EDUARDO CASTILHO DE ALMEIDA PRADO

ARQUITETURA DE UM SISTEMA DE RECOMENDAÇÃO BASEADO EM UMA REDE NEURAL

São Paulo
2022

EDUARDO CASTILHO DE ALMEIDA PRADO

ARQUITETURA DE UM SISTEMA DE RECOMENDAÇÃO BASEADO EM UMA REDE NEURAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

São Paulo
2022

EDUARDO CASTILHO DE ALMEIDA PRADO

ARQUITETURA DE UM SISTEMA DE RECOMENDAÇÃO BASEADO EM UMA REDE NEURAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Área de Concentração:
Engenharia da Computação

Orientador:
Prof. Dr. Jorge Luís Risco Becerra

São Paulo
2022

AGRADECIMENTOS

Aos meus pais pelo apoio incondicional em todos os momentos e em especial durante este longo e desafiador período de graduação.

Ao professor Jorge Luis Risco Becerra pela paciência, disposição e orientação durante a realização deste trabalho. Seu direcionamento foi essencial e possibilitou o aprendizado de lições valiosas sobre a engenharia.

Aos meus tios e tia pelos incentivos e instruções que serviram de guia para realização deste projeto. Serei eternamente grato pela dedicação e pelo apoio de vocês.

Aos amigos e colegas da faculdade por darem suporte constante durante os anos de graduação e tornarem todo esse tempo mais divertido.

E a todas as pessoas que contribuíram direta ou indiretamente com a realização deste trabalho.

RESUMO

A taxa de adoção de inteligência artificial no mercado atingiu um novo recorde em 2022, porém ainda se encontram barreiras para a empresas conseguirem implementar efetivamente sistemas inteligentes dentro de seus processos. Uma delas é a dificuldade de alinhar as necessidades do ponto de vista do negócio e suas demandas de dados, causando uma desconexão entre modelos criados e aplicações reais. Este trabalho apresenta uma arquitetura capaz de integrar um sistema de recomendação que utiliza aprendizado de máquina com uma aplicação web dentro de um contexto de sites de relacionamento. Para isso foi desenvolvido uma sistema de recomendação que utiliza uma rede neural convolucional para avaliar imagens escolhidas pelos usuário e classificá-las a partir de sua preferência. Esse sistema esta incorporado em um aplicação web nos modelos MVP que tem as principais características de uma aplicação de namoro.

Palavras-Chave – Inteligência artificial; Arquitetura de plataformas integradas; Redes Neurais; Transferência de aprendizado; Sistemas de recomendação; Aplicativos de relacionamento.

ABSTRACT

The adoption rate of artificial intelligence in the market reached a new record in 2022, but there are still barriers for companies to be able to effectively implement intelligent systems within their processes. One of them is the difficulty of aligning the needs of the business and its demands for data, causing a disconnect between models and real applications. This work presents an architecture capable of integrating a recommendation system that uses machine learning with a web application within the context of online dating. For this, a recommendation system was developed that uses a convolutional neural network to evaluate images chosen by the user and classify them based on their preference. This system is incorporated into a web application as a MVP that has the main characteristics of a dating website.

Keywords – Integrated platform architectures; Neural networks; Transfer Learning; Recommendation systems; Dating applications.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

MVP - Minimum Viable Product

IA - Inteligência Artificial

DSR - Design Science Research

CRISP-DM - Cross-Industry Standard Process for Data Mining

WWW - World Wide Web

TPR - True positive rate

FPR - False positive rate

TP - True positive

TN - True negative

FP - False positive

FN - False negative

SUMÁRIO

PARTE I

INTRODUÇÃO

1 INTRODUÇÃO

"The ultimate purpose of colleting the data is to provide a basis for action or a recommendation"

-- W. Edward Deming

Neste capítulo serão abordados os principais aspectos preliminares que serviram de base para a concepção desse trabalho de conclusão de curso. Serão expostas as motivações que vieram a originar a ideia do trabalho, o principal problema identificado a ser atacado pelo projeto e os objetivos a serem alcançados com seu desenvolvimento. Além disso, também serão apresentadas as justificativas para a relevância dessa dissertação, quais foram as principais metodologias utilizadas e por fim um breve resumo sobre a composição dos próximos capítulos.

1.1 Contexto Inicial

Em Maio de 2022 a IBM publicou seu índice global anual de adoção de inteligência artificial (IA) (IBM CORPORATION, 2022), pesquisa que analisou 7502 empresas de diferentes tamanhos e em diferentes países (incluindo o Brasil) para trazer a tona os dados sobre a implementação de IA no mercado. Nele foi averiguado que houve um aumento de 4% em relação ao ano passado na adoção de sistemas de IA ao redor do mundo atingindo um novo recorde de 35%. Fora essas empresas que já estão implementado IA em seus processos atualmente, um adicional de 42% estão fazendo pesquisa na área ou explorando formas de inserir essa tecnologia em seu negócio.

Esses dados apontam para um novo paradigma do mercado onde a utilização de sistemas inteligentes vai ser algo extremamente comum e generalizado no cotidiano das pessoas. Porém, assim como constatado nesse relatório, uma das maiores barreiras para empresas adotarem esse tipo de tecnologia é que muitas vezes os projetos de IA podem ser muito complexos, difíceis de integrar e tornar escalável. Frequentemente se observa uma dificuldade durante o ciclo de vida de desenvolvimento de um projeto em alinhar as necessidades do ponto de vista do negócio e suas demandas de dados, que poderiam ser satisfeitas pela área de ciência de dados através de modelos inteligentes.

Isso pode ser um problema ainda maior quando consideramos empresas menores, como *startups*, onde o time e os recursos são limitados, sendo incomum haverem pessoas capacitadas para implementar esses sistemas ou inviável ter o tempo necessário para executar projetos desse tipo.

Já em empresas maiores essa dificuldade acaba se demonstrando de forma diferente, onde a área de dados acaba evoluindo no seu próprio ecossistema e apenas solucionando demandas pontuais das outras áreas ou criando produtos próprios internos apenas para análise. Nesses casos, se torna comum haver uma desconexão entre os algoritmos desenvolvidos por cientista de dados e sua implementação prática em aplicações reais. Quem acaba sofrendo com isso são os usuários que não tem sua experiência diretamente afetada pelo que foi desenvolvido.

Sendo assim, se evidencia a importância de criar arquiteturas onde os modelos de inteligência artificial estejam diretamente conectados com o valor principal do negócio, utilizando algum método que facilite esse processo. Um dos sistemas onde isso é extremamente comum são os sistemas de recomendação, que além de terem como principal agente os próprios usuários, muitas vezes utilizam redes neurais (ou algum tipo de aprendizado de máquina) para gerar as informações que estão sendo observadas por eles.

1.2 Objetivos

Esse trabalho tem como objetivo criar uma arquitetura capaz de integrar um sistema de recomendação que utiliza aprendizado de máquina com uma aplicação web. O algoritmo será modelado tendo como objetivo principal a visão de negócio do sistema desenvolvido e deverá ser utilizado da forma que mais terá impacto na experiência do usuário.

O modelo de negócio que foi escolhido como inspiração para desenvolver o produto final foi o mercado de namoro e relacionamento online, onde redes neurais já são bem utilizadas para a recomendação e tem obtido muito sucesso nessa área. Um exemplo disso é o aplicativo *Tinder*, que tem desenvolvido algumas soluções de aprendizado de máquina para melhorar o pareamento de seus usuário (Tinder Group, 2019) (CARMAN, 2018) (LIU, 2017).

Será implementado uma aplicação web em modelo MVP (*minimun viable product*) onde usuários poderão criar suas contas, avaliar outros usuários com base em suas imagens, receber novas recomendações com base nesses usuários avaliados e, possivelmente, conta-

tar os usuários recomendados para estabelecer um relacionamento. Essas recomendações serão geradas por um modelo de aprendizado de máquina que utiliza redes neurais para classificar imagens e determinar a preferência. Cada usuário terá um modelo exclusivo e feito sob medida para encontrar seu par ideal dentre as pessoas disponíveis na plataforma. Esse modelo deverá ser capaz de determinar um grau de compatibilidade entre usuários com base nas imagens avaliadas e a classificação de *"like"* (aprovado) ou *"dislike"* (desaprovado) do usuário que avaliou os perfis.

Esses modelos também serão avaliados e ajustados por um especialista que terá acesso a uma plataforma separada para acompanhar os pedidos de recomendação dos usuários e analisar por meio de um dashboard a performance desses modelos. O especialista deverá ser capaz de julgar, a partir da plataforma, se o treinamento do algoritmo de recomendação obteve bons resultados e informar para o usuário um resumo simples dessa averiguação. Caso os resultados não estejam satisfatórios, o especialista poderia alterar os parâmetros necessários no modelo para obter resultados melhores.

A meta do sistema é elucidar o funcionamento de um aplicativo de relacionamento que utiliza modelos de aprendizado de máquina para fazer recomendações. Assim, seria possível demonstrar como implementar uma arquitetura modelada pensando no impacto que a inteligência artificial teria no usuário final, sem prejudicar a participação do cientista de dados no processo.

1.3 Justificativa

Algoritmos de recomendação são essenciais para tornar a experiência do usuário em aplicativos mais personalizada e agradável. Eles ajudam os usuários a encontrar o que procuram quando há um grande número de opções a se escolher e o tempo de escolha é limitado. Isso além de ser um caso de uso muito comum em e-commerces é também especialmente o caso para aplicativos de relacionamento com muitos usuários onde a escolha para encontrar as pessoas que mais agradam um determinado usuário é limitada pela fila disponível para este naquele momento.

Por isso, muitos aplicativos nesta área conhecida como *online dating*, (*Tinder*, *Bumble*, *Hinge*, *OkCupid*, entre outros) adotam diferentes estratégias de recomendação que normalmente são mantidas em segredo. Muito pode se aprender pela criação de uma arquitetura complexa que consegue implementar um sistema de recomendação dentro de uma aplicação funcional.

Tais algoritmos de recomendação, quando aplicados com sucesso nas empresas como foi o caso da *Netflix* (Gokul, 2020), comprovadamente garantem a retenção dos usuários uma vez que sua experiência personalizada ajuda na recompensa psicológica ao aumentar o número de pessoas que o usuário aprecia ao mexer no aplicativo e a eficácia do pareamento de pessoas, já que ele causa um aumento no número de possíveis conexões bem sucedidas.

1.4 Metodologia

A metodologia principal escolhida para desenvolver a monografia foi baseada nas diretrizes do método de *Design Science Research* (WIERINGA, 2014), que é basicamente divido em três etapas: definição do problema, ciclo do projeto e ciclo empírico. Essa estrutura foi utilizada como fundamento para delimitar a organização dos capítulos deste trabalho de conclusão de curso, sendo a primeira corresponde aos capítulos 1 e 2 de introdução, a segunda referente aos capítulos 3, e a última referente ao capítulo 4.

Para a elaboração da arquitetura principal do projeto também foi utilizada parte das técnicas de modelagem conhecidas como RM-ODP (Reference Model of Open Distributed Processing). Essas técnicas auxiliam no desenvolvimento de sistemas complexos, pois estabelecem diferentes visões arquiteturais sobre ele.

Além disso, para o desenvolvimento específico do algoritmo de aprendizado de máquina foi utilizada a metodologia conhecida como CRISP-DM (*Cross-Industry Standard Process for Data Mining*), criada no final da década de 90 por um grupo de empresas preocupadas em definir processos que pudessem orientar o desenvolvimento de aplicações de mineração de dados.

1.4.1 Design Science Research Framework

O *Design Science Research Framework* (DSR) consiste em investigar e projetar um artefato em um contexto específico com a intenção explícita de melhorar algum aspecto desse contexto . Esse contexto reflete algum problema que foi identificado, seja na empresa, na instituição ou em qualquer meio no qual ele estiver inserido. O objetivo da metodologia é aprimorar ou resolver esse problema com a implantação desse artefato, que nada mais é que um modelo, um método ou uma instanciação construída pelo homem.

O contexto do problema de um artefato pode ser estendido com os *stakeholder* do artefato e com o conhecimento usado para projetar o artefato. Este contexto estendido pode ser considerado o contexto do projeto de *design science* como um todo. Podemos

observar como essas partes se relacionam na Figura 1.

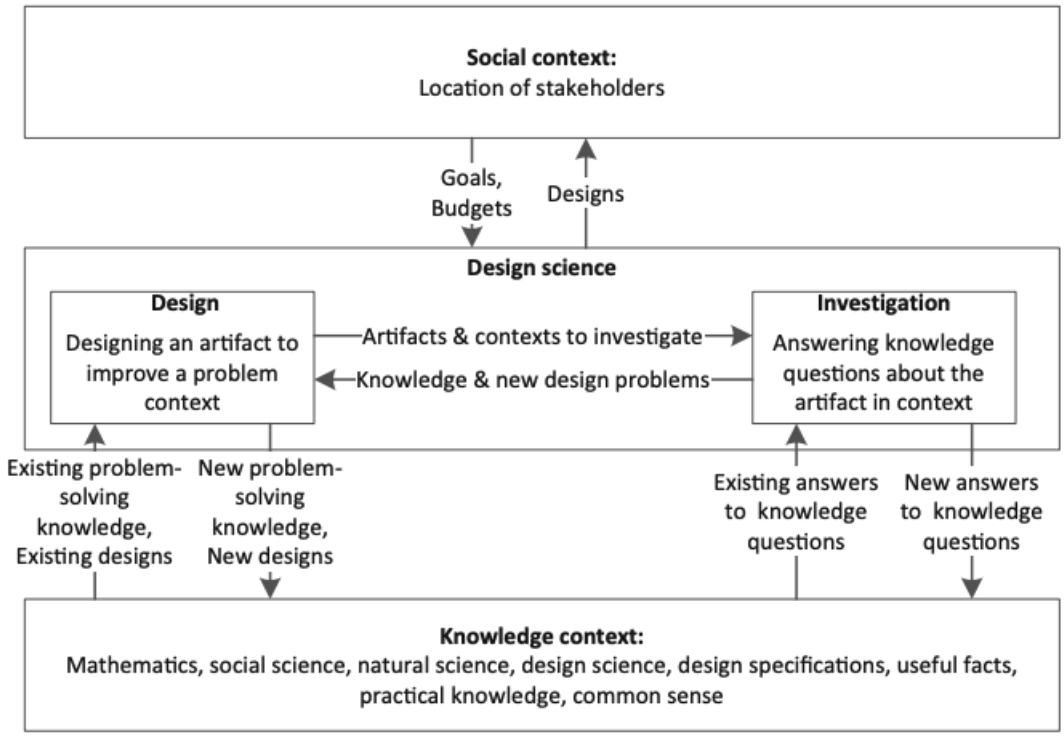

Figura 1: Framework para design science
Fonte: (WIERINGA, 2014)

Essa metodologia é extremamente relevante para o desenvolvimento do projeto de conclusão de curso, pois um dos objetivos deste trabalho é conseguir criar uma solução viável e prática para um problema identificado no contexto de integração de sistema inteligentes.

