



Faculdade de Ciências Exatas e da Engenharia  
Licenciatura em Engenharia Informática

Projeto/Estágio  
2021/2022

# SISTEMA DE RECOMENDAÇÃO DE EVENTOS PARA O CLIENTE FINAL

<b>Aluno</b>	<b>Orientador</b>
Luís Gouveia	Filipe Quintal
nº 2045519	Carlos Faria

# Índice

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Sistemas de Recomendações . . . . .	2
1.2	Descrição do Problema . . . . .	3
1.3	Yacooba . . . . .	3
<b>2</b>	<b>Solução</b>	<b>4</b>
2.1	Decomposição do Problema . . . . .	4
2.1.1	Recomendação Não Personalizada . . . . .	4
2.1.2	Recomendação Semi-Personalizada . . . . .	5
2.1.3	Recomendação Personalizada . . . . .	5
2.2	Considerações Finais . . . . .	5
<b>3</b>	<b>Revisão de Literatura</b>	<b>7</b>
3.1	<i>The Cold Start Problem</i> [1] . . . . .	7
3.2	<i>Cosine Similarity</i> . . . . .	7
3.3	<i>Matrix Factorization</i> . . . . .	8
3.3.1	<i>Real Value and Binary Matrix Factorization</i> . . . . .	10
3.4	Sistemas Comerciais . . . . .	11
<b>4</b>	<b>Implementação</b>	<b>12</b>
4.1	Requisitos . . . . .	12
4.2	Ferramentas . . . . .	13
4.3	Processo de Desenvolvimento . . . . .	13
4.3.1	<i>Domain Driven Design</i> . . . . .	13
4.3.2	Cálculo do <i>Score/Heurística</i> . . . . .	13
4.3.3	Recomendação Semi-Personalizada . . . . .	15
4.3.4	Recomendação Não Personalizada . . . . .	16
4.3.5	Recomendação Personalizada . . . . .	17
<b>5</b>	<b>Avaliação</b>	<b>19</b>
5.1	Metodologia de Avaliação . . . . .	19
5.2	Resultados . . . . .	19
<b>6</b>	<b>Discussão</b>	<b>22</b>
<b>7</b>	<b>Conclusão</b>	<b>23</b>
<b>8</b>	<b>Bibliografia</b>	<b>24</b>

# 1. Introdução

Nesta secção vamos apresentar o que são sistemas de recomendações e como estes são utilizados, será descrito o problema geral no qual este projeto se baseia que será a criação de um sistema de recomendações de eventos e haverá uma subsecção dedicada à empresa onde este projeto será realizado.

## 1.1 Sistemas de Recomendações

Com um aumento de plataformas e negócios *online* foi existindo uma maior necessidade de desenvolvimento de sistemas de recomendação de modo a melhorar a experiência do utilizador com conteúdo personalizado e mais relevante para o mesmo. Com estes sistemas de recomendações, serviços como a *Netflix*<sup>1</sup> e *Amazon*<sup>2</sup>, através do *feedback* de utilizadores, conseguem fornecer aos seus utilizadores sugestões relevantes sobre itens que estes poderão estar interessados, desde a recomendação de filmes na *Netflix* até à recomendação de produtos na *Amazon*.

Recomendações podem ser feitas com vários tipos de informação recolhidas do utilizador tal como o número de visualizações de uma certa página no *website*, as suas compras, o tempo que passa em cada página de um produto, o clique em links para expansão de descrição, ou até dados inseridos pelos utilizadores a indicar os seus gostos. Com estes dados as recomendações podem ser feitas tendo em conta várias abordagens, podem ser exploradas semelhanças entre itens (filmes, produtos e livros) tendo em conta vários aspetos que os caracterizam como, por exemplo, a categoria de um filme. Podem ser exploradas semelhanças entre utilizadores e itens onde são procuradas semelhanças entre, por exemplo, compras anteriores do utilizador, e através destas encontrar itens semelhantes que o utilizador talvez esteja interessado. Por fim também poderemos analisar semelhanças entre utilizadores, onde são procurados utilizadores semelhantes a um certo utilizador e recomenda-lo as diferenças entre os mesmos, por exemplo, o *utilizador\_1* comprou o *filme\_1* e *filme\_2*, e o *utilizador\_2* comprou o *filme\_1*, *filme\_2* e *filme\_3*, com esta informação vemos uma semelhança entre os dois utilizadores e podemos recomendar o *filme\_3* ao *utilizador\_1* devido à sua semelhança com o outro utilizador.

---

<sup>1</sup>*Netflix*: <https://www.netflix.com/pt/>

<sup>2</sup>*Amazon*: <https://amazon.es/>

## 1.2 Descrição do Problema

Neste projeto, será construído um sistema para a recomendação de eventos (ex.: *Summer Opening* e *Cinderella The Musical*). Serão feitos vários tipos de recomendações, uma recomendação com uma abordagem *Item-Item Collaborative Filtering* que será uma recomendação de eventos similares a um evento a ser apresentado na página do evento, abordagem esta que procurará semelhanças entre eventos para chegar a uma melhor recomendação. Será apresentada uma recomendação mais personalizada ao utilizador com uma abordagem parecida com *Item-Item Collaborative Filtering* onde as recomendações, ao contrário da abordagem anterior em vez de se serem vistas semelhanças entre um evento, são procuradas semelhanças entre os vários eventos previamente frequentados pelos utilizador, sendo assim recomendados eventos semelhantes a esses e também será feita uma recomendação de eventos geral tendo em conta a popularidade dos mesmos de modo a fornecer uma recomendação, mesmo que o utilizador não esteja autenticado.

## 1.3 Yacooba

O projeto será realizado na *Yacooba*, empresa esta que desenvolve um novo modelo de redistribuição de bilhetes para promotores de eventos e viajantes em todo o mundo. Esta mistura um protocolo de emissão de bilhetes na *blockchain* com o *one-click-buy* do mercado de compra de viagens. Este serviço permite a promotores de eventos submeterem os seus eventos na plataforma e permite aos seus utilizadores a compra dos mesmos, juntamente com os bilhetes de avião e estadia aos preços mais baixos tendo em conta a localização do evento.

O sistema de recomendações criado neste projeto será integrado no sistema da *Yacooba* na página dos eventos, recomendando eventos semelhantes ao evento da página e serão colocadas recomendações personalizadas e não personalizadas na página inicial dependendo se o utilizador está autenticado ou não no sistema, caso esteja, recomenda eventos tendo em conta a sua atividade. A empresa tem como objetivo, com estas recomendações, fornecer ao utilizador uma melhor experiência de utilização do sistema.

