: <https://www.figma.com/pt-br/about/>. Acesso em: 02 setembro. 2024.

FLASK-SQLALCHEMY. *Flask-SQLAlchemy*. 2024. Disponível em: <https://flask-sqlalchemy.palletsprojects.com/en/2.x/>. Acesso em: 17 setembro. 2024.

GRINBERG, M. *Flask Web Development*. 2nd edition. ed. [S.l.]: O'Reilly, 2018. ISBN 9781491991732.

IBM. *What is collaborative filtering?* 2024. Disponível em: <https://www.ibm.com/topics/collaborative-filtering>. Acesso em: 07 agosto. 2024.

KAGGLE. *Target store retail products dataset*. 2024. Disponível em: <https://www.kaggle.com/datasets/crawlfeeds/target-store-retail-products-dataset>. Acesso em: 24 janeiro. 2024.

KOLONKO, K. Performance comparison of the most popular relational and non-relational database management systems. 2018.

LIANGXING, Y.; AIHUA, D. Hybrid product recommender system for apparel retailing customers. In: *2010 WASE International Conference on Information Engineering*. [S.l.: s.n.], 2010. v. 1, p. 356–360.

MCGILL, J. W. T. R.; LARSEN, W. A. Variations of box plots. *The American Statistician*, ASA Website, v. 32, n. 1, p. 12–16, 1978.

MYSQL. *MySQL 8.0 Reference Manual*. 2024. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/>. Acesso em: 23 ago. 2024.

NAEEM, T. *Definição da API REST: O que é uma API REST (API RESTful)?* 2020. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em: 04 Maio de 2024.

ORACLE. *O que é o MySQL*. 2024. Disponível em: <https://www.oracle.com/br/mysql/what-is-mysql/>. Acesso em: 02 junho. 2024.

PYTHON. *Whats is Python?* 2021. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em: 28 Abril de 2024.

RESNICK, P. et al. Grouplens: an open architecture for collaborative filtering of netnews. 1994.

RICCI, F. et al. *Recommender Systems Handbook*. 1st edition. ed. [S.l.]: Springer-Verlag, 2010. ISBN 0387858199.

SMARTDRAW. *Super Store Layout*. 2024. Disponível em: <https://wcs.smartdraw.com/store-layout/examples/super-store-layout/>. Acesso em: 17 outubro. 2023.

SQLALCHEMY. *The Python SQL Toolkit and Object Relational Mapper*. 2024. Disponível em: <https://www.sqlalchemy.org/>. Acesso em: 13 agosto. 2024.

STRANG, G. *Introduction to Linear Algebra*. 3rd edition. ed. [S.l.]: Wellesley-Cambridge Press, 1993. ISBN 9780961408817.

VENKIS. *Dijkstra Algorithm*. 2024. Disponível em: <https://venkys.io/articles/details/dijkstra-algorithm>. Acesso em: 05 agosto. 2024.

YELP. *About Us*. 2024. Disponível em: <https://www.yelp.com/about>. Acesso em: 22 outubro. 2024.

YUCHEN, P.; WU, D. Personalized online-to-offline (o2o) service recommendation based on a novel frequent service-set network. *IEEE Systems Journal*, PP, p. 1–9, 04 2019.

ZHANG, D. et al. Localization technologies for indoor human tracking. In: *Proceedings of the 5th International Conference on Future Information Technology (FutureTech)*. [S.l.: s.n.], 2010. p. 1–6.