1.4.2 Reference Model of Open Distributed Processing (RM-ODP)

O objetivo do RM-ODP é fornecer uma estrutura para especificar e construir sistemas grandes ou complexos. Esses sistemas produzidos são denominados sistemas ODP e podem ser sistemas de TI clássicos, sistemas de informação, sistemas embarcados, entre outros (LININGTON, 2012).

Este *framework* consiste em dividir uma especificação complexa em 5 distintas visões ou arquiteturas, cada uma seu conjunto de regras e observadas sob o ponto de vista de diferentes *stakeholders*. Esses pontos de vista são:

- **Visão da empresa**: Representa objetivos, regras de negócio e políticas a serem

inseridos no sistema a ser desenvolvido;

- **Visão da informação**: Modela a informação compartilhada e manipulada dentro da empresa ou organização de interesse;
- **Visão da computação**: Desenvolve o design de alto nível dos processos e aplicações suportados pelas atividades da empresa;
- **Visão da engenharia**: Define um conjunto consistente de serviços de comunicação e outros serviços de suporte para uso das aplicações;
- **Visão da tecnologia**: Representa principalmente as tecnologias utilizadas no sistema e os componentes de hardware e software, além da comunicação entre estes.

Dentre essas, apenas 3 delas (visão da empresa, computacional e tecnologia) foram utilizadas no desenvolvimento do projeto dada a baixa complexidade do sistema. Elas serão exemplificadas no capítulo 3, ARQUITETURA DO SISTEMA.

1.4.3 Metodologia CRISP DM

A metodologia CRISP DM busca fornecer uma estrutura para a realização de projetos de mineração de dados de forma a tornar eles menos custosos, mais confiáveis, mais iteráveis, mais gerenciáveis e tudo de forma mais rápida.(WIRTH; HIPP, 2000). Ao longo dos anos essa metodologia se consolidou no mercado e tornou-se a mais utilizada por desenvolvedores tanto no campo de mineração de dados quanto no campo ciência de dados, uma vez que ela auxilia muito na organização dos processos que precisam ser feitos para se obter um modelo que se adeque às necessidades do negócio.

Essa metodologia consiste em seis etapas que não necessariamente devem ocorrer em uma sequência restrita, já que muitas vezes existe um processo cíclico entre algumas etapas conforme as especificações de cada sistema. Na Figura 2 estão explicitadas as etapas do processo e como se dá a relação entre elas.

Podemos definir o que ocorre em cada etapa da seguinte forma:

- *Busines Understanding*: Compreensão e estabelecimento dos objetivos e requisitos do projeto sob uma perspectiva de negócios seguido da conversão desse conhecimento em um problema de mineração de dados.
- *Data Understanding*: Coleta inicial e familiarização com o conjunto de dados a fim de identificar quaisquer problemas de qualidade ou vieses;

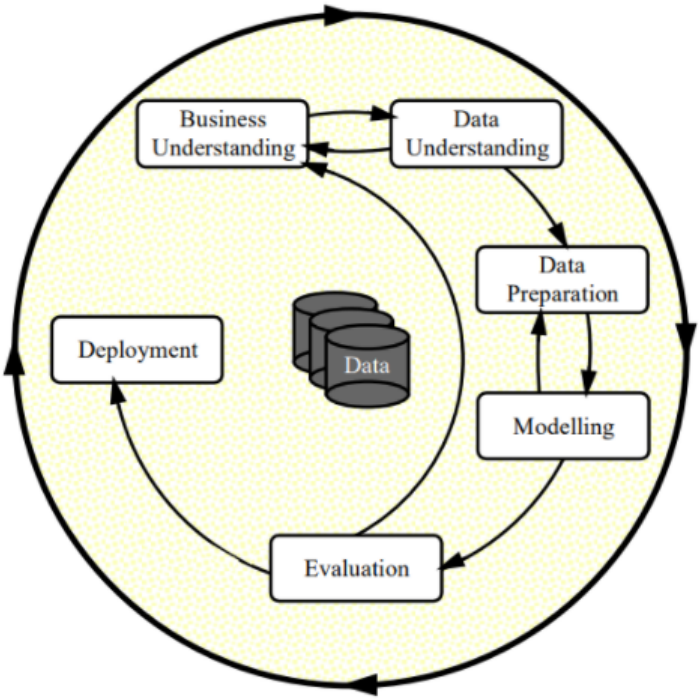

Figura 2: Fases do modelo original CRISP-DM.
Fonte: (WIRTH; HIPP, 2000)

- *Data Preparation*: Limpeza de dados problemáticos ou irrelevantes, integração de conjuntos de dados diferentes, transformação de dados existentes através de agragações ou normalizações e seleção e construção dos atributos do modelo. O objetivo é criar o conjunto de dados final que será utilizado para alimentar o modelo de aprendizagem;

- *Modeling*: Seleção da técnicas de modelagem considerando os objetivos identificados na fase de compreensão do negócio, geração de testes para calibragem dos parâmetros, construção do modelo e sumário preliminar dos resultados;

- *Evaluation*: Avaliação dos resultados e do desempenho do modelo considerando os requisitos mínimos e revisão do processo como um todo para identificação de possíveis problemas não averiguados previamente;

- *Deployment*: Apresentar o modelo de forma que o consumidor ou o cliente que necessitava de tal serviço consiga interagir com ele da maneira que ele precisava;

A utilização dessa metodologia foi necessária para garantir que os objetivos levantados durante a fase de solução do problema fossem alcançados e os requisitos mínimos do modelo se adequassem a arquitetura desenvolvida para o projeto.

1.5 Estrutura do trabalho

A seguir está descrita a composição dos próximos capítulos do presente trabalho e seu principal objetivo no contexto desta monografia:

No capítulo 2, ASPECTOS CONCEITUAIS, foram apresentados os principais conceitos explorados e empregados no desenvolvimento do trabalho e o atual estado da arte da literatura existente.

No capítulo 3, ARQUITETURA DO SISTEMA, é explicado qual foi o processo de desenvolvimento do projeto, quais são as especificações do sistema, como foi definida a arquitetura e quais são as tecnologias e ferramentas mais relevantes utilizadas para o desenvolvimento do sistema.

No capítulo 4, IMPLEMENTAÇÃO, é apresentado o processo de desenvolvimento do sistemas dividido em suas principais etapas de realização.

Por fim, no capítulo 5, CONSIDERAÇÕES FINAIS, são apresentados as conclusões obtidas, a avaliação do cumprimento dos objetivos elucidados e as possíveis continuações em trabalhos futuros.

2 ASPECTOS CONCEITUAIS

Nesse capítulo serão abordados os principais aspectos conceituais a serem utilizados nessa monografia. Começa-se apresentando os principais conceitos teóricos discutidos: aprendizado de máquina, redes neurais convolucionais, aprendizado de máquina e sistemas de recomendação. Em seguida, são exemplificados algumas representações genéricas do funcionamento de aplicativos de relacionamento e suas principais caraterísticas. Por fim, uma breve explicação do atual estado da arte, com uma análise de outros trabalhos e abordagens que seguiram com projetos similares.

2.1 Aprendizado de máquina

O aprendizado de máquina, do inglês *machine learning* (ML), é um ramo da área de inteligência artificial focado na criação de algoritmos com o fim de permitir a um computador reconhecer padrões e realizar inferências, sem que ele seja diretamente programado para isso. Um sistema baseado em algoritmos de aprendizagem é capaz de, pouco a pouco, melhorar os seus resultados à medida em que vai sendo exposto a dados e consegue acumular experiência a partir deles, algo similar ao que acontece no aprendizado humano. (ALPAYDIN, 2014)

Existem diversos tipos de aprendizagem de máquina e algoritmos que processam dados para cumprir diferentes funções de negócio. Para intuito deste trabalho iremos apenas nos focar nas metodologias de redes neurais convolucionais que são os principais tópicos utilizados na implementação da solução.

2.2 Rede neural convolucional (CNN)

Uma rede neural convolucional (CNN) é um tipo de aprendizado de máquina que utiliza algoritmos de aprendizado profundo (*deep learning*) com o intuito principalmente de classificar imagens, uma vez que trabalha essencialmente com *inputs* de maneira matricial.

Essas redes neurais, também conhecidas como *ConvNet*, são capazes de capturar com sucesso as dependências espaciais e temporais em uma imagem através da aplicação de filtros relevantes. A arquitetura realiza um melhor ajuste ao conjunto de dados da imagem devido à redução no número de parâmetros envolvidos e reutilização de pesos.

Os principais elementos das CNNs para realização de tais ajustes são os *kernels*, matrizes utilizadas para convolução com os dados de entrada de cada camada, e os *stride lengths*, que correspondem ao tamanho do passo de cada operação de convolução. Os *kernels* e *stride lengths* podem variar em tamanho conforme a operação específica.

Tais redes neurais utilizam a ideia de *pooling*, ou seja, agregação de informação para trabalhar com essas matrizes de dados. O objetivo desse processo é diminuir o número de parâmetros entre as etapas, otimizando o desempenho sem haver a perda de informações relevantes sobre a matriz de uma etapa em relação a outra de forma. A Figura 3 evidencia esse processo de forma mais clara.

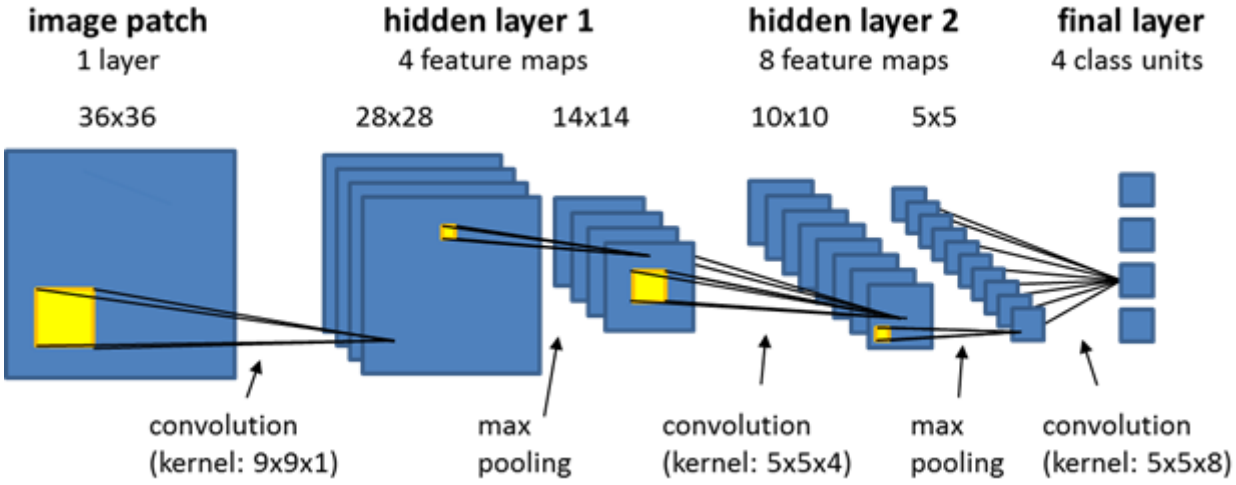

Figura 3: Exemplo do funcionamento de um CNN a partir do uso de *max-pooling*, onde o maior valor máximo da vinhança é selecionado.

Fonte: eCognition documentation

2.3 Transferência de aprendizado (*Transfer learning*)

Transferência de aprendizado é uma técnica utilizada para algoritmos de aprendizado profundo (*deep learning*) onde um algoritmo que já foi pré-treinado é utilizado como fonte para modelar um novo algoritmo com apenas algumas alterações em seus nós neurais. Essa estratégia tem como objetivo principal aproveitar o conhecimento desses modelos pré-treinados especializados em um determinado *dataset* que muitas vezes envolvem milhões

de pontos de dados que foram treinados durante muitas horas em supercomputadores para resolver problemas novos com características semelhantes.

Esse tipo de abordagem é particularmente comum em algoritmos que envolvem o tratamento de imagem onde já existem diversos *ConvNets* com alta performance desenvolvidas por grandes empresas do mercado (Google, IBM). Seus algoritmos podem ser adaptados para novas funcionalidades de forma a ainda utilizar a classificação de imagens do modelo inicial e economizando um número enorme de datapoints que seriam precisos para identificar características específicas da imagem.

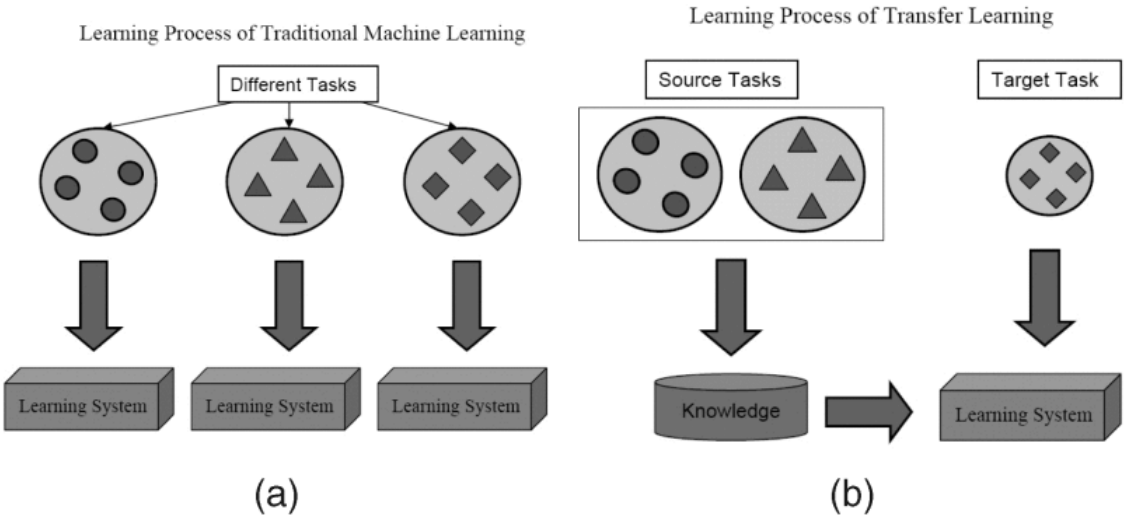

Figura 4: Diferentes processos de aprendizagem entre (a) aprendizagem de máquina tradicional e (b) aprendizagem por transferência.
Fonte: (PAN; YANG, 2009)

2.4 Classificação binária

A classificação binária é a tarefa de classificar os elementos de um conjunto de dados em dois grupos distintos com base em uma regra de classificação. Quando aplicado no contexto de aprendizado de máquina, esse tipo de problema é considerado um aprendizado supervisionado, onde duas categorias são predefinidas em um conjunto de dados inicial que serve de base para categorizar novas observações probabilísticas aplicando algum tipo de algoritmo.