## 2. Solução

De modo a abordar o problema descrito acima, será feita uma divisão do mesmo em pequenas partes visto que cada uma dessas partes poderá recorrer a um método diferente para a sua resolução. Estes três principais objetivos serão o desenvolvimento da recomendação *personalizada*, *semi-personalizada* e *não personalizada* visto que são, de forma geral, os três principais tipo de recomendações. Estes serão utilizados para a *landing page* do sistema onde o utilizador não está autenticado, e quando este estiver autenticado será usada uma recomendação *personalizada*. Quanto à recomendação *semi-personalizada* será usada para a recomendação de eventos semelhantes de modo a serem colocadas dentro da página de cada evento.

### 2.1 Decomposição do Problema

Para a divisão do problema foi tido em conta o nível de personalização de cada recomendação. Na figura 2.1 podemos verificar a divisão separando a recomendação em três níveis, *não personalizado*, *semi-personalizado* e *personalizado*. Uma classificação semelhante a esta pode ser consultada em [1].

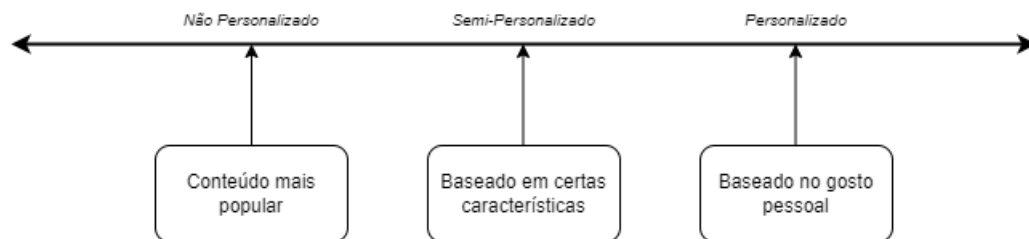


Figura 2.1: Níveis de Personalização

#### 2.1.1 Recomendação Não Personalizada

Com um baixo nível de personalização na recomendação, será usada informação geral sobre eventos. Essa informação poderá incluir algo como as datas (*ex.: eventos mais recentes*), número de compras de cada bilhete ou até por um *ratio* de bilhetes vendidos para total de bilhetes disponíveis. Em poucas palavras, com um nível baixo de personalização como este, a recomendação de eventos terá em conta a popularidade dos mesmos.

Para o propósito deste projeto o que irá ditar a popularidade de um evento será a quantidade de visualizações obtidas na página de um evento, ou seja, quantas mais visualizações o evento tiver mais será recomendado. Neste projeto este tipo

de recomendação será utilizado para a recomendação de eventos para a *Landing Page* do sistema quanto o utilizador não está autenticado no mesmo.

### 2.1.2 Recomendação Semi-Personalizada

Para um nível intermédio de personalização terá de ser tida em conta mais informação do que o número de visualizações da página de um evento, será necessário ter em conta a semelhança entre eventos de modo a ter em conta certas características dos mesmos. De modo a conseguir este tipo de personalização primeiro terão de ser analisadas que informações de cada evento serão utilizadas para atingir o objetivo desejado.

Foi feita a análise de toda a informação que é armazenada sobre cada evento, esta análise foi feita de modo a descartar toda a informação que não seria relevante para a recomendação. Após alguma análise dos dados relativos a cada evento chegamos à conclusão que as informações mais relevantes para a recomendação seriam a localização do mesmo, *ratio* de quantidade de bilhetes vendidos para a quantidade de bilhetes disponíveis, tipo de evento e *keywords* relacionadas com o mesmo. Toda esta informação, para ser utilizada, terá de ser processada e aglomerada de forma quantitativa de modo a saber quais eventos a recomendar. Para isso foi decidido que seria dado um *score* a cada evento.

Este tipo de recomendação com este nível de personalização será usada para recomendar eventos similares dentro da página de um evento. Com essa finalidade em mente todos os eventos futuros serão comparados ao evento da página, calculando o *score* de cada um desses eventos em relação ao evento da página, por fim serão recomendados os eventos com um maior *score*.

### 2.1.3 Recomendação Personalizada

O terceiro e último nível de recomendação terá um nível de personalização alto onde todas as recomendações terão em conta os gostos do utilizador. A fim de "descobrir" os gostos do utilizador serão tidas em conta compras de bilhetes para eventos passados. À semelhança da recomendação semi-personalizada será calculado um *score* para os eventos, que é calculado fornecendo dois eventos, um dos eventos cujo utilizador comprou bilhetes e o evento a ser comparado com esse que receberá um *score*. Esse *score* será calculado para todos os eventos comparando-os com os eventos cujo utilizador comprou bilhetes obtendo assim os eventos com maior semelhança com os com os bilhetes dos eventos que o utilizador comprou no passado.

## 2.2 Considerações Finais

Tendo em contas os requisitos que serão mencionados nas secções seguintes, temos como objetivo a recomendação para a *landing page* do sistema e para a página de cada evento. Cada uma destas recomendações recorrerá a uma abordagem diferente já mencionada acima (*personalizada*, *semi-personalizada* e *não personalizada*). Tudo isto será resolvido com o calculo de um *score/heurística* para obter eventos semelhantes a um certo evento, que será utilizado para a recomendação dentro da página de cada evento, e para obter eventos semelhantes a eventos cujo

utilizador comprou bilhetes no passado, para obter as recomendações para a *landing page* com o utilizador autenticado. Caso este não esteja autenticado, será tomada uma abordagem *não personalizada* recorrendo ao número de visualizações da página de cada evento com recurso a uma plataforma de *analytics*, no caso, *Google Analytics*.

As próximas secções apresentam métodos que poderiam ter sido utilizados em substituição à solução apresentada acima e será apresentada a implementação da solução aqui apresentada.

## 3. Revisão de Literatura

A abordagem tomada na solução acima apresentada, utiliza o calculo de um *score/heurística* para a recomendação de eventos. Existem muitas outras soluções que foram exploradas neste projeto que veremos abaixo. Estas soluções não foram utilizadas devido ao problema conhecido por *The Cold Start Problem*[1]. No entanto de modo a enriquecer a discussão iremos apresentar as várias soluções, *Cosine Similarity* e *Matrix Factorization*, pois são ótimas soluções usadas em múltiplos sistemas comerciais e serão soluções que puderam vir a ser utilizadas quando o *The Cold Start Problem*[1] deixar de ser um problema na implementação de uma solução.

### 3.1 *The Cold Start Problem*[1]

Se não tivermos conhecimento dos utilizadores não poderemos ter algum tipo de recomendação personalizada, algo que poderá afetar a experiência de utilizadores fazendo com que não voltem a utilizar a plataforma.

O *The Cold Start Problem*[1] é o termo usado para recomendações para novos utilizadores ou para recomendações que novos itens que poderão não aparecer em algumas estatísticas pois os utilizadores ainda não tiveram tempo para interagir com os mesmos. Estas recomendações serão um problema pois não existe informação para fazê-las e não existe o conhecimento dos utilizadores para as tais recomendações mais personalizadas.