Em algoritmos de classificação binária a saída muitas vezes é uma pontuação de previsão. A pontuação indica a certeza do sistema de que determinada observação pertence à classe positiva. A Figura 5 demonstra como essa classificação funciona em um conjunto de observações.

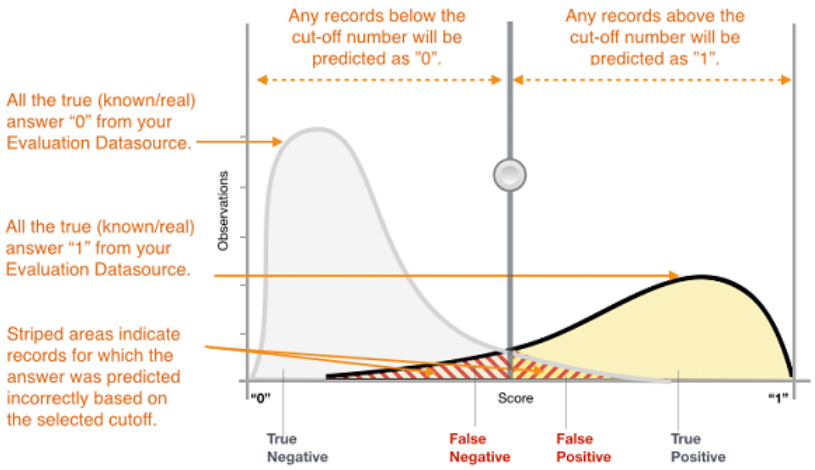

Figura 5: Distribuição de pontuação para um modelo de classificação binária
Fonte: (AWS, 2021)

Existem muitas métricas que podem ser usadas para medir o desempenho de um classificador e cada aplicação terá um conjunto específico que fará mais sentido pros objetivos daquele sistema. Porém, no geral, as principais métricas/informações utilizadas nesse sistema são (ZHENG, 2015):

- **Matriz de Confusão**: Tabela que permite a visualização dos 4 principais valores absolutos de uma classificação binária: *true positive* (TP), **true negative** (TN), *false positive* (FP), *false negative* (FN).
- **Precisão**: Determina qual proporção de identificações positivas estava correta e pode ser definida como:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: Determina qual proporção de positivos verdadeiros foi identificada corretamente e pode ser definida como:

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score**: A precisão e o recall tem sempre valores relacionados e muitas vezes melhorar um significa reduzir o outro por isso foi criado uma métrica que relaciona ambos a partir de sua média harmônica:

$$Recall = \frac{2 * Precision * Recall}{Precision + Recall}$$

- **Curva ROC**: A curva de característica de operação do receptor é um gráfico que mostra o desempenho de um modelo de classificação em todos os limiares de classificação. Ela representa dois parâmetros, a taxa de verdadeiro positivo (TPR) e a taxa de falso positivo (TPR) para valores de 0 a 1. Normalmente, quanto maior a área sob a curva desse gráfico (AUC - *area unde curve*), melhor o resultado do classificador.

2.5 Sistemas de recomendação (Recommender systems)

Sistemas de recomendação são técnicas de software que fornecem sugestões de itens a serem indicados para um usuário. Tais indicações auxiliam o usuário na tomada de decisão quando o número de itens é grande demais e o processo de descoberta de novos itens é limitado.

Normalmente são soluções muito utilizadas em lojas de vendas online, mas suas aplicações são variadas e existem muitos desenvolvimento no ramo de aplicativos de relacionamento. Existem diversas abordagens para sistemas de recomendação, mas os tipos mais comuns serão explicados a seguir.

2.5.1 Filtragem Colaborativa

Filtragem colaborativa parte do princípio que usuário que tem gostos parecidos no passado terão gostos parecidos no futuro e eles provavelmente gostarão de itens parecidos com os quais eles gostaram no passado. Tais sistemas se baseiam bastante nos conceitos de usuários vizinhos que têm características similares entre si e assim a itens interessantes para um, provavelmente serão interessantes para o outro.

O exemplo mais comum desse tipo de filtragem seria a recomendação de amigos do *Facebook* que se baseia nas conexões de seus amigos para determinar "usuários que você talvez conheça".

2.5.2 Filtragem baseada em conteúdo

Filtragem baseada em conteúdo já não utiliza a informação de outros usuários para determinar a preferência de um grupo de pessoas. Tal filtragem se baseia somente na relação entre os atributos de um determinado item e a relação dele com outros itens com atributos parecidos. Nesse caso o conceito de vizinho é mais aplicado para a relação entre os itens em si e seus atributos do que entre os usuários.

Um exemplo desse tipo de filtragem seriam as recomendações do Spotify de músicas que tocam após determinada música de um estilo específico ter acabado de terminar.

2.5.3 Recomendação híbrida

Atualmente é provavelmente o modelo mais adotado por sistemas de recomendação uma vez que ele busca complementar a falha de um modelo com os pontos fortes de outro e criar uma recomendação personalizada mais correta e que melhor reflete os usuários. Podem ser uma combinação de diversos tipos de modelo de recomendação, inclusive alguns não mencionados neste trabalho, mas comumente são um modelo misto entre uma filtragem colaborativa e uma filtragem baseada em conteúdo.

2.6 Modelo e comportamento de aplicativos de relacionamento

Para a compreensão dos casos de uso elucidados no sistema desenvolvido é importante estabelecer o funcionamento padrão da maioria dos aplicativos de relacionamento que o sistema irá simular e estabelecer alguns conceitos que serão utilizados ao longo desta monografia.

Vale ressaltar que apesar de aplicativos de relacionamento terem funcionalidades distintas conforme sua proposta de negócio e público alvo a maioria segue alguns padrões de design e comportamento que serão utilizados como base para o desenvolvimento da plataforma web.

Os aplicativos mais famosos e proeminentes no mercado de relacionamentos (*Tinder*, *Bumble*, *PlentyOfFish*, *OkCupid*, *Grindr* e *Hinge*) utilizam um sistema parecido para demonstrar interesse ou desinteresse entre os usuários. Nestes aplicativos, se um usuário está interessado em outro ele demonstra seu interesse “curtindo” o perfil deste outro usuário. Se ele não estiver interessado, ele pode ”discutir”esse perfil. Essas ações além de

serem “anônimas” para os usuários que a receberam (um usuário só saberia a reação de outro se houvesse compatibilidade entre ambos ou estivesse contratando um serviço pago do aplicativo) estão muitas vezes associadas a um botão ou um movimento de swipe na tela (deslizar para a direita caso haja interesse e para esquerda caso não haja). Pode-se observar um exemplo da interface desses aplicativos na Figura 6.

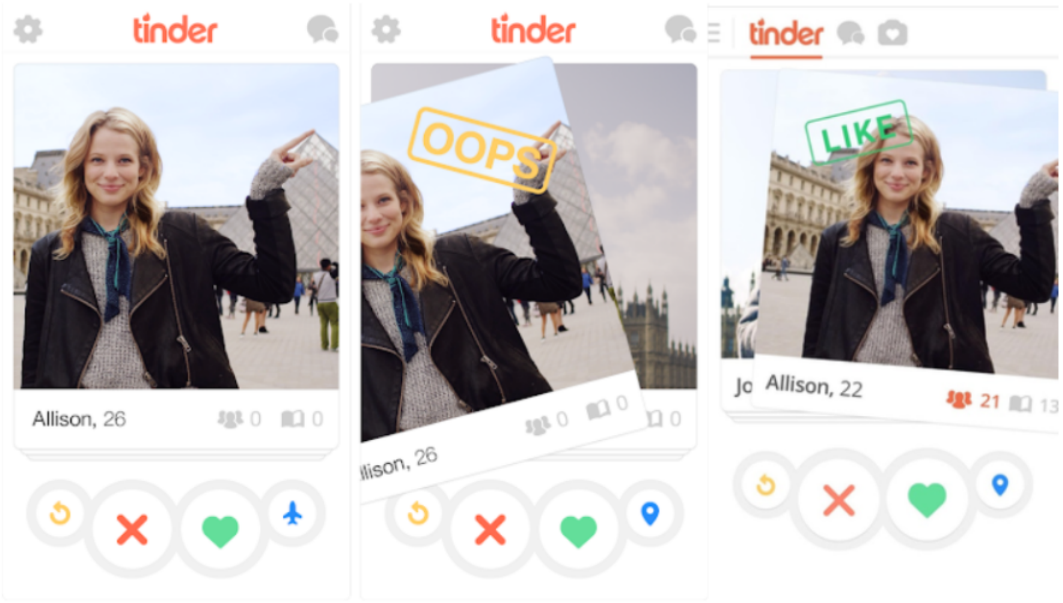

Figura 6: Interface do aplicativo *Tinder*.
Fonte: https://blog.photofeeler.com/if-you-swipe-right-on-tinder-do-they-know/

Muitas vezes esses aplicativos também contém outras formas de demonstrar interesse, como a funcionalidade de *Super Like* do *Tinder*, uma forma de “curtir” alguém sem ser anonimamente. Porém para as especificações deste trabalho tal funcionalidade não será analisada devido a complexidade que ela adicionaria ao sistema.

Além das reações que os usuários podem fazer entre si (“curtir” e ”descurtir”), a maioria dos aplicativos de relacionamentos utilizam uma fila de cartões de usuários contendo suas fotos para que o usuário avaliador possa descobrir novas pessoas. Esta fila impede o usuário avaliador de reagir a mais de um usuário avaliado por vez (representado pelo cartão do usuário na tela principal). Desta forma ele deve primeiro avaliar o usuário na frente da fila para seguir continuando avaliando outros usuários. Devido a essa limitação que os algoritmos de recomendação são tão importantes para gerar uma fila ideal na qual os primeiros usuários da fila tem a maior probabilidade de agradar o usuário avaliador.

PARTE II

DESENVOLVIMENTO

3 ARQUITETURA DO SISTEMA

Neste capítulo será apresentado como a metodologia escolhida para o projeto foi aplicada durante o processo de desenvolvimento, assim como ela serviu de base para a definição dos requisitos do sistema. Será definido e explicado a arquitetura final concebida para o projeto considerando o *framework* de RM-ODP. Por fim, será apresentado todas as principais tecnologias que foram utilizadas para desenvolver o sistema proposto e permitir a solução do problema.

3.1 Processo de Desenvolvimento

Considerando a metodologia de *Design Science* utilizada no projeto e com o problema bem definido (criar uma arquitetura capaz de integrar um sistema de recomendação e uma aplicação web), seguiu-se para a etapa de ciclo do projeto.

Nessa etapa serão propostas soluções para o problema estabelecido, porém para isso é necessário primeiro identificar os principais *stakeholders* do sistema. Como um dos objetivos é unir duas frontes dentro de uma mesma aplicação, são identificados dois *stakeholders* principais para o sistema que estão definido na tabela 1.

Em seguida foi levantado as principais características de um aplicativo de relacionamento e quais são suas funcionalidades essenciais para o desenvolvimento de uma aplicação MVP.

Stakeholders	**Descrição**
Usuários comuns	Todos os usuários que se cadastraram no sistema com o intuíto de se relacionarem com pessoas da plataforma.
Especialista de dados	Usuário especial, da categoria *admin* com capacidade técnica para compreender o funcionamento de redes neurais.

Tabela 1: Stakeholders do sistema.

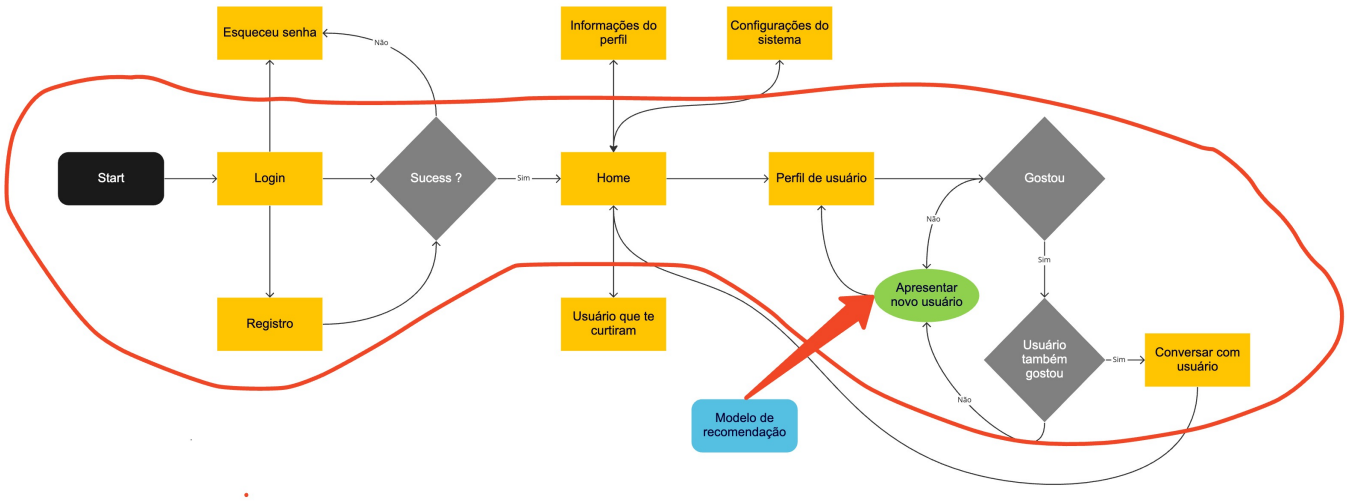

Figura 7: Principais funcionalidades de aplicativos de relacionamento.
Fonte: Autor

O fluxograma da figura 7 demonstra quais são essas funcionalidades em apps como *Tinder* e *Bumblee*. A área delimitada em vermelha foi identificada como o fluxo fundamental nesse sistema, que contém os principais casos de uso e o maior impacto na experiência do usuário. Como explicado na seção anterior, o módulo da cor verde de "apresentar novo usuário"é o módulo responsável por criar a fila de usuários apresentando a próxima pessoa e, em muitos aplicativos do mercado, seria nele que se encontraria algum tipo de algoritmo de recomendação.

3.1.1 Desenvolvimento da aplicação

Com os principais casos de uso e agentes do sistema definidos foi possível desenvolver um diagrama de casos de uso para iniciar o desenvolvimento da aplicação. Porém, antes disso, seguindo o *framework* RM-ODP e considerando que o produto desenvolvido está acompanhando as diretrizes do processo aplicado no curso de laboratório de engenharia de software I (BECERRA, 2019), nessa etapa alguns quesitos da visão da empresa precisam ser definidos.

Primeiramente foi necessário nomear o sistema de forma que espelhe os principais valores do projeto e que reflita a identidade da empresa. Logo, foi escolhido o nome *perfect match*, já que o diferencial do produto é ter uma recomendação integrada no sistema com o intuito de encontrar o par ideal para seus clientes.

O segmento da empresa definitivamente é o mercado de relacionamentos online e sua missão pode ser categorizada como: *"Criar um aplicativo de relacionamento que consiga*

Figura 8: Logo da empresa.
Fonte: Autor

encontrar a pessoa que mais combina com você.”.

Finalmente, com essas etapas definidas podemos analisar os casos de uso do sistema para cada agente e delimitar qual será o escopo dessa aplicação MVP. Na figura 9 temos o diagrama de casos de uso do sistema.

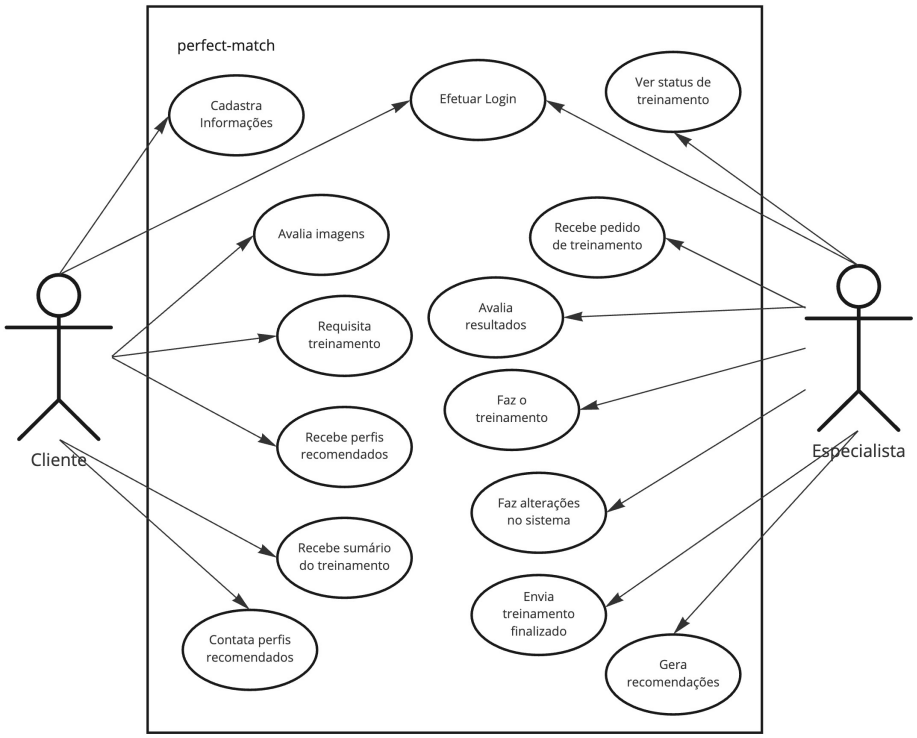

Figura 9: Diagrama de casos de uso do perfect-match.
Fonte: Autor

3.1.2 Desenvolvimento do sistema de recomendação

O sistema de recomendação utilizado segue os modelos desenvolvidos em outros projetos que obtiveram sucesso para o caso de um indivíduo avaliando sua própria preferência (AOYAMA, 2020) (LI, 2018). Este sistema se baseia em uma rede neural convolucional capaz de classificar imagens com base nas reações dos usuários.

Para o desenvolvimento dessa rede neural foi utilizada a metodologia CRISP-DM para alinhar as necessidades do produto com a arquitetura da rede que seria desenvolvida. A seguir as etapas da metodologia serão descritas exemplificando seus respectivos desenvolvimentos.

3.1.2.1 Compreensão da necessidade do negócio do ponto de vista da rede neural

No caso do algoritmo a ser desenvolvido para essa dissertação o plano a ser seguido consiste em identificar o gosto do usuário através das reações dele na aplicação e assim gerar um sistema de recomendação inteligente. Para isso o algoritmo utilizado irá usar o conceito de *transfer learning* para retreinar um modelo de CNN (convolutional neural network) desenvolvido pela Google (Inception v3) para encontrar padrões nas imagens categorizadas pelos usuários (que irão ser utilizadas como *input* de treino para o modelo) e conseguir determinar a probabilidade de uma nova imagem ser bem ou mal avaliada com uma acurácia significativa.

Esse modelo define os parâmetros de aprendizado do sistema que, portanto, receberá como *input* informações de imagens e uma informação de categorização do tipo binária, onde ou o usuário aprova a imagem ou ele desaprova ela. Além disso o *output* do sistema será uma classificação que determinará a qual dessas classes uma imagem pertence.

3.1.2.2 Compreensão dos bancos de imagens a serem utilizados na aplicação

Após os requisitos estarem bem estipulados etapa anterior será necessária a coleta e compreensão dos dados a serem analisados para gerar o modelo desejado.

Existem diversos conjuntos de dados que contém exclusivamente imagens que podem ser utilizados como fonte principal de dados da aplicação disponíveis de forma gratuita na plataforma *Kaggle*. Esses dados por serem de domínio público não precisam sofrer qualquer tipo de alteração para proteção da identidade das pessoas retratadas.

O banco de dados escolhido para ser usado não só como treinamento do modelo, mas como fonte de usuários da aplicação foi o banco Labeled Faces in the Wild (LFW). Este contém 13234 imagens de pessoas famosas no formato jpg com resolução de 250x250 px. Apesar de ser um banco utilizado principalmente com o objetivo de resolver problemas da área de reconhecimento facial, ele é extremamente útil para o caso de uso do projeto. Os principais motivos para a escolha deste banco em particular foram:

- Ele é um dos poucos banco de dados disponíveis de imagens de rosto de pessoas (principalmente) de uso aberto que contém a informação do nome associado a imagem analisada. Isso é de extrema importância pois como essas imagens representarão usuários na plataforma é necessário que seja disponível o nome de tais usuários para emular verídica mente um aplicativo de relacionamento. Ademais essa informação pode ser utilizada para determinar o gênero do usuário com uma alta probabilidade de acerto e sem utilizar informações da imagem, o que poderia interferir nos resultados finais de determinação de preferência do usuário.

- 1680 pessoas do banco tem mais de uma imagem, o que auxilia o usuário "avaliador"a tomar uma decisão mais informada e auxilia o próprio algoritmo a aprender mais padrões com um número menor de perfis avaliados.

- O banco têm uma boa distribuição de perfis similar a um aplicativo de relacionamento. Com uma distribuição relativamente boa de perfis masculinos (66%), femininos (24%) e indeterminados (10%). Tal distribuição foi calculada utilizando a biblioteca de python *gender_guesser*. Além disso a distribuição de idades também segue o padrão de aplicativos de relacionamento com muito poucas crianças, nenhum bebê e muito poucas pessoas com mais de 80 anos.

- As imagens do banco já estão padronizados em termos de tamanho, o que facilita a etapa de pré processamento e garante um tempo de resposta menor para análise dos perfis.

3.1.2.3 Preparação dos dados

Com o conjunto principal de dados coletado, analisado e definido para atender os requisitos da aplicação se iniciará a próxima etapa do processo onde será necessário realizar modificações nesse conjunto para atender as necessidades do algoritmo em si e conseguir uma melhor acurácia. Essa etapa andará lado a lado com a etapa seguinte em um processo cíclico até se definir um conjunto de dados que não gere problemas para a modelagem, seja por estarem com valores não normalizados ou pelo fato do modelo estar super adaptado ao conjunto de dados analisado, problema normalmente chamado de *overfitting* (BILBAO; BILBAO, 2017).

Nessa etapa 2 processos foram o que geraram os melhores resultados em termos de acurácia do sistema. Primeiro, foi identificado que um problema frequente que pode ocorrer é o fato do usuário acabar tendo duas classes não equivalentes quando finalizar

seu fluxo de avaliação dos usuários. Normalmente, em sites de relacionamento, é comum as pessoas serem mais exigentes na escolhas o que causa que a maioria das imagens sejam classificadas como o grupo de *dislike*.

Classes desequilibradas acabam gerando um aprendizado pior para o sistema, que acabaria também favorecendo a classificação negativa. Sendo assim, foi implementado uma tática para lidar com o balanceamento das classes que foi gerar amostras sintéticas da classe menos favorecida. Por meio de rotações e reflexões nas imagens do grupo menos representado seria capaz em aumentar em até 7 vezes o numero absoluto de imagens na amostra, o que já lida com a maioria dos problemas de desbalanceamento. Um exemplo disso aplicado na prática pode ser visto na figura **??**.

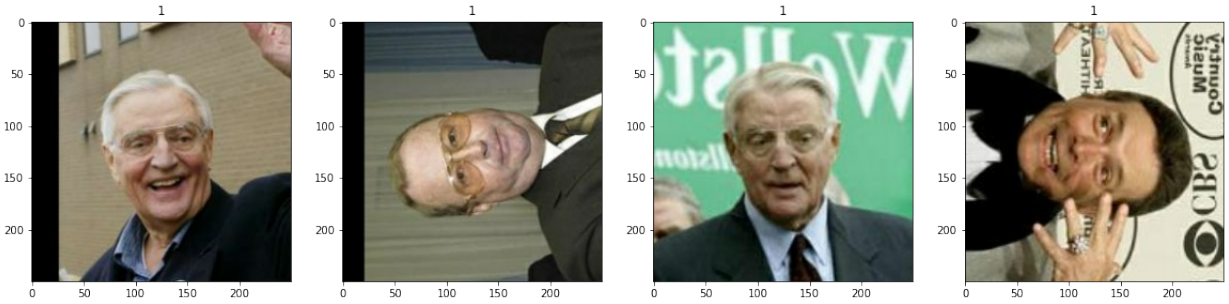

Figura 10: Imagens geradas sinteticamente para balancear os grupos. o número 1 em cima das imagens representa a classe de *likes*

Fonte: Autor

Depois, foi necessário normalizar as matrizes que representam os pixels das imagens de forma que os valores variam apenas de [0 - 1] e não de [0 - 255] como é o caso das imagens coloridas.

Por fim, o conjunto de dados é separado em dados de validação e dados para treinamento. Depois de algumas iterações, foi definido que com 70% dos dados utilizados para treinamento se obtém os resultados mais satisfatórios.

3.1.2.4 Modelagem

Como foi evidenciado na etapa anterior, essa fase consiste de um processo contínuo de iterações entre a limpeza dos dados e a geração de um modelo com base nesses novos dados "limpos" de teste. Durante o processo é comum que seja necessária a adoção de novas estratégias ou até mesmo novos modelos que melhor se adequam ao conjunto de dados para gerar o algoritmo desejado como uma acurácia significativa.

Para a rede neural desenvolvida a técnica utilizada foi a de aplicar transferência de

aprendizado (*transfer learning*) de um modelo já treinado para classificação de imagens (Inception v3) de forma a conseguir retreina-lo no novo conjunto de dados com os novos rótulos (*likes* ou *dislikes*) que fazem sentido para o objetivo de compreensão de preferência do usuário.

Essa técnica é a mais eficaz não só por ser uma estratégia normalmente utilizada para esse tipos de algoritmos que envolvem classificação de imagens, mas também pois ela diminui o problema que outras técnicas mais customizadas (como por exemplo a criação de um modelo CNN do zero) teriam de necessitar de uma enorme quantidade de dados de avaliação para gerar um resultado significativo.

Logo, uma vez com o modelo base importado no notebook, os seguintes passos foram feitos:

- As camadas do modelo base são "congeladas", ou seja, seus parâmetros de treinamento são definidos como falsos.
- Uma camada referente a classificação binária do sistema é adicionada no fim da estrutura do modelo do inception v3, que realize um *average pooling*.
- Ao fim da rede é adicionada uma camada para agregar os dados em um único output que determinará a previsão da classificação binária. A função de ativação utilizada é a *sigmoid*.
- Com a rede finalizada são estabelecidos os parâmetros de compilação do modelo. Optou-se por utilizar o RMSprop como algoritmo de otimização, para atualizar iterativamente os pesos da rede durante o treinamento, por ser o que obteve melhor resultado nos testes. Para calcular a taxa de erro (loss) durante o avanço das épocas, optou-se pelo método de Binary Categorical Cross-Entropy, o mais indicado para modelo de classificação binária.

Ao final dessa etapa, o modelo desenvolvido pode ser visualizado na figura 11. Nele podemos ver que apenas as duas últimas camadas tem parâmetros treináveis, o que significa que o aprendizado contido no modelo *inception v3* de identificar padrões permanece intacto.

3.1.2.5 Avaliação

Para avaliar a eficácia de um algoritmos de *machine learning* existem diversas métricas e abordagens, porém iremos avaliar a performance sobre as métricas definidas no capítulo

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 1)                 2049

=================================================================
Total params: 21,804,833
Trainable params: 2,049
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figura 11: Sumário do modelo após aplicar técnica de transferência de aprendizado
Fonte: Autor

anterior de Aspectos Conceituais, na seção sobre sistema de recomendação. Essas métricas são consideradas as melhores para avaliar sistemas de classificação binária (ZHENG, 2015) e também servirão de base para definir quais métricas serão expostas para o especialista na plataforma web.

Após alguns testes e modificações para diferentes conjuntos de dados e critérios, os resultados encontrados foram bem satisfatórios, com a maioria deles tendo uma acurácia acima de 90%. Podemos ver alguns exemplos dessas métricas nas figuras a seguir.