Devido a este problema as alternativas de solução apresentadas neste capítulo não poderiam ser realizadas pois a recomendação de eventos será feita para um novo sistema onde ainda não existem dados dos utilizadores. Após a recolha de mais dados dos utilizadores poderão ser utilizadas estas soluções de modo a que o utilizador tenha uma experiência cada vez mais personalizada no sistema.

### 3.2 *Cosine Similarity*

Similaridade por cosseno ou *Cosine Similarity* [2] [3] permite a medição, como o nome indica, da similaridade entre 2 vetores. Esta usa o cosseno entre dois vetores permitindo verificar se estes estão a "apontar" para a mesma direção.

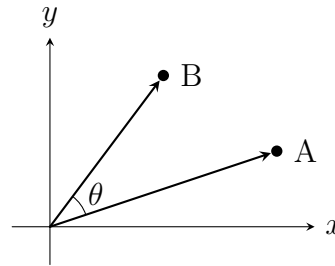
Este método, geralmente usado para análise de texto usando a semelhança do mesmo, poderia ser utilizado para as recomendações de eventos. Visto que a sua grande utilização é na análise de texto poderiam ser usadas informações textuais dos eventos, tais como a descrição, *keywords*, tipo de evento e o título do mesmo. Com isto utilizava-mos uma abordagem *Item-Item Content-Based Filtering*[4] onde



a recomendação seria feita tendo em conta a semelhança entre itens que neste caso seriam eventos.

Para a implementação deste método, primeiro seria necessário um pré-processamento da informação onde toda a informação que tivesse os valores *NaN*, *null* ou *undefined* seria substituída por uma *string* vazia de modo a não gerar erros durante a execução de código. Para concluir o pré-processamento, toda a informação teria de ser combinada num único bloco permitindo que a similaridade por cosseno possa apresentar um valor para a semelhança dos mesmos.

Após os passos acima pode ser utilizada a similaridade por cosseno. Em 3.1 verificamos a equação para a utilização deste método.



$$\cos(\theta) = \frac{A \cdot B}{\|A\| \times \|B\|} \quad (3.1)$$

Para a realização de todos os cálculos seria utilizada a linguagem *Python*[5] com a biblioteca *sklearn*[6], tal como as bibliotecas *pandas*[7] e *numpy*[8] para o processamento mais fácil dos dados.

É de notar que os resultados da similaridade entre os blocos de texto seriam apresentadas entre 0 e 1, onde 1 significa que os blocos são 100% iguais. Logo os eventos mais recomendados seriam os que tivessem um valor mais próximo de 1 em relação aos eventos cujos bilhetes o utilizador comprou no passado.

### 3.3 *Matrix Factorization*

*Matrix Factorization*[4] é um dos métodos usados em sistemas de recomendações, geralmente quando é utilizada uma abordagem *collaborative filtering* [1][4] (procura de utilizadores semelhantes ao utilizador principal de modo a recomendar itens). Este método decompõe uma matriz com a interação dos utilizadores com os itens em duas matrizes mais pequenas. Este método foi bastante conhecido no *Netflix Prize Challenge*[1].

Vejamos agora a matriz abaixo, esta representa a interação dos utilizadores com os itens, que neste caso serão eventos. Cada linha representa um utilizador e cada coluna um evento.

$$\begin{bmatrix} ? & 4 & ? & 0 & 1 \\ 3 & 1 & 2 & ? & ? \\ ? & 2 & 3 & 1 & ? \\ ? & 0 & ? & 4 & ? \end{bmatrix} \quad (3.2)$$

Os valores apresentados serão avaliações de 0 a 5 dadas pelos utilizadores a cada evento, no caso de não existir nenhum *feedback* por parte do utilizador são colocados pontos de interrogação (?). Esta informação nunca será muito completa,

estará carregada de ? pois os utilizadores apenas fizeram *review* de um número pequeno de eventos, o que nos leva ao problema de termos de prever os valores desconhecidos.

Uma forma de conseguirmos obter esses valores é atribuindo um valor a cada tipo de atributo de um evento, no caso o tipo de evento.

Teremos duas matrizes, uma delas representará os gostos de cada utilizador em cada tipo de evento (3.3), cada linha representará um utilizador e cada coluna um tipo de evento. A segunda matriz representará o quanto cada evento está associado a cada categoria(3.4), onde cada coluna será um evento e cada linha um tipo de evento.

De seguida podemos realizar a multiplicação das duas matrizes e normalizar os valores obtidos de modo a obter uma previsão para cada evento (3.5).

$$\begin{bmatrix} 3 & 0 \\ 2 & 2 \\ 4 & 4 \\ 0 & 4 \end{bmatrix} \quad (3.3)$$

$$\begin{bmatrix} 1 & 0 & 4 & 1 & 4 \\ 3 & 4 & 3 & 2 & 0 \end{bmatrix} \quad (3.4)$$

$$\begin{bmatrix} 0 & 0 & 1.5 & 0 & 1.5 \\ 1 & 1 & 2 & 1 & 1 \\ 2 & 2 & 3.5 & 3 & 2 \\ 1.5 & 2 & 1.5 & 1 & 0 \end{bmatrix} \quad (3.5)$$

O problema com esta abordagem é que não existem apenas dois tipos de eventos, logo um melhoramento seria adicionar mais tipos de eventos, mas para tal teríamos de questionar todos os utilizadores sobre as suas preferências quanto a cada tipo de evento, algo que a *Netflix* chegou a realizar no passado (Fig. 3.1).

Isto leva a outros problemas pois realizar um grande questionário para perceber as preferências de um utilizador iria afasta-lo da plataforma e, por vezes, os utilizadores têm dificuldade em expressar o que realmente preferem.

*Matrix Factorization* e *Machine Learning* ajudam a resolver esse problema. Primeiro começamos com a matriz com os dados que tínhamos dos utilizadores (3.2), e através dela obtemos os valores das outras matrizes que relacionam os itens e utilizadores aos tipos de evento, ou seja, simplesmente revertemos a abordagem anterior. O que o algoritmo de *machine learning* fará é tentar "adivinhar" os valores das matrizes mais pequenas e fica com os valores que geram uma matriz principal com valores muito próximos àqueles que já sabíamos.