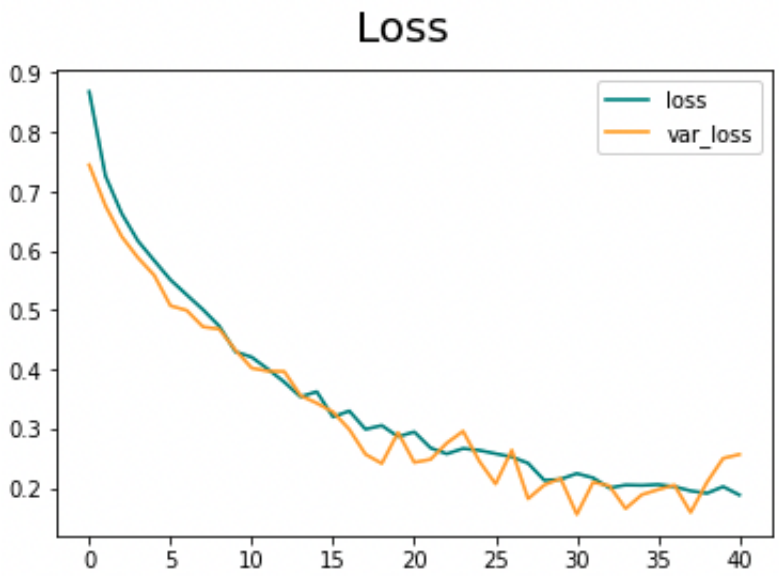

Figura 12: Gráfico de perda do aprendizado estabilizado em menos de 25%
Fonte: Autor

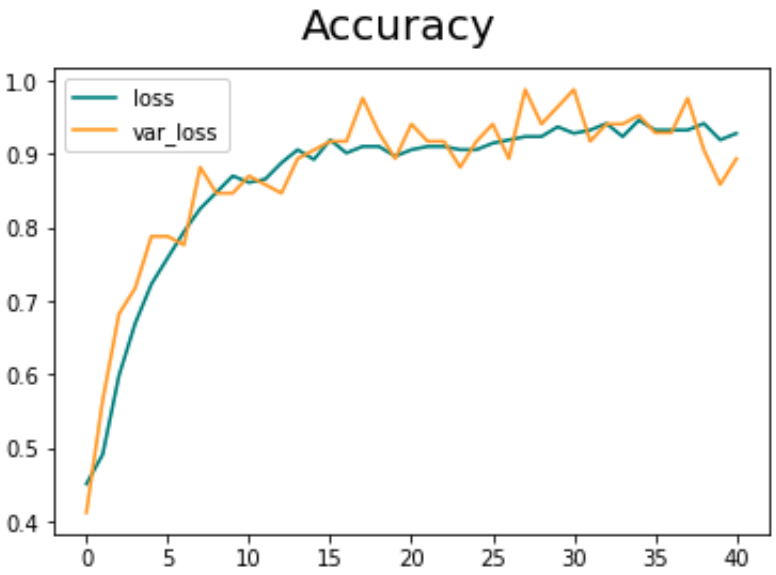

Figura 13: Gráfico de acurácia do processo de treinamento
Fonte: Autor

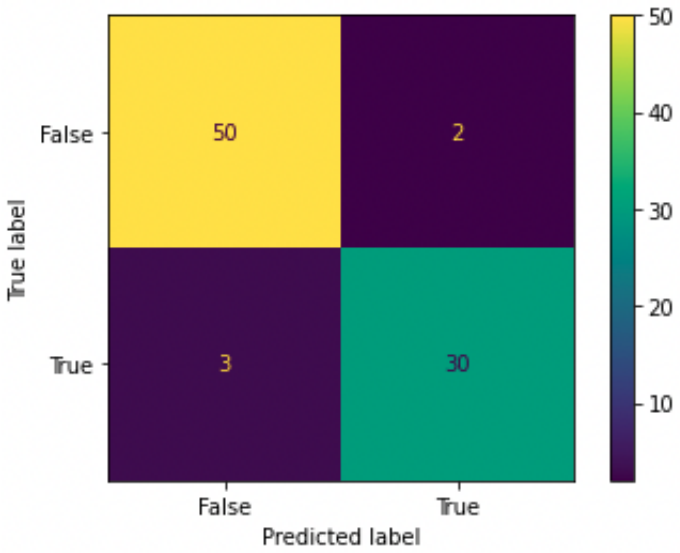

Figura 14: Matriz convolucional para este usuário teste treinado.
Fonte: Autor

3.1.2.6 *Deploy*

A etapa de *deploy* do algoritmo será mais enfatizada na seção 4, Implementação, onde será exemplificado como foi feita a partir dos sistema de recomendação a conexão entre a rede neural e a aplicação.

Porém nessa etapa obtive diversas dificuldades para hospedar o algoritmo de forma que ele fosse de fácil acesso e plausível de se automatizar junto do sistema. Sendo assim, ele foi mantido em seu ambiente de desenvolvimento onde já está em uma tecnologia *cloud*

(Google Colab) e seria de fácil acesso para o *stakeholder* especialista.

As vantagens de utilizar esse sistema é que, na configuração atual, ele não gera custos para a aplicação, porém sua performance é bem mais lenta que a esperada.

3.1.2.7 Aplicação da rede neural no sistema de recomendação

Dado que as reações são binárias, as imagens são classificadas também de forma binária, podendo ser consideradas como curtidas (*like*); o que significa que o usuário gostou daquele perfil, ou como "descurtidas"(*dislike*); o que significa que o usuário não gostou daquele perfil. Já que a reação se aplica a um perfil que pode ter múltiplas imagens de um mesma pessoa, todas as imagens desta pessoa são associadas a essa reação. Não há uma distinção de qual imagem de um perfil o usuário preferiu mais ou menos pois não é o comportamento padrão em aplicativos de relacionamento e não foi identificado durante a etapa de levantamento de funcionalidades do projeto.

A classificação binária gerada pela rede neural pode ser considerada uma pontuação associada a uma imagem de uma pessoa da plataforma e será denominada de *likeability*. Sendo assim, quanto maior a *likeability* de uma imagem, maior a chance do usuário que forneceu os dados para treinar a rede terá de gostar dessa imagem.

Essa métrica que será utilizada no sistema de recomendação como o critério de recomendação. Logo, as pessoas que serão recomendados para o usuário serão as que tiverem a maior *likeability* no geral em suas imagens.

3.2 Requisitos funcionais

Tendo em vista o processo de desenvolvimento que determinou as funcionalidades do sistema e o objetivo a ser atingido pelo projeto, define-se os seguintes requisitos funcionais para o sistema:

- **Cadastrar e se autenticar na plataforma**: O sistema deve ser capaz de cadastrar novos usuários na aplicação, garantir a segurança das requisições e permitir a autentificação de usuários em sua conta individual.
- **Avaliar perfis**: O usuário deve conseguir dar "curtidas" ou "descurtidas"nos outros usuários do banco e ver a imagem dos perfis que ele precisa avaliar.

- **Treinar um modelo capaz de classificar imagens**: O sistema deve ser capaz de utilizar as informações de avaliação do usuário para treinar um modelo de aprendizado de máquina capaz de classificar imagens.

- **Informar os resultados**: Tanto o usuário quanto o especialista devem ter acesso aos resultados de treinamento do modelo. O usuário por não ter o conhecimento necessário será apenas informado de um resumo, enquanto o especialista deve ter acesso completo aos dados de *performance*.

- **Aprovar ou recusar um modelo e recomeçar o processo**: O especialista deve ser capaz de aprovar ou recusar o desempenho de um modelo conforme seus dados apresentados na plataforma. Caso o modelo seja rejeitado, o usuário deve ser capaz de tentar novamente o treinamento (que com algumas alterações do especialista pode obter resultados melhores), adicionar mais informações ao sistema (avaliando mais imagens) ou recomeçar o processo do zero, com um conjunto novo de reações.

- **Gerar recomendações personalizadas**: O sistema deve ser capaz de recomendar perfis para usuários que tiveram seus modelos treinados e passaram da avaliação. Essa recomendação será com base na *likeability* do perfil.

3.3 Definição da Arquitetura

As principais visões arquiteturais do sistema foram desenvolvidas seguindo a modelagem RM-ODP. Como foi explicado na seção de Metodologia, do capítulo 1, apenas três da cinco visões foram escolhidas para compor a aplicação final do projeto. Essas visões são as que melhor explicitam o funcionamento do sistema e, agregadas ao processo de desenvolvimento, foram utilizadas para implementação final do sistema.

3.3.1 Visão de negócio

Para exemplificar a visão do negócio do sistema, dados seus principais *stakeholder*, iremos utilizar o modelo BPMN da figura 15.

Dados os principais casos de uso estabelecidos na figura 9, podemos ver como eles se relacionam quando aplicados dentro da arquitetura do sistema.

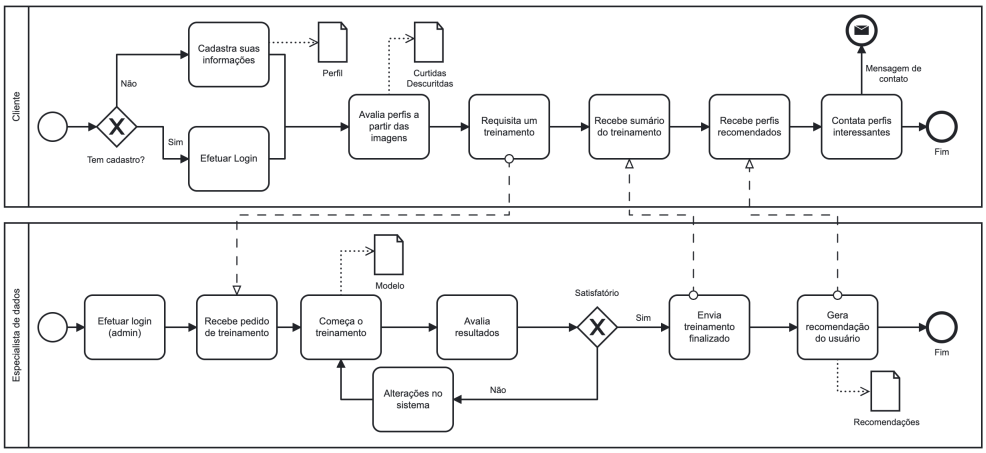

Figura 15: Modelagem em BPMN das atividades realizadas pelos agentes.
Fonte: Autor

3.3.2 Visão da computação

Na visão da computação são definidos os módulos de computação necessários para a execução dos processos do sistema. Esses módulos podem ser encontrados na tabela 2.

Porém, apenas com os módulos e seus serviços não é possível ter uma ideia completa de como eles se relacionam e com que outras parte do sistema eles se comunicam. Por isso, na figura 16 temos a arquitetura de camadas exemplificando como se da a relação do sistema como um todo.

Na arquitetura de camadas, cada camada tem sua função e responsabilidade específica e, normalmente, os sistemas são divididos em 3 principais camadas:

- **Camada de apresentação**: Responsável por lidar com toda a interface do usuário e lógicas de comunicação com navegadores.
- **Camada de negócio**: Responsável por executar operações e fluxos de negócios específicos associados à uma requisição. Aqui que se encontram o módulos do sistema.
- **Camada do banco de dados**: Responsável por armazena os dados manipulados pelo sistema.

Cada camada forma uma abstração em volta da sua necessidade e trabalho à ser realizado dentro de uma requisição em particular.

Módulo	**Serviços**	**Entrada**	**Saída**
Autentificação	Resposável por cuidar da autenticafação do sistema. Garante um nível de segurança as requisições do sistema.	Email e senha	Token de autentificação
Registro de usuário	Cria um novo usuário no sistema.	Usuário	Token de autentificação
Avaliação de usuário	Submete a avaliação do usuário de um perfil para o banco de dados	Id de usuário	Curtida ou descurtida
Gerador de fila	Busca um novo usuário não avaliado para apresentar para o usuário	Id de usuário	Usuário
Mensageiria	Manda emails automatizados para indivíduos	Id de usuário	Email
Resultados	Retorna a informação de treinamento e serve como serviço avaliador dos resultados	Id de usuário	Resultados de treinamento e sumário
Módulo de treinamento	Treina o modelo de aprendizado de máquina	Id do admin e id do usuário	Status de treinamento
Módulo de classificação	Utiliza o modelo de classificação para gerar uma lista de usuários recomendados	Modelo	Recomendações

Tabela 2: Visão da computação.

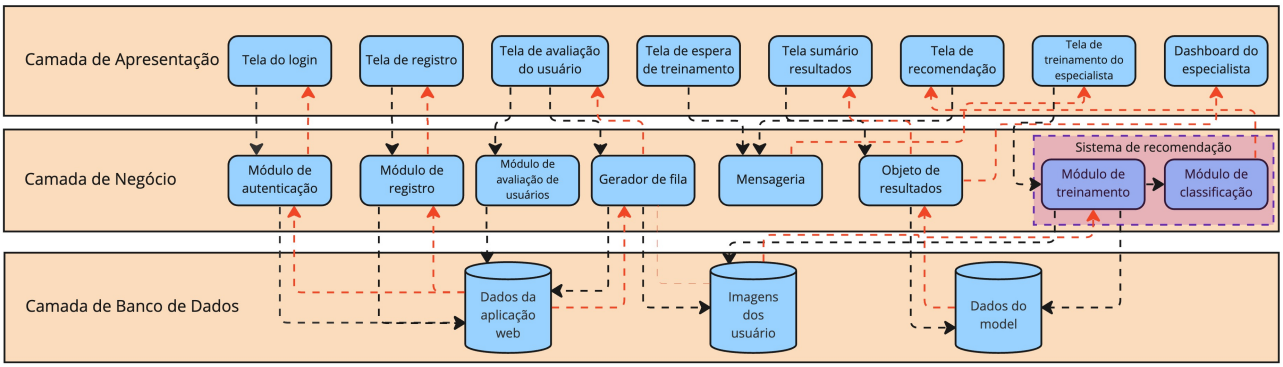

Figura 16: Arquitetura de camadas do sistema.
Fonte: Autor

3.3.3 Visão da tecnologia

De forma a viabilizar a implementação dos componentes especificados na demais visões se faz necessário uma série de tecnologias, cada uma com sua responsabilidade dentro do sistema.

As tecnologias utilizadas serão aprofundadas na seção seguinte, onde será justificado o uso de cada tecnologia em particular e sua funcionalidade específica no sistema. A figura 17 ilustra como essas tecnologias se relacionam e em qual parte da aplicação elas se encontram.

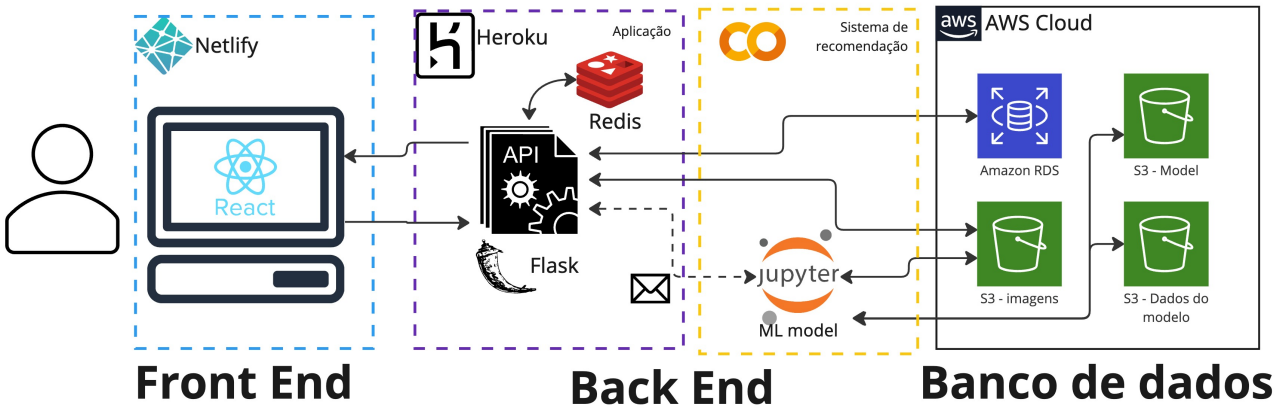

Figura 17: Arquitetura do sistema considerando as tecnologias utilizadas.
Fonte: Autor

3.4 Tecnologias Utilizadas

As principais tecnologias que foram aplicadas no sistema levem em consideração as restrições estabelecidas pelos requisitos funcionais e não funcionais além das arquiteturas

definidas na seção anterior. Levando em conta a estrutura apresentada na figura X e as tecnologias ilustradas na figura Y, um breve resumo do que são essas tecnologias será apresentado, além do porquê da escolha destas para a implementação do sistema.

3.4.1 Frontend

O *Frontend* é a parte da aplicação que se comunica diretamente com os clientes e *stakeholders* do produto. São todos os componentes, conteúdos, textos e interações que um usuário vê e interage ao acessar uma URL ou abrir um aplicativo. Na arquitetura de camadas, ele equivale a camada de apresentação e existem diferentes tecnologias que podem ser aplicadas para desenvolver esse sistema.

3.4.1.1 JavaScript

Javascript é uma linguagem de programação orientada a objetos extremamente flexível e capaz de implementar itens complexos em páginas web. Considerada uma linguagem de script do lado do cliente, junto do HTML e do CSS é uma das tecnologias essenciais da *World Wide Web* (WWW). Esse fator, aliado à facilidade de uso e familiaridade de desenvolvimento, foram os principais motivos que resultaram na sua escolha como linguagem base do *frontend*.

3.4.1.2 React

O React é uma biblioteca de JavaScript *open source* utilizada para criar aplicações web e suas interfaces gráficas com o usuário. Se baseia no conceito de componentes encapsulados que gerenciam seu próprio estado e é extremamente fácil e intuitivo de se utilizar. Tem um grande suporte da comunidade com muitas bibliotecas auxiliares por ser uma tecnologia em alta de desenvolvimento.

As razões de escolher esse *framework* como núcleo do *frontend* são: sua facilidade de implementação, clareza no gerenciamento de testes, simplicidade de aporte para serviços de nuvem e por ter suporte global nos aparelhos. Apesar de não ser uma tecnologia destinada a aplicativos de celular (como o *Tinder* e outras empresas dessa área de relacionamento são), a camada de apresentação do sistema serve apenas o propósito de emular as funcionalidades principais de um aplicativo de namoro dentro de um contexto maior de uma arquitetura integrada de recomendação.

Caso a meta do projeto fosse criar uma empresa nesta área, capaz de trazer uma

proposta diferente para o mercado de relacionamento, provavelmente a tecnologia utilizada seria adaptada para ser implementada em aparelhos móveis, já que a maioria do mercado se encontra nesse meio (GRAND-VIEW-RESEARCH, 2020).

3.4.1.3 Netlify

Netlify é uma empresa de tecnologia de automação que oferece serviços de hospedagem de web sites de graça. Seus serviços ajudam a simplificar o processo para desenvolvedores implantarem e hospedarem seu site online, utilizando o próprio repositório do GitHub e executando um processo de compilação para pré-renderizar todas as páginas da aplicação.

Esse serviço foi escolhido para ser utilizado como tecnologia de nuvem do *frontend*, pois ele não tem custos, é fácil de utilizar, consegue prover uma URL parcialmente customizada e tem um versionamento simples de debugar erros. Com ele, foi registrado o domínio oficial da aplicação, que pode encontrado em: https://perfect-match-tcc.netlify.app

3.4.2 Backend

O *Backend* é a estrutura da aplicação que permite a operação do sistema. Ele é responsável pro toda lógica "por detrás dos panos"que ocorre no programa, possibilitando a comunicação e manipulação de dados entre o banco de dados e o cliente além de cuidar da parte de segurança do sistema. Na arquitetura de camadas, ele equivale a camada de negócio e no modelo cliente-servidor, ele normalmente é considerado o servidor.

3.4.2.1 Python

O Python é uma linguagem de programação muito adotada para aplicações de aprendizado de máquina e desenvolvimento de dados dada a sua facilidade de compreensão, versatilidade de funções e suporte bibliográfico de usuários. Por ser uma linguagem de alto nível interpretada, ela facilita muito a implementação de algoritmos por usuários novos na área de ciência de dados e sua tipagem forte e dinâmica auxilia na correção e verificação de erros.

Porém o principal motivo pela escolha de tal tecnologia com certeza é devido ao suporte de inúmeras ferramentas que auxiliam muito o desenvolvimento de algoritmos pré treinados e visualização de dados que auxiliam na verificação dos modelos desenvolvidos. O suporte de bibliotecas e frameworks como o Tensorflow são essenciais para o desenvolvimento de tais projetos e permitem uma vasta possibilidade de soluções de forma simples

e intuitiva.

3.4.2.2 TensorFlow

O TensorFlow é uma plataforma open-source desenvolvida em Python dedicada ao aprendizado de máquina. A plataforma tem um conjunto flexível de ferramentas, bibliotecas, *datasets* e recursos da comunidade que permite aos desenvolvedores criar e implementar facilmente soluções baseadas em *machine learning*.

3.4.2.3 Keras

Keras é uma API de *deep learning* projetada para seres humanos conseguirem interagir mais facilmente com o framework do Tensorflow. Ela minimiza o número de ações do usuário necessárias para casos de uso comuns e fornece mensagens de erro claras e acionáveis, o que facilita muito o desenvolvimento de algoritmos de aprendizado de máquina.

3.4.2.4 Google Colab

O Google Colab é um produto da Google Research que permite qualquer pessoa escrever e executar códigos Python por meio dos navegadores. Ele é especialmente adequado para tarefas de aprendizado de máquina e análise de dados já que é um serviço de Jupyter notebook hospedado que não requer qualquer tipo de configuração para uso, além de ser gratuito.

Esse produto foi escolhido para hospedar o sistemas de recomendação, que comporta não só a função de aprendizado de máquina, mas também a função de classificação dos usuários, pois ele não tem custos de processamento, é de fácil acesso para compartilhamento de código e seria um ambiente familiar para o especialista de dados, único *stakeholder* que teria acesso direto a ele. Além disso, ele facilitaria o processo de fazer alterações e ajustes no modelo para obter resultados melhores, uma vez que o código está aberto para quaisquer modificações necessárias.

3.4.2.5 Flask

O Flask é um *micro framework* escrito na linguagem Python voltado para o desenvolvimento web. Sua estrutura acompanha o mínimo de componentes possível, para tornar seu uso rápido e escalável. (Flask, 2022). Esse web *framework* foi escolhido pois ele é ideal

para realizar operações de *backend* de websites de forma simples, faceis de compreender e leves de rodar.

3.4.2.6 Inception v3

O Inception v3 é um modelo de reconhecimento de imagem amplamente usado que demonstrou atingir uma precisão superior a 78,1% no conjunto de dados ImageNet (dataset com mais de dez milhões de URLs de imagens rotuladas). O modelo em si é composto de componentes simétricos e assimétricos, incluindo convoluções, agrupamentos médios, agrupamentos máximos, concatenações e camadas totalmente conectadas. A normalização em lote é usada extensivamente em todo o modelo e aplicada às entradas de ativação. A perda é calculada por meio da função softmax. A figura 18 ilustra seus componentes da rede neural.

A escolha deste modelo como base para desenvolver o algoritmo de avaliação de preferência do usuário se dá pelo fato de ser um dos modelos mais bem consolidados do mercado e já ter aplicações parecidas de avaliação de perfil de usuários de aplicativos de relacionamentos baseados nele (BRERETON, 2019).

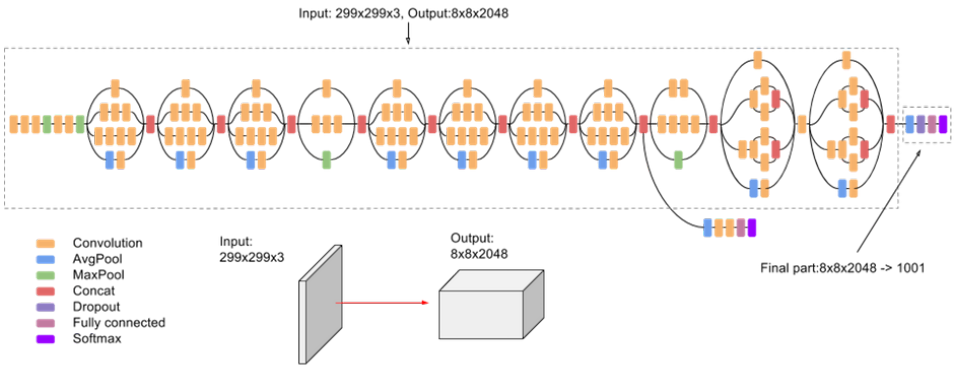

Figura 18: Arquitetura da rede neural Inception v3
Fonte: Google Cloud

3.4.2.7 Heroku

O Heroku é uma plataforma como serviço (PaaS) em nuvem baseada em contêiner. Esta é uma das plataforma mais disseminada no mercado, utilizada para implantar, gerenciar e dimensionar aplicativos que podem ser hospedados em uma estrutura de nuvem. Ele é extremamente simples de usar, flexível e até o dia 28 de Novembro de 2022 ele dispunha de um plano gratuíto, o que o tornava muito atrativo para pequenas empresas ou projetos.

Esta plataforma foi utilizada no sistema para fazer a implantação do servidor em Flask de forma a disponibilizar uma API pública para uso. O Heroku, em específico, foi escolhido pois ele é simples para escalar e gerenciar a aplicação do *backend*, tem fácil monitoramente de *bugs* e *crashes* e dispõe de créditos de graça para uso para estudantes.

3.4.3 Armazenamento de dados

Existem alguns tipos de banco de dados diferentes na aplicação, alguns mais dedicados para armazenar informações de segurança, outros para armazenar informações da aplicação e outros para armazenar informações do sistema de recomendação. Todos estes, apesar de se comunicarem com serviços diferentes e especializados do sistema, são considerados a camada de banco de dados da arquitetura.

3.4.3.1 PostgresSQL

O PostgresSQL é um sistema de banco de dados relacional avançado de código aberto que suporta tanto consultas de SQL (relacionais), como consultas de JSON (não relacionais). Ele é banco de dados altamente tolerante a falhas, com recursos robustos e altamente escalável, por isso foi escolhido como banco de dados fundamental da aplicação. No projeto ele contém todas as informações dos usuários, sejam estes os clientes principais ou os especialistas, as informações de status dos modelos e as informações de interação com a aplicação (curtidas/descurtidas).

3.4.3.2 Redis

O Redis (*remote dictionary server*) é uma tecnologia de armazenamento de estrutura de dados no formato chave-valor na memória que possui código aberto. Ele pode ser utilizado como banco de dados, *cache* e *message broker* para diferentes estruturas de dados. Ele é um ótima opção para armazenamento de informações importantes e restritas dos usuários, como os *tokens* individuais de autentificação que são utilizados múltiplas vezes para validação das chamadas da API. Por ser simples de configurar e já estar embutido na plataforma Heroku como uma opção de nuvem ele foi escolhido para essa parte do sistema.

3.4.3.3 S3

O Amazon Simple Storage System (Amazon S3) é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e desempenho líderes do setor (AWS, 2022b). Utilizando os serviços da AWS (*amazon web services*) é possível gerenciar, configurar acessos e monitorar o banco de dados de forma completamente remota, simples e *serverless*.

Esse serviço foi escolhido para armazenar as informações do sistema que não são estruturadas e que podem ter mais espaço de memória, como são os casos das imagens dos usuários, o arquivo do modelo treinado e as informações de treinamento. Dessa forma, essas informações podem ser guardadas de uma forma escalável, sem custos adicionais e com um ótimo tempo de resposta já o que o banco é especializado para este tipo de funcionalidade.

3.4.3.4 Amazon RDS

O Amazon Relational Database Service (Amazon RDS) é uma coleção de serviços gerenciados que facilita a configuração, operação e escalabilidade de bancos de dados na nuvem (AWS, 2022a). Ele foi utilizado para hospedar o banco de dados em PostregSQL da aplicação, pois ele é simples de utilizar através dos sistemas da AWS, tem alta disponibilidade e confiabilidade para cargas maiores de trabalho, além de oferecer um plano grátis até um certo patamar de dados que a aplicação mvp não ultrapassa.

4 IMPLEMENTAÇÃO

Neste capítulo será apresentado como foi implantada a arquitetura desenvolvida nos capítulos anteriores. Primeiro será explicado como foi dada a implementação do sistema de recomendação, com dados de teste e validação de métricas. Em seguida será demonstrado a implementação da arquitetura como um todo, exemplificando o fluxo já finalizado do usuário. Por fim será feita a validação com um usuário específico.

4.1 Implementação do sistema de recomendação na infraestrutura

Com o sistema de recomendação finalizado e funcionando para usuários do banco de dados testados dentro do ambiente de desenvolvimento do Google Colab, foi necessário estabelecer as conexões finais que permitiram que as funcionalidades estabelecidas para o especialista estivessem de fato no ar.

Para isso duas conexões tiveram que ser feitas:

1. Os dados de entrada do modelo devem vir diretamente das tabelas de *likes* e *dislikes* do banco de dados principal que contém as avaliações do usuário. Com isso, o modelo conseguiria montar o conjunto de dados que corresponde as imagens avaliadas pelo usuário.

2. Os dados de saída do sistema, tanto as informações de performance, quanto as recomendações precisam ser diretamente conectados com um banco de dados que o sistema tenha acesso. Sendo assim, as informações de performance por serem no formato json, são salvas em um banco do s3 de dados do modelo a qual a aplicação irá consultar. Já os usuários recomendados são salvos em uma tabela diretamente no banco da aplicação chamada *recommendation*. Para não haver uma sobrecarga do banco, apenas os top 10 usuários recomendados são salvos nessa tabela.