Com os valores o mais próximo possível dos dados que já tínhamos os pontos de interrogação ficariam preenchidos com um valor que nos indicará se o utilizador irá gostar do evento, permitindo-nos recomendar eventos baseados nesses valores.

$$\begin{bmatrix} 0 & 4 & 2 & 0 & 1 \\ 3 & 1 & 2 & 1 & 2 \\ 8 & 2 & 3 & 1 & 4 \\ 0 & 0 & 4 & 4 & 0 \end{bmatrix} \quad (3.6)$$

In general, how much do you like watching movies from the following genres?						
	Really dislike	Dislike	Neither like nor dislike	Like	Really like	Not sure of genre definition
Action	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Adventure	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Animation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Comedy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Crime/Gangster	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Documentary	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Drama	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fantasy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Film-Noir	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Foreign	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Horror	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Indie	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Musical	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Mystery	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Romance	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Rom-Com	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sci-Fi	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sport	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Thriller	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
War	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figura 3.1: Inquérito sobre as preferências do utilizador

### 3.3.1 *Real Value and Binary Matrix Factorization*

Nos exemplos acima foram usados valores entre 0 e 5 que representariam *reviews* dos utilizadores nos eventos, essa abordagem seria *Real Value Matrix Factorization*[9]. No âmbito deste projeto essa abordagem não servirá pois os eventos não terão nenhum tipo de avaliação logo não seria possível obter uma avaliação quantitativa de 0 a 5 por exemplo, para um evento só tendo a informação de se o utilizador realizou a compra de bilhetes para o mesmo ou não. Tendo isso em conta foi encontrada a abordagem *Binary Matrix Factorization*[10] onde os valores seriam representados com 0 ou 1, onde 0 poderia representar “*não gostou*” e 1 “*gostou*”, algo que funcionaria num sistema de *likes*, algo que também não está presente no sistema. Apesar disso foi testado esse método, utilizando a linguagem *Python* na plataforma *Google Colab*[11], dando um valor de 1 a todos os eventos cujo utilizador comprou bilhetes, mas como esperado, não houve sucesso pois como não havia a presença de zeros, o algoritmo na sua previsão todos os valores iam tender para 1 (Fig. 3.2).

Em conclusão, apesar de estas serem duas ótimas abordagens, nenhuma delas foi utilizada devido ao facto do sistema sobre o qual o sistema de recomendações está a ser desenvolvido não apresenta nenhum sistema de *likes* ou avaliação em estrelas.

```
array([[1.06147181, 0.93998258, 1.12169077, 1.0652807 ],
       [0.90290006, 0.79955994, 0.954123 , 0.90613995],
       [0.99017233, 0.87684358, 1.04634636, 0.99372538],
       [0.97486656, 0.86328961, 1.03017227, 0.97836469],
       [0.87699056, 0.77661587, 0.92674361, 0.88013747]])
```

Figura 3.2: Output de previsão recorrendo a *Binary Matrix Factorization*

### 3.4 Sistemas Comerciais

Empresas como a *Amazon* e *Netflix* utilizam sistemas de recomendações nas suas plataformas de modo a melhorar a experiência do utilizador.

No caso da *Amazon* esta tem em conta o histórico de compras dos utilizadores e utiliza algoritmos envolvendo *machine learning* e *inteligência artificial* de modo a obter uma boa recomendação tendo em conta esses dados. Não são públicos os algoritmos utilizados pela mesma, como foi visto aqui [12] e [13] esta utiliza algo derivado da *Matrix Factorization* e utilizam a sua própria plataforma *A9* para complementar essa recomendação. É de notar que esta tem um próprio sistema que fornece recomendações tendo em conta *datasets* fornecidos, este serviço está presente na *AWS* e é chamado *AWS Personalize*[14].

Quanto à *Netflix* também não são públicos os detalhes de implementação, mas no geral, e como visto em [1], [15] e [16], são recolhidos todos os dados possíveis da interação que o utilizador teve com o sistema, desde histórico de pesquisa a *scrolls* num *carousel* horizontal(ex.: [17]). Essa informação será processada também utilizando *machine learning* e *inteligência artificial* de modo a obter as recomendações.

## 4. Implementação

Após a decomposição do problema, revisão da literatura e descoberta de uma solução foi passada à parte da implementação da mesma já discutida em secções anteriores. Nesta secção serão listados os requisitos de baixo nível que terão de ser cumpridos para que a implementação corresponda ao delineado na solução proposta (*secção 2*), as ferramentas que serão utilizadas e como foi o processo de desenvolvimento e evolução do protótipo ao longo da implementação da solução previamente explicada.

### 4.1 Requisitos

Bons requisitos permitem a criação de uma boa base para onde começar um projeto, ajudando a perceber o que poderá faltar implementar, organizar o processo de desenvolvimento e ajudar a perceber qual será a melhor ordem de implementação de cada uma das funcionalidades do projeto. Tendo em conta os objetivos da *Yacoooba* e os requisitos gerais de qualquer sistema de recomendações, abaixo estão apresentados os requisitos recolhidos para este projeto:

- R1: O sistema deverá recomendar 4 eventos semelhantes a um certo evento.
- R2: O sistema deverá recomendar 5 eventos para a *Landing Page*.
- R3: O sistema deverá disponibilizar recomendações para a *Landing Page* tendo em conta compras passadas do utilizador.
- R4: O sistema deverá utilizar o *Google Analytics* para a recolha de informação sobre as visualizações das páginas de eventos.
- R5: O sistema deverá persistir os dados vindos do *Google Analytics* utilizando *Redis*[18].
- R6: O sistema deverá utilizar a localização de cada evento para os vários tipo de recomendações.
- R7: O sistema deverá utilizar o tipo de cada evento para os vários tipo de recomendações.
- R8: O sistema deverá utilizar as *keywords* de cada evento para os vários tipo de recomendações.
- R9: O sistema deverá utilizar a quantidade de bilhetes vendidos de cada evento para os vários tipo de recomendações.

R10: O sistema deverá utilizar a quantidade de bilhetes disponível de cada evento para os vários tipo de recomendações.

R11: O sistema deverá identificar se utilizador está autenticado.

## 4.2 Ferramentas

Para o desenvolvimento deste sistema de recomendações foi utilizado o apoio de várias ferramentas, não só para as recomendações em si, mas para a implementação do mesmo no sistema já existente da *Yacooba*. Neste, de forma geral, para o desenvolvimento *backend* são utilizados *Typescript*[19], *GraphQL*[20] e *PostgreSQL*[21] e quanto ao *frontend* é utilizado *React*[22], *Next.js*[23] e *Material UI*[24].

Já quanto ao próprio sistema de recomendações será usado *Typescript*, a linguagem já utilizada pela empresa, e com esta será feita toda a lógica para o cálculo dos *scores/heurísticas* utilizadas para a recomendação. O *Google Analytics* será utilizado para a obtenção de estatísticas sobre a página de cada evento. Para a utilização dessa biblioteca com *Typescript* serão necessárias as bibliotecas *@google-analytics/data*[25] e *@types/google.analytics*[26]. Visto que, devido a limitações verificadas com o *Google Analytics* que serão analisadas no processo de desenvolvimento (secção 4.3), foi utilizado *Redis*[18] onde serão persistidos os dados recolhidos pelo *analytics* de modo a limitar a quantidade de chamadas à *API*.

## 4.3 Processo de Desenvolvimento

Como já abordado na secção *Solução* o problema será decomposto em 3 tipos de recomendação, *não personalizada*, *semi-personalizada* e *personalizada*. Para a implementação dessas soluções no *backend* da *Yacooba* foi utilizado o modelo arquitetural *Domain Driven Design*[27] já utilizado pela empresa.