Com as conexões estabelecidas os sistemas já estavam se comunicando e integrados. Alterações feitas na plataforma, seja dar mais reações, exigir um novo modelo ou cancelar um conta iriam causar impacto no aprendizado da máquina e modificações feitas no algoritmo iriam gerar resultados diferentes para a recomendação do usuário. Basta o especialista receber a chamada do sistema que existe um usuário precisando de treinamento do seu modelo que, ao inicializar o processo manualmente na plataforma do Google Colab atualizando o dado de execução para utilizar id do usuário, todas as informações do sistema seriam atualizadas.

4.1.1 Testes

Concluído a implementação do sistema de recomendação na arquitetura principal alguns testes foram conduzidos para observar o funcionamento do projeto.

Primeiro foram conduzidos testes para determinar se as conexões estavam de fato funcionando e o sistema como um todo estava responsivo. Em seguida, foram conduzidos testes para determinar os tempos de execução desse sistema, uma vez integrado.

O sistema de recomendação tem duas fases distintas que consomem um certo tempo para serem realizadas. A primeira é a própria fase de treinamento do modelo de rede neural, que pode variar conforma a quantidade de dados fornecido para o treinamento. Quanto mais dados, maior o tempo de demora.

A segunda é a fase de recomendação do sistema, na qual o modelo gerado é utilizado para categorizar todos os outros perfis não avaliados pelo usuário do sistema. Essa fase, em contraposição a anterior, demora menos tempo quanto mais imagens foram avaliadas pelo usuário. Porém, em comparação, essa fase representa bem menos tempo de execução que a anterior.

Na tabela 3 vemos um comprativo de tempos para diferentes patamares de imagens. Apesar de serem processos bem longos e demorados existem algumas soluções possíveis para diminuir esses valores:

- **Hospedar o sistema de recomendação em uma plataforma mais potente do que a Google Colab**. Apesar de ser uma opção de graça, o poder computacional disponível na plataforma é extremamente limitado o que é a principal razão do tempo de espera do sistema ser tão grande. Utilizando outras plataformas pagas, como por exemplo a Amazon, e contratando um serviço computacional com mais GPUs e RAM esse tempo de espera poderia cair em até 10 vezes.

Total de imagens	Tempo de treinamento	Tempo de recomendação
40	42 min	13 min
309	1h03 min	12 min
754	1h46 min	10 min

Tabela 3: Tempo de execução do sistema de recomendação para diferente *thresholds* de imagens.

- **Utiliza a recomendação apenas em parte dos usuários**. Como o objetivo do sistema era achar o "par ideal", então o sistema de recomendação avalia TODOS os perfis que não foram avaliados pelo usuário. Isso apesar de garantir encontrar o melhor par daquele banco de dados é extremamente ineficiente. Em aplicativos que estão no mercado, como *Tinder*, alguns filtros são aplicados quando usuários são recomendados, como por exemplo, a distância entre eles.

- **Utilizar uma pipeline de dados para pré processar informações**. Na arquitetura atual todos os dados são processados simultaneamente toda vez que o treinamento de um usuário é iniciado. Deixar informações salvas de treinamento anteriores e apenas atualizar com informações novas ou criar estruturas mais robustas de pipeline de dados poderia reduzir significativamente o tempo, principalmente para re-treinamentos.

4.2 Implementação da arquitetura

Nesse capítulo será demonstrado como ficou a implementação final do projeto, considerando a plataforma web e suas conexões.

4.2.1 Estrutura do banco de dados final

Com a finalização do projeto é possível analisar como ficou a arquitetura do banco de dados do sistema dado os requisitos mínimos considerados e o diagrama de casos de uso. Na figura 19 vemos o diagrama ERD (*Entity Relationship Diagram*) do banco principal da aplicação, o banco PostgresSQL. Nele temos duas tabelas distintas que comportam as informações dos *stakeholders* da aplicação: a tabela de *admins*, dedicada aos usuários denominados de "especialistas", e a tabela de *users* dedicada a todos os outros usuários do sistema.

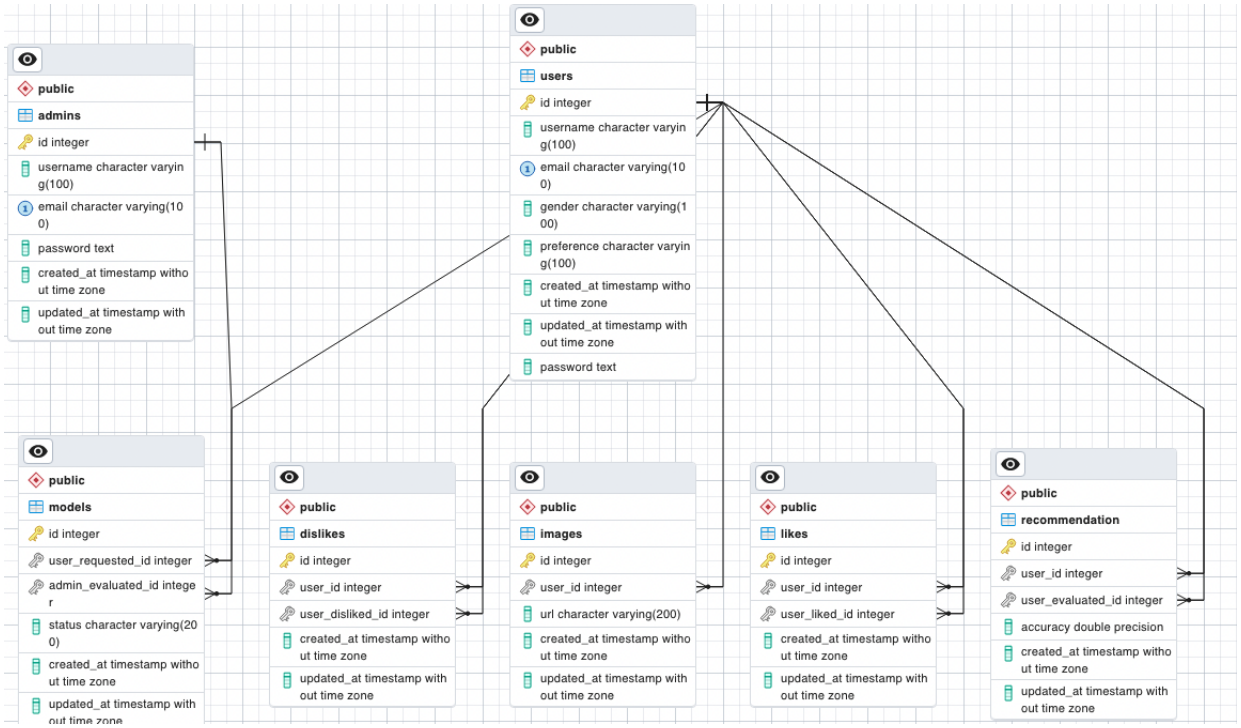

Figura 19: Diagrama ERD do banco de dados PostgresSQL
Fonte: Autor

4.2.2 Fluxo final do usuário

Antes de se apresentar o fluxo passo a passo de um usuário no sistema, a figura 20 apresenta uma visão ampla de todas as telas desenvolvidas no projeto.

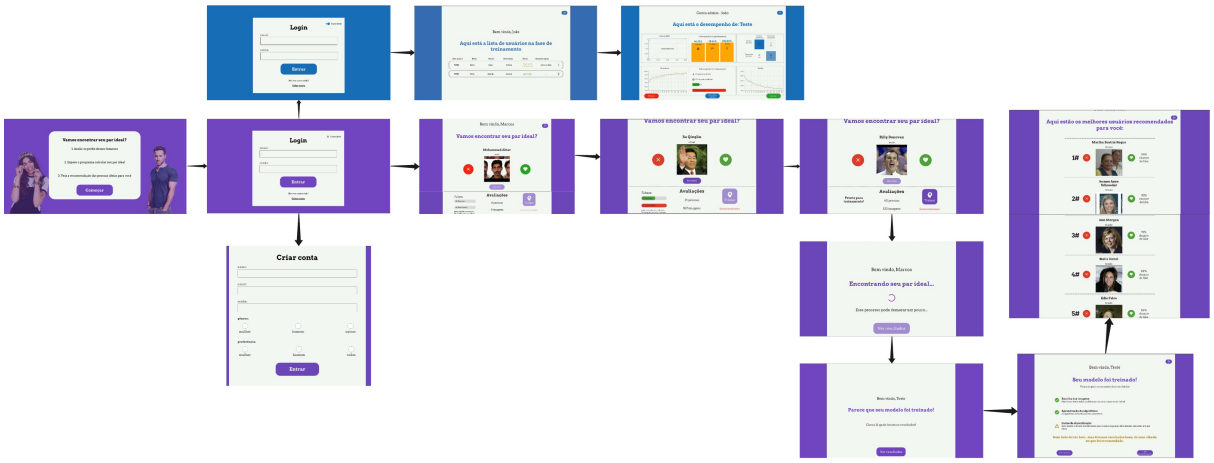

Figura 20: Diagrama de todas tela do perfect match
Fonte: Autor

O usuário começa sua jornada pelo perfect-match se deparando com a tela de *home* (figura 21). Nela é apresentada um passo a passo de como funcionará a experiência do usuário.

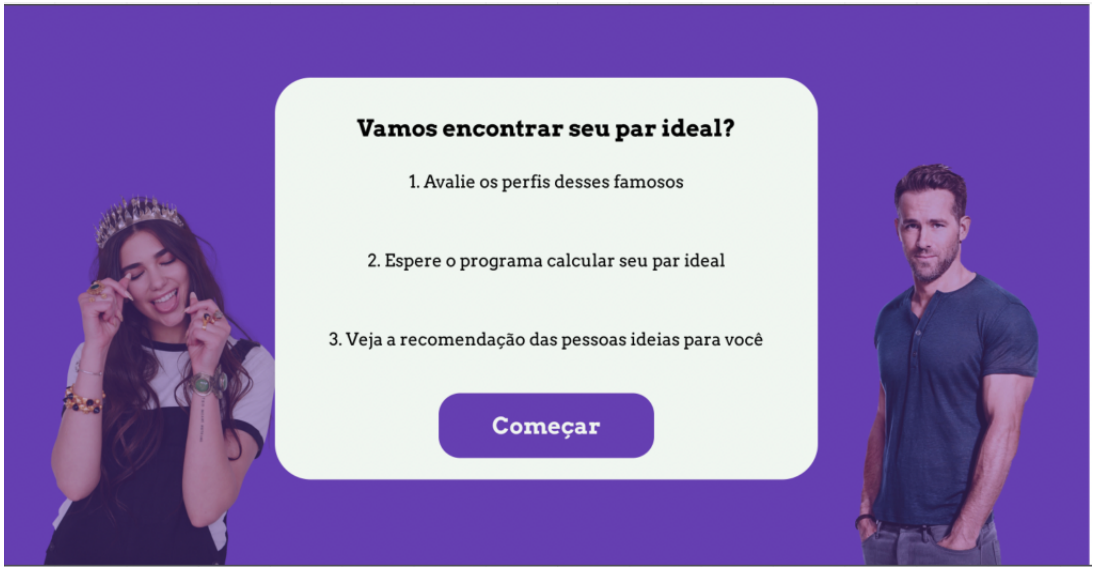

Figura 21: Tela de home
Fonte: Autor

Em seguida ele deverá se autenticar para acessar o sistema na tela de login (figura 22). Nessa tela que o fluxo do usuário comum se distância do especialista já que eles irão acessar partes distintas da plataformas dada suas credenciais. A autentificação do sistema é feita a partir de *tokens* salvos no *backend* no Redis, o que serve como um grau de proteção para ataques maliciosas.

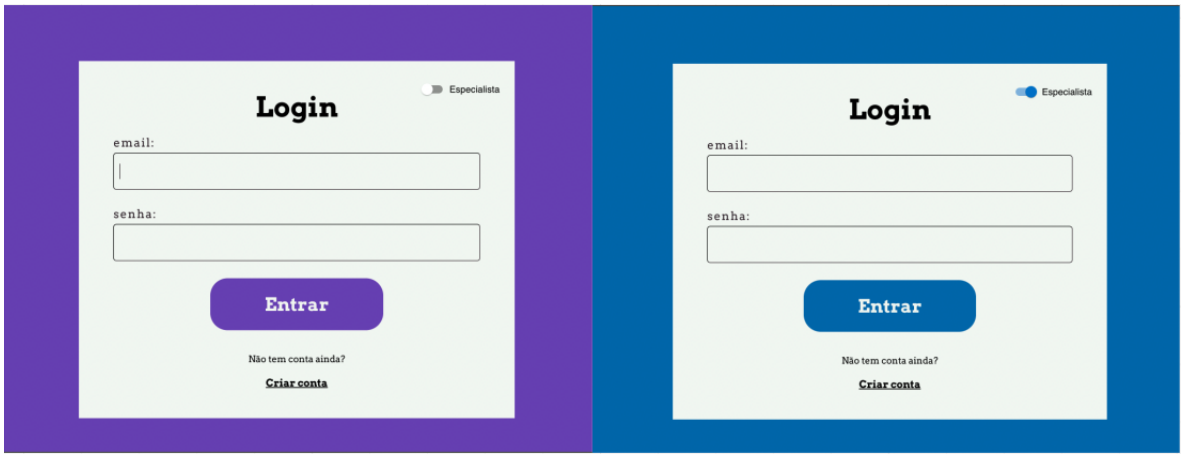

Figura 22: Comparativo entre tela de login do usuário e tela de login do especialista
Fonte: Autor

Acompanhando o fluxo do principal cliente do sistema, o usuário, temos a tela central do sistema (figura 23), na qual o usuário irá interagir por mais tempo. No canto direito superior da tela o usuário pode sempre sair da aplicação, clicando no botão de *logout*. Na parte central da tela temos o perfil de quem ele está avaliando com: o nome, identificação de gênero, fotos e as opções de *like* e *dislike* assim como visto no modelo de aplicativos de

relacionamento na figura 6. Na parte inferior da tela temos quantas pessoas e imagens já foram avaliadas, quantas faltam ser avaliadas e se é possível ir para a etapa de treinamento já. Existe um limite definido pela especificação do sistema de recomendação de no mínimo 40 imagens (20 de uma classe e 20 de outra) para que os resultados sejam satisfatórios.

Figura 23: Tela principal do usuário
Fonte: Autor

Uma vez que o usuário avaliar perfis até se sentir confortável para receber recomendações (e já tiver o valor mínimo necessário das duas categorias). Ele pode requisitar o treinamento do seu modelo. Uma vez que o modelo é treinado o usuário consegue ver um breve resumo dos resultados dele (figura 24).