### 4.3.1 *Domain Driven Design*

*Domain Driven Design* é uma abordagem ao desenvolvimento de *software* que permite transmitir um problema complexo em *software* expressivo e evolutivo. Com este modelo conseguimos abstrair a complexidade de negócio através de uma representação simplificada que separa o problema em vários blocos que serão usados para a construção do software, havendo uma arquitetura em camadas. Este modelo segue princípios de arquitetura descritos por *Robert Martin* no livro *Clean Architecture*[28].

### 4.3.2 Cálculo do *Score/Heurística*

Quer para as recomendações *semi-personalizadas*, quer para as recomendações *personalizadas*, será necessário uma forma de "avaliar" o quão recomendado um evento será. Para isso foi construído um método que calcula um *score/heurística*, algo que será usado nas recomendação e já mencionado na secção *Solução*. Para tal será tido em conta a localização dos eventos, o tipo, as *keywords* e o *ratio* de bilhetes vendidos para bilhetes disponíveis. Cada uma destas características será avaliada

de 0 a 10, sendo que 10 é quando as mesmas são 100% iguais, esses valores serão somados e teremos o *score* final para um evento.

Quanto ao tipo de evento, visto que este pode ter vários associados a este, iremos considerar a lista de tipos como um conjunto, e calcularemos a similaridade desse conjunto com o conjunto de tipos do evento com o qual queremos comparar. Podemos verificar em 4.2 a formula utilizada para tal. Algo semelhante será usado para comprar a lista de *keywords* dos eventos.

$$similarity = \frac{\#(A \cap B)}{\#A} \times 10, \text{ sendo A o conjunto principal e B o conjunto com o qual estamos a comparar} \quad (4.1)$$

Quanto ao *ratio* de bilhetes vendidos para bilhetes disponíveis pode ser calculado por uma simples divisão:

$$ratio = \frac{\#ticketsSold}{\#ticketSupply} \times 10 \quad (4.2)$$

Para a comparação da localização terá sido em conta a distância entre os dois locais no planeta Terra que será calculada tendo em conta as coordenadas das mesmas e para tal será utilizada a formula de *Haversine*[29]:

$$d = 2r \arcsin \left( \sqrt{\sin^2 \left( \frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left( \frac{\lambda_2 - \lambda_1}{2} \right)} \right)$$

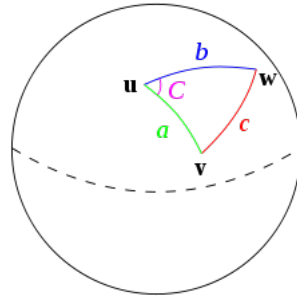


Figura 4.1: *Haversine Formula*

Esse cálculo será utilizado para calcular a distância entre a localização onde serão decorridos dois eventos fornecidos. Tendo essa distância calculada temos de normalizar os dados de modo a obter valores de 0 a 10, o que pode ser facilmente utilizando uma equação linear, onde 12756 é a máxima distância (em km) entre locais no planeta terra):

$$distance\_score = \frac{-10}{12756} \times distance + 10 \quad (4.3)$$

Por fim, tendo os cálculos todos feitos somamos todos os resultados e obtemos o *score* para um dado evento. Abaixo podemos verificar o *pseudo-código* a o cálculo do *score*:

---

```
function calculate_score (main_event: Event, other_event: Event){
    let score = 0;
    score += calculate_similarity(
        main_event.keywords, other_event.keywords
    );
    score += calculate_similarity(
        main_event.types, other_event.types
    );
    score += calculate_distance(
        main_event.location, other_event.location
    );
    score += calculate_ticket_ratio(
        main_event.tickets_sold,
        main_event.ticket_supply,
        other_event.tickets_sold,
        other_event.ticket_supply
    );
    return score;
}
```

---

### 4.3.3 Recomendação Semi-Personalizada

Para a recomendação *semi-personalizada* terá de ser processada toda a informação sobre cada evento para ser possível a recomendação. Esta recomendação será utilizada para a recomendação de eventos semelhantes a um evento para serem apresentados na página de um evento. Como mencionado na secção da *Solução* será realizado o cálculo de um *score/heurística*.

O sistema receberá um evento e irá buscar à base de dados todos os outros eventos futuros que estão disponíveis, após obter essa informação é percorrida a lista de eventos futuros e é calculado o *score* para cada um desses eventos em comparação ao evento da página para a qual as recomendações são direcionadas. Por fim, a lista é ordenada de forma decrescente de *score* e serão retornados os primeiros quatro eventos com um maior *score*, ou seja, os quatro eventos mais semelhantes ao evento fornecido.

Abaixo podemos verificar o *pseudo-código* para a realização desta recomendação e a implementação no *frontend* deste tipo de recomendação:



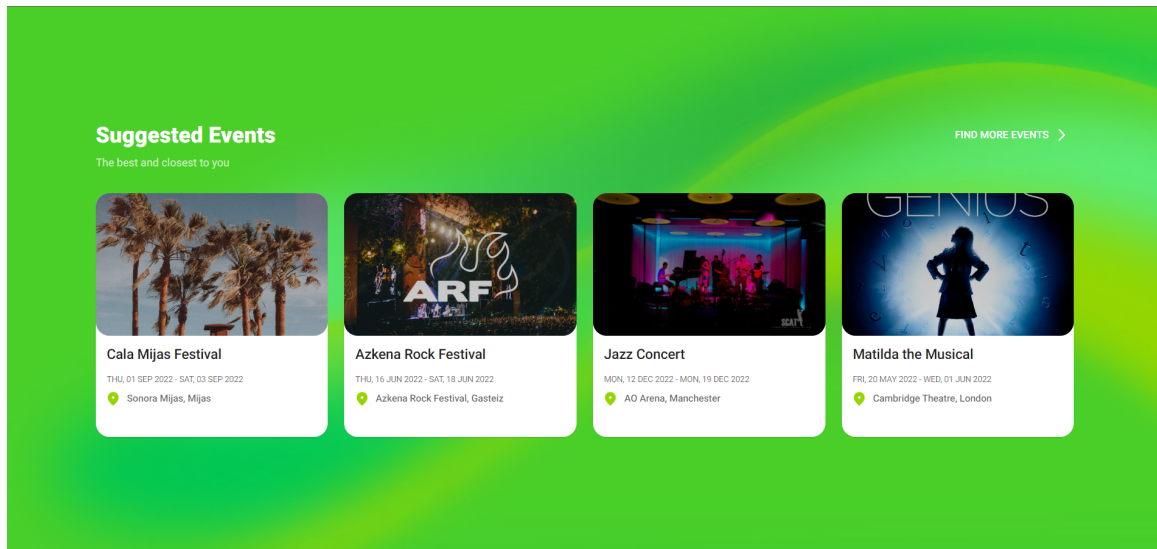


Figura 4.2: Eventos recomendados na página de cada evento

---

```
function semi_personalized_recommendation (event: Event) {
  const future_events: Event[] = getAllEvents();
  const events_with_score: Event[];
  for e in future_events {
    events_with_score.push({
      score: calculate_score(event, e),
      event: e,
    });
  }
  events_with_score.sort();
  return events_with_score.slice(0, 4);
}
```

---

#### 4.3.4 Recomendação Não Personalizada

De forma a obter recomendações sem muita informação do utilizador teremos de fazer uma recomendação não personalizada que será ditada pela popularidade de um evento que é decidida através das visualizações que a página do mesmo tem. De modo a obter essa informação utilizamos o *Google Analytics*[30] que permite a extração de várias informações, sendo a quantidade de visualizações por página uma delas. Para tal são necessárias algumas bibliotecas para a comunicação com a *API* do *Google Analytics*, entre elas a *@google-analytics/data*[25] e *@types/google.analytics*[26]. De forma a que não sejam considerados eventos passados, a *query* à *API* será feita com filtro para obter as informações só para os eventos enviados.

A *API* tem uma cota limite de quantidade de pedidos diários e por hora, sendo

que cada pedido poderá ter um maior peso nessa cota dependendo da complexidade do mesmo. Após alguma pesquisa e testes, concluímos que os valores de visualizações de páginas só são atualizados a cada 24 horas. Tendo isso em mente, o problema do limite da cota diária poderá ser resolvido de forma mais simples utilizando *Redis*, onde só será feita uma chamada à *API* por dia, e a informação extraída será guardada utilizando o *Redis* onde essa informação terá um *timeout* de 24 horas e a informação será atualizado no fim do mesmo. Em conclusão, será feita uma *query* à *API* do *Google Analytics* para obter o número de visualizações de cada página dos eventos, informação esta que será guardada com a ajuda da base de dados em memória do *Redis*.

Os eventos com um maior número de visualizações serão passados para o cliente sobre a forma de recomendações na *landing page* quando este não está autenticado no sistema.

Na Fig. 4.3 podemos ver a implementação no *frontend* dos eventos recomendados utilizando este tipo de recomendação. Abaixo também podemos verificar o *pseudo-código* para o descrito acima:

---

```
function non_personalized_recommendation () {
  if (redis.get("event_views")) {
    return redis.get("event_views");
  } else {
    const future_events: Event[] = getAllEvents();
    const event_views = getViewsGoogleAnalyticsAPI(future_events);
    redis.set("event_views", event_views.slice(0, 5));
    return event_views.slice(0, 5);
  }
}
```

---

#### 4.3.5 Recomendação Personalizada

Por fim, temos uma recomendação personalizada que terá em conta as compras passadas do utilizador, sendo um requisito o facto de utilizador estar autenticado no sistema. Em primeiro lugar terá de ser visto se o utilizador tem compras passadas, caso não as tenhas teremos de recorrer a uma recomendação não personalizada pois não teremos informação sobre o utilizador.

Caso este tenha compras passadas, serão obtidos todos os eventos cujo utilizador já comprou e, utilizando uma abordagem semelhante à recomendação *semi-personalizada*, esses eventos serão comparados com os eventos futuros, os que tiverem uma maior semelhança (um maior *score*) com os eventos cujo utilizador comprou bilhetes então esses serão os mais recomendados. A lista de eventos futuros será percorrida e cada um deles será comparado com cada um dos eventos cujo utilizador comprou e o maior desses *scores* será associado ao evento, depois os eventos com um maior *score* serão transmitidos ao utilizador sob a forma de recomendação na *landing page*. De seguida está apresentado o *pseudo-código* para a realização deste tipo de recomendações:

---

```

function personalized_recommendation (user: User){
  const bought_events: Event[] = getBoughtEvents(user.id);
  if (bought_events.isEmpty()) {
    return non_personalized_recommendation();
  } else {
    const future_events: Event[] = getAllEvents();
    const events_with_score: Event[];
    for e in future_events:
      const score_values = []
      for b in bought_events:
        score_values.push(calculate_score(b, e));
      events_with_score.push({
        score: score_values.max(),
        event: e,
      });
    events_with_score.sort();
    return events_with_score.slice(0, 4);
  }
}

```

---

Abaixo podemos ver o resultado da implementação no *frontend* dos eventos recomendados utilizando uma recomendação *não personalizada* e *personalizada*.

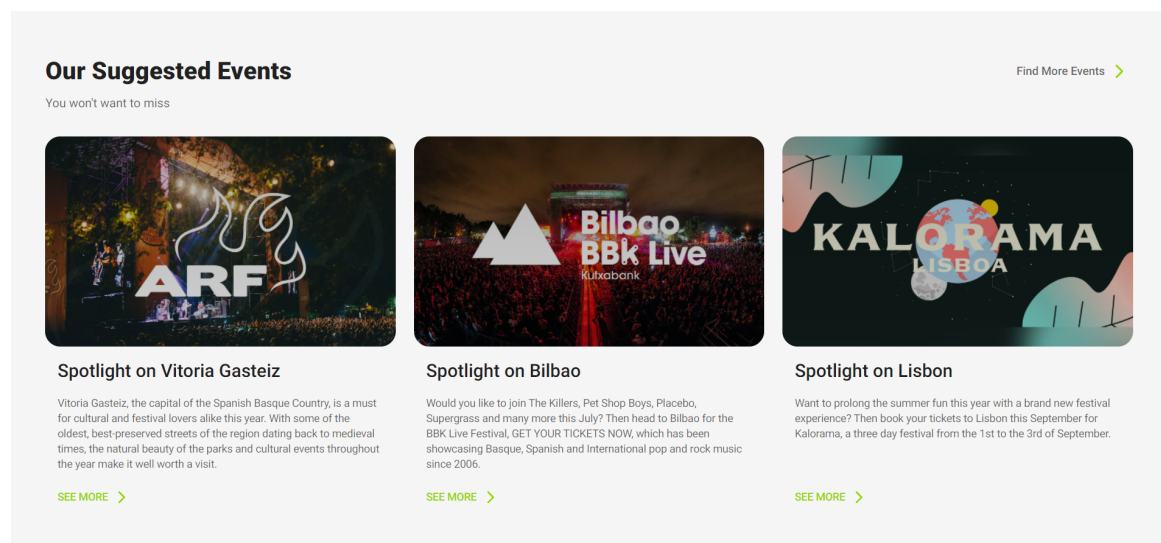


Figura 4.3: Eventos recomendados na *Landing Page*

## 5. Avaliação

O propósito de testar um sistema é confirmar que este vai ao encontro de todas as especificações e requisitos definidos no início e durante o desenvolvimento do sistema. Nesta secção será abordado como as funcionalidades foram testadas de modo a verificar se o sistema de recomendações está a gerar recomendações relevantes aos utilizadores.

### 5.1 Metodologia de Avaliação

A melhor forma de testar um sistema de recomendações é tendo *feedback* de utilizadores pois serão estes que utilizaram o sistema. Com isso em mente, e tendo em conta que o sistema construído recomendará eventos, foi pedido a vários utilizadores que escolhessem entre um a cinco eventos que gostariam de ir de modo a simular o ato de compra de bilhete dos mesmos, de seguida esses eventos foram passados pelo sistema construído onde serão apresentados cinco eventos recomendados. Por fim os utilizadores avaliaram cada um dos eventos recomendados numa escala de zero a cinco no quão boa era a recomendação, sendo zero uma recomendação que não nenhuma semelhança com os eventos escolhidos e cinco um evento que gostariam de ir. É de notar que os testes foram realizados num protótipo e não no próprio sistema pois este não possui utilizadores ativos por ainda estar numa fase inicial.

### 5.2 Resultados

Tendo em conta a metodologia mencionada anteriormente, foram obtidas avaliações para as recomendações de sete utilizadores, onde foram utilizados os eventos presentes na tabela 5.1. Os utilizadores escolhidos foram pessoas num faixa etária compreendida entre os vinte e trinta e cinco anos, com experiência em utilização de sistemas de compras de bilhetes ou reversas de hotéis. Foram escolhidos estes utilizadores pois eram os caracterizavam melhor o público alvo do sistema principal da *Yacooba*.

No *dataset* estão presentes nove eventos de música e sete de desporto, e neste também estão presentes a sua localização, *keywords* e quantidade de bilhetes vendidos e quantidade de bilhetes disponíveis.

ID	Name	Tipo	Localização	Bilhetes		Keywords
				Vendidos	Disponíveis	
1	Chelsea x Lille — Champions League	Sports	Stamford Bridge	43611	45000	[ 'chelsea', 'lille', 'champions league', 'soccer', 'football', 'stamford bridge' ]
2	Manchester United v Chelsea	Sports	Old Trafford	52928	70000	[ 'manchester united', 'chelsea', 'old trafford', 'manchester', 'soccer', 'football' ]
3	Chelsea v Watford	Sports	Stamford Bridge	10362	45000	[ 'chelsea', 'watford', 'stamford bridge', 'soccer', 'football' ]
4	Newcastle United v Arsenal	Sports	St. James' Park Stadium	7779	52000	[ 'newcastle', 'arsenal', 'soccer', 'football' ]
5	Wimbledon Championship 2022 - Day 13	Sports	All England Lawn Tennis and Croquet Club	8878	10000	[ 'tennis', 'wimbledon championship' ]
6	Wimbledon Championship 2022 - Day 14	Sports	All England Lawn Tennis and Croquet Club	4116	10000	[ 'tennis', 'wimbledon championship' ]
7	Al Nassr FC v Al-Shabab FC	Sports	King Fahd Stadium	14867	68000	[ 'soccer', 'football', 'al nassr', 'al shabab' ]
8	Kalorama Festival	Music	Parque da Bela Vista	860	10000	[ 'lisbon', 'summer', 'festival' ]
9	Cala Mijas Festival	Music	Sonora Mijas	10000	30000	[ 'festival', 'mijas', 'international' ]
10	Aladdin on Broadway	Music	New Amsterdam Theatre	981	1702	[ 'aladdin', 'broadway', 'theatre' ]
11	TINA - The Tina Turner Musical	Music	Aldwych Theatre	769	1200	[ 'tina', 'musical', 'theatre' ]
12	Matilda the Musical	Music	Cambridge Theatre	769	1231	[ 'matilda', 'musical', 'theatre' ]
13	Burning Man Festival	Music	Black Rock Desert	9300	10307	[ 'rock', 'desert', 'festival' ]
14	Bilbao BBK Live	Music	Bilbao BBK Live	3755	15000	[ 'live', 'bilbao', 'festival' ]
15	Azkena Rock Festival	Music	Azkena Rock Festival	8177	20000	[ 'rock', 'festival' ]
16	MAMMA MIA!	Music	Novello Theatre	917	1105	[ 'theatre', 'mamma mia', 'musical' ]

Tabela 5.1: Eventos utilizados para a avaliação do sistema de recomendações

Após os utilizadores escolherem eventos e receberem as suas recomendações avaliaram as mesmas, os valores da avaliação dada a cada uma das recomendações está apresentada abaixo:

User 1			User 2		
Escolhido	Recomendado	Avaliação	Escolhido	Recomendado	Avaliação
10	12	5	11	12	3
16	13	5	16	13	2
-	11	5	8	16	4
-	1	2	-	15	3
-	15	5	-	9	3
User 3			User 4		
Escolhido	Recomendado	Avaliação	Escolhido	Recomendado	Avaliação
11	12	4	1	5	4
14	13	3	2	16	2
16	10	3	14	11	4
9	15	4	15	12	3
-	8	4	-	13	0
User 5			User 6		
Escolhido	Recomendado	Avaliação	Escolhido	Recomendado	Avaliação
13	12	4	2	1	5
14	11	1	3	6	5
15	10	4	4	7	5
16	9	5	5	16	2
-	8	5	-	11	2
User 7					
Escolhido	Recomendado	Avaliação			
1	5	4			
4	2	4			
-	6	4			
-	3	4			
-	7	4			

Tabela 5.2: Resultados da avaliação do sistema de recomendações

Como podemos ver pela tabela acima os resultados foram positivos mostrando que o sistema foi avaliado numa nota positiva. Existem algumas avaliações com uma nota mais baixa que podem ser justificadas pelo facto de apenas terem sido considerados dezasseis eventos, por exemplo, nas avaliações do *User 6* verificamos que os eventos escolhidos foram todos de desporto e foram lhe recomendados mais três, fazendo com que não existam mais eventos de desporto para recomendar, recomendando assim eventos num local mais próximo dos eventos escolhidos. Todos os problemas vistos através desta avaliação são uma demonstração do fenómeno abordado na secção *Revisão da Literatura (The Cold Start Problem)*. Com esta avaliação concluímos que os utilizadores responderam, de uma forma geral, positivamente às recomendações que lhes foram apresentadas, e que estas podem ser melhoradas, especialmente, com a adição de novos eventos.

## 6. Discussão

Neste projeto foi necessário a criação de três tipos de recomendações, *personalizada*, *semi-personalizada* e *não personalizada*, onde foram usadas as compras passadas cujo um certo utilizador comprou bilhetes para a recomendação personalizada para o mesmo e foram usadas estatísticas obtidas pela *API* do *Google Analytics* para as recomendações não personalizadas para o utilizador, métodos que foram abordados na secção *Implementação*. Podemos verificar que todos os requisitos anteriormente mencionados foram cumpridos e tendo em conta a secção *Avaliação* verificamos que as recomendações tiveram um *feedback* positivo dos utilizadores.

Com todos os requisitos e objetivos cumpridos teremos de pensar nos próximos passos. Com o passar do tempo, serão adicionados cada vez mais eventos e haverá mais informação sobre a interação dos utilizadores com o sistema, isto permitirá que no futuro seja criado um grande *dataset* com a informação que existe disponível e poderão ser geradas melhores recomendações com ajuda dos métodos discutidos em *Revisão da Literatura* onde serão usados, especialmente, algoritmos de *machine learning* e *inteligência artificial* para poderem ser procuradas, por exemplo, semelhanças entre utilizadores de modo a descobrir possíveis gostos de cada um dos utilizadores melhorando assim as recomendações, que conseqüentemente melhoraram a experiência do utilizador no sistema.

## 7. Conclusão

Para este projeto foi proposto, pela *Yacooba*, a construção de um sistema de recomendações de eventos para ser implementado no seu sistema de venda de bilhetes de eventos. Para tal houve pesquisa sobre o problema em mãos, e consequentemente foi dividido em pequenos problemas mais pequenos, sendo assim, seria necessário criar três tipos de recomendações, *personalizada*, *semi-personalizada* e *não personalizada*, onde foram usadas estatísticas sobre a página de cada evento, a informação de cada evento e os bilhetes para eventos que o utilizador comprou no passado.

Após isso foi passa a uma revisão da literatura onde foram procurados métodos para o desenvolvimento desta solução, sendo maior parte destes métodos inutilizáveis devido ao problema *The Cold Start Problem*. Tendo isso em conta foi passada para a implementação onde foram listados os requisitos para o projeto e abordadas todas as ferramentas utilizadas, tais como *Google Analytics*, *Typescript*, entre outras. Também foi abordado o processo de desenvolvimento utilizado para tal e como foram desenvolvidas cada um dos tipos de recomendações com o auxílio do cálculo de um *score/heurística* para a avaliação dos eventos que determinará o quão recomendado este será.

Tendo já o sistema a gerar recomendações estava na hora de avalia-lo, algo que foi feito utilizando o *feedback* de utilizadores. A sete utilizadores, foi dado a escolher entre um a cinco eventos que gostariam de estar presentes, de seguida foram recomendados cinco eventos e cada um dos utilizadores teve a tarefa de avaliar cada uma das recomendações numa escala de um a cinco. Os resultados foram, de forma geral, positivos, havendo um ou outro evento que não estava a gosto do utilizar, algo que foi justificado pela reduzida quantidade de eventos e pela limitada quantidade de informação disponível. Estes problemas encontrados, tal como já discutido, serão resolvidos com o tempo quando chegarem mais eventos ao sistema e quando existir mais interação dos utilizadores com o mesmo. Havendo mais informação, o *The Cold Start Problem* é mitigado e poderão ser implementados métodos mais complexos de *machine learning* e *inteligência artificial* e poderão ser utilizados os outros métodos discutidos na secção *Revisão da Literatura*.

Todos os objetivos foram cumpridos com sucesso, e foi possível a aplicação de conteúdos de toda a licenciatura neste projeto, sendo a experiência profissional e conhecimentos obtidos pela realização deste projeto na empresa *Yacooba* muito valiosos.



## 8. Bibliografia

- [1] K. Falk, *Practical Recommender Systems*, 1st ed. Manning Publications Co., 2019
- [2] A. R. Lahitani, A. E. Permanasari and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," *2016 4th International Conference on Cyber and IT Service Management*, 2016
- [3] P. Sitikhu, K. Pahi, P. Thapa and S. Shakya, "A Comparison of Semantic Similarity Methods for Maximum Human Interpretability," *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, 2019
- [4] Charu C. Aggarwal, *Recommender Systems*, The Textbook, 1st ed. Springer International Publishing, 2016
- [5] Python Programming Language, <https://www.python.org>, visitado em 25-02-2022
- [6] scikit-learn: Machine Learning in Python, <https://scikit-learn.org>, visitado em 25-02-2022
- [7] pandas: Python Data Analysis, <https://pandas.pydata.org>, visitado em 25-02-2022
- [8] NumPy: Mathematical functions, <https://numpy.org>, visitado em 25-02-2022
- [9] Matrix Factorization for Recommender Systems, [https://everdark.github.io/k9/notebooks/ml/matrix\\_factorization/matrix\\_factorization.nb.html](https://everdark.github.io/k9/notebooks/ml/matrix_factorization/matrix_factorization.nb.html), visitado em 01-03-2022
- [10] Z. Zhang, T. Li, C. Ding and X. Zhang, "Binary Matrix Factorization with Applications," *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007
- [11] Google Colab: Colab notebooks, <https://research.google.com/colaboratory/>, visitado em 10-03-2022
- [12] *The history of Amazon's recommendation algorithm*, <https://www.amazon.science/the-history-of-amazons-recommendation-algorithm>, visitado em 03/05/2022

- [13] *Amazon's Product Recommendation System In 2021: How Does The Algorithm Of The eCommerce Giant Work?*, <https://recostream.com/blog/amazon-recommendation-system>, visitado em 03/05/2022
- [14] *Amazon Personalize*, <https://aws.amazon.com/pt/personalize/>, visitado em 03/05/2022
- [15] *Deep Dive into Netflix's Recommender System*, <https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48>, visitado em 03/05/2022
- [16] *How Netflix's Recommendation Engine Works?*, [https://medium.com/@springboard\\_ind/how-netflixs-recommendation-engine-works-bd1ee381bf81](https://medium.com/@springboard_ind/how-netflixs-recommendation-engine-works-bd1ee381bf81), visitado em 03/05/2022
- [17] *Carousel*, <https://getbootstrap.com/docs/4.0/components/carousel/>, visitado em 03/05/2022
- [18] *Redis*, <https://redis.io>, visitado em 29/03/2022
- [19] *Typescript*, <https://www.typescriptlang.org/>, visitado em 12/03/2022
- [20] *GraphQL — A query language for your API*, <https://graphql.org/>, visitado em 12/03/2022
- [21] *PostgreSQL*, <https://www.postgresql.org/>, visitado em 12/03/2022
- [22] *React – A JavaScript library for building user interfaces*, <https://reactjs.org/>, visitado em 12/03/2022
- [23] *Next.js by Vercel - The React Framework*, <https://nextjs.org/>, visitado em 12/03/2022
- [24] *MUI: The React component library you always wanted*, <https://mui.com/pt/>, visitado em 12/03/2022
- [25] *@google-analytics/data*, <https://www.npmjs.com/package/@google-analytics/data>, 20/03/2022
- [26] *@types/google.analytics*, <https://www.npmjs.com/package/@types/google.analytics>, 20/03/2022
- [27] E. Evans, *Domain-Driven Design*, 1st ed. Addison-Wesley Professional, 2003
- [28] Martin R., *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, 1st ed. Pearson, 2017
- [29] *Finding Nearest pair of Latitude and Longitude match using Python*, <https://medium.com/analytics-vidhya/finding-nearest-pair-of-latitude-and-longitude-match-using-python-ce50d62af546>, visitado em 22/03/2022
- [30] *Google Analytics*, <https://developers.google.com/analytics/devguides/reporting/core/v4>, visitado em 20/03/2022