Após ver o resumo o usuário pode finalmente chegar ao propósito da aplicação, a tela de recomendação de usuários (figura 25). Nela ele vê as top 10 pessoas recomendadas para ele com base em sua preferência e com a *likeability* delas calculada. O usuário pode reagir a essas pessoas nesta tela também, mas, ao invés de servirem como dados para recomendação, essa ação ativa o módulo de mensageira do sistema para entrar em contato com o e-mail cadastrado do perfil avisando que há alguém interessado.

Em contrapartida e esse fluxo principal, temos o fluxo do especialista. Após ele se autenticar na plataforma ele acessará a sua tela central (figura 26). Nela o especialista consegue ver o status de todos os modelos do sistema. Os modelos já avaliados são

Figura 24: Tela de sumário do treinamento para o usuário
Fonte: Autor

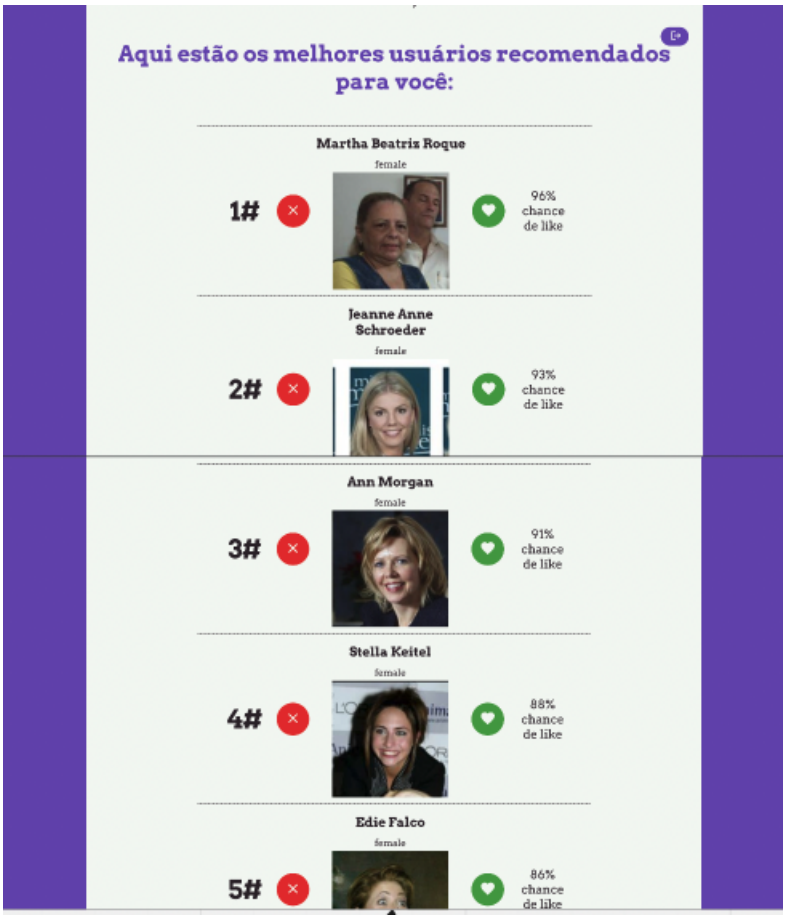

Figura 25: Tela onde o usuário pode ver as recomendações do sistema
Fonte: Autor

ordenados por último e os que estão pendentes são ordenados primeiro por tempo de espera. Além do status, o especialista consegue ver as informações do usuário, inclusive

o id deles, já que essa informação é necessária para ele iniciar o treinamento no Google Colab.

Nesse momento, se houver algum treinamento pendente que ainda não foi iniciado, o especialista deverá entrar na plataforma da Google onde ele começaria o processo do sistema de recomendação.

Figura 26: Tela principal do especialista
Fonte: Autor

Ao final do processo de treinamento, o especialista poderia acessar o *dashboard* daquele modelo gerado na plataforma (figura 27). Lá ele conseguiria ver todas as principais métricas de como foi o treinamento do modelo e qual foi sua performance. Com essas informações ele seria capaz de avaliar se modelo está aprovado ou recusado. Caso esteja recusado, ele mesmo poderia acessar o Google Colab para fazer ajustes no sistema e rodar o processo de novo. Se o erro for originado por um motivo que não é do controle dele (exemplo: falta de imagens), ao reprovar o usuário a tela de sumário dele irá refletir esse feedback.

Figura 27: Tela de dashboard e avaliação do treinamento
Fonte: Autor

PARTE III

CONCLUSÃO

5 CONSIDERAÇÕES FINAIS

Nesta seção são apresentadas as conclusões alcançadas com o projeto de formatura, se houve o cumprimento dos objetivos propostos, quais as contribuições deixadas e as perspectivas de continuidade em trabalho futuros.

5.1 Cumprimento dos objetivos

O principal objetivo deste trabalho era desenvolver uma arquitetura capaz de integrar um sistema de recomendação que utiliza aprendizado de máquina com uma aplicação web. Essa arquitetura seria implementada em um sistema online que simula um website de relacionamento em um modelo MVP de uma aplicação real. Com isso, seria possível demonstrar quais metodologia e processos foram escolhidos para desenvolver uma arquitetura que fosse capaz de alinhar as necessidades do ponto de vista do negócio e suas demandas de dados.

Considerando os requisitos mínimos estabelecidos e o que foi desenvolvido é possível dizer que o projeto conseguiu se adequar a essas premissas e cumprir o objetivo da monografia. A plataforma desenvolvida, *perfect match*, tem todas as funcionalidades propostas e consegue simular bem uma web site de relacionamento online.

O sistema de recomendação tem bons resultados na maioria dos casos, principalmente quando consideramos casos com mais imagens utilizadas no treinamento em um padrão que é mais abundante e distinto no banco (exemplo: mulheres loiras). Para os resultados insatisfatórios o sistema consegue não só informar os problemas detectados para o especialista, quanto dar um *feedback* para o usuário caso a medida que devesse ser tomada parta da ação dele de avaliar mais perfis.

Por fim, apesar da integração do sistema de recomendação e a aplicação não estar totalmente automatizada, pode se dizer que existe sim um alto grau de relação entre esses sistemas. A arquitetura foi desenvolvida seguindo uma metodologia bem definida de eta-

pas, garantindo que os principais problemas normalmente encontrados no desenvolvimento de projetos como esse fossem evitados.

5.2 Contribuições

A grande contribuição do trabalho foi a metodologia aplicada para desenvolver a arquitetura proposta que possibilitou a integração de um sistema de recomendação com uma aplicação web. Seguindo o processo que foi estabelecido, é de se esperar que o projeto desenvolvido ajude a guiar quaisquer pessoas interessadas em adotar, de forma mais simples e impactante, o uso de sistemas inteligentes em aplicações.

Além disso, a arquitetura desenvolvida pode servir como um caso de estudo para projetos futuros que buscam referências de sistemas de recomendações inseridas no mercado de relacionamento online.

5.3 Perspectivas de Continuidade

Considerando as limitações do projeto e pensando em um escopo mais amplo do que um trabalho de conclusão de curso, muitos avanços poderiam ser feitos para aperfeiçoar o sistema em trabalhos futuros. Essas perspectivas de continuidade podem envolver melhorias em parâmetros do sistema, novas funcionalidades e até alterações para arquiteturas mais sofisticadas. Dentre elas, podem se destacar principalmente:

- **Implementação completa das funcionalidades de uma aplicação de relacionamento para o cliente** - O projeto atual focou em desenvolver apenas um modelo MVP de um produto para exemplificar uma arquitetura integrada. Seguindo com o desenvolvimento do site, todas as funcionalidade mais comuns de uma aplicação de namoro poderiam ser implementadas, como: fazer alteração do perfil, adicionar mais fotos e outras informações pessoais, ter um chat dentro da aplicação, ver as pessoas que já te "curtiram", adicionar outros tipos de reação e até implementar algum tipo de monetização. Essas melhorias poderiam inclusive fazer a plataforma se tornar um negócio, caso fosse implementada para dispositivos móveis, teria a chance de ser um bom competidor nesse mercado de relacionamento;
- **Funcionalidades mais complexas para o *stakeholder* especialista com automatizações** - A plataforma desenvolvida para o usuário especialista é relativamente simples, pois ele apenas precisa ter um controle do status dos modelos do

sistema. Porém, se pensarmos em melhorias nessa área poderíamos desenvolver uma plataforma completamente a parte e independente totalmente automatizada para fazer diferentes alterações no modelo. Dessa forma, ao invés do especialista fazer alterações direto nos algoritmos ele teria uma interface única e simples para comandar esses ajustes;

- **Aumentar a complexidade dos parâmetros de treinamento** - O sistema de recomendação atual utiliza apenas as informações da imagem do usuário e um único parâmetro de classificação que pode ser positivo ou negativo (*like* ou *dislike*). Como explicado, isso foi escolhido pois a imagem é o principal critério de avaliação em aplicativos de relacionamento e a reação é o principal parâmetro de avaliação. Contudo, para adicionar maior complexidade para o sistema poderiam ser introduzidos outros parâmetros de avaliação e até mesmo diferentes componentes para caraterizar o que significa a preferência do usuário. Alguns exemplos são: adicionar uma informação textual da biografia do usuário, adicionar *tags* de diferente interesses pessoais (profissões, comidas, atividades, etc), ter uma avaliação específica para cada imagem ao invés do perfil, ter uma avaliação de componentes das imagens (olhos, boca, nariz, etc) e até mesmo usar outros tipos de recomendação como filtragem colaborativa para melhorar o resultado;

- **Implementação de uma arquitetura mais escalável** - A principal falha do sistema atual quando consideramos um produto real que estaria em mercado é o fato da arquitetura não ser completamente escalável. Uma das alterações que poderia ser feita para melhorar o projeto seria implementar uma arquitetura pensada para um contexto maior de milhares de usuários. Para que isso fosse possível, provavelmente a estrutura do algoritmo de aprendizado de máquina teria que ser alterada da seguinte forma: ao invés de haver uma fluxo exclusivo de treinamento para cada usuário, todos os usuários e informações individuais seriam parâmetros de entrada para um único algoritmo treinado apenas uma vez que sofreria ajustes de tempos em tempos e seu modelo seria utilizado como classificador; além disso não haveria uma plataforma dedicada para avaliar cada usuário por um especialista de forma manual e sim um fluxo automatizado de ajustes do modelo com base no re-treinamento e *pipeline* de dados. Uma possível arquitetura desse sistema se encontra na figura 28. Essa arquitetura considera que todos os sistemas estariam hospedados em serviços de *cloud* que estes sim poderiam ser supervisionados por um time de cientista de dados;

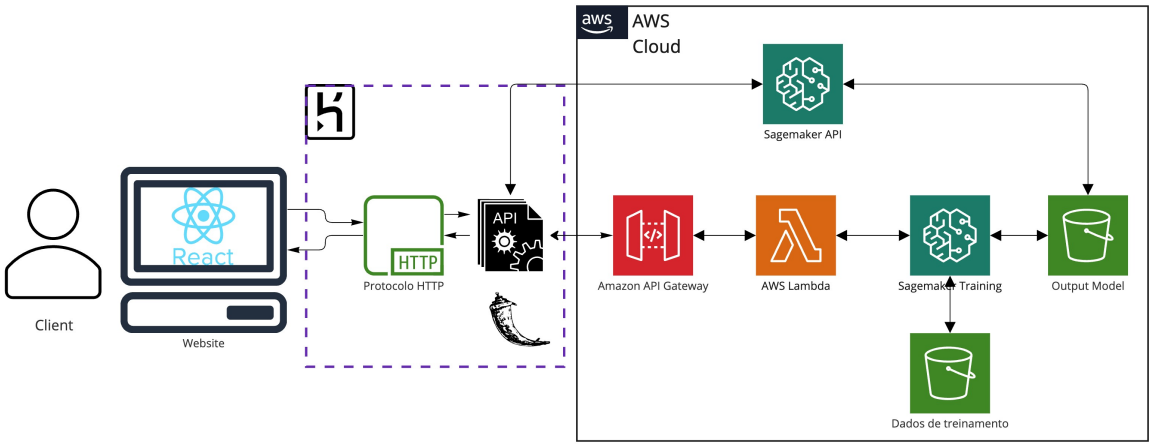

Figura 28: Porposta de arquitetura escalável.
Fonte: Autor

REFERÊNCIAS

ALPAYDIN, E. *Introduction to machine learning (adaptive computation and machine learning series)*. [S.l.]: MIT Press Cambridge, 2014.

AOYAMA, H. *Using AI to Find a Partner on Tinder*. 2020. Disponível em: ⟨https://medium.com/random-daydreams/how-i-used-ai-to-find-a-partner-on-tinder-c59cc0bb154c⟩. Acesso em: 15 maio 2021.

AWS. *Classificação binária*. 2021. Disponível em: ⟨https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/binary-classification.html⟩. Acesso em: 12 outubro 2022.

AWS. *Amazon RDS*. 2022. Disponível em: ⟨https://aws.amazon.com/pt/rds/⟩. Acesso em: 04 setembro 2021.

AWS. *What is Amazon S3?* 2022. Disponível em: ⟨https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html⟩. Acesso em: 04 setembro 2021.

BECERRA, J. L. R. Laboratório de engenharia do produto. In: . [S.l.]: Apresentado como material do curso de Laboratório de Engenharia de Software I em 2019, 2019.

BILBAO, I.; BILBAO, J. Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks. In: *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*. [S.l.: s.n.], 2017. p. 173–177.

BRERETON, A. E. *Teaching a Robot to Swipe on Tinder*. 2019. Disponível em: ⟨https:https://onezero.medium.com/teaching-a-robot-to-like-27c5e8bf6f0⟩. Acesso em: 12 maio 2021.

CARMAN, A. *Hinge's newest feature claims to use machine learning to find your best match*. 2018. Disponível em: ⟨https://www.theverge.com/2018/7/11/17560352/hinge-most-compatible-dating-machine-learning-match-recommendation/⟩. Acesso em: 13 abril 2021.

Flask. *Flask Documentation*. 2022. Disponível em: ⟨https://flask.palletsprojects.com/en/1.1.x/⟩. Acesso em: 01 setembro 2021.

Gokul. *How does Netflix's recommendation engine manage to maintain a low churn rate?* 2020. Disponível em: ⟨https://www.argoid.ai/blog/how-does-netflixs-recommendation-engine-manage-to-maintain-such-a-low-churn-rate-case-study⟩. Acesso em: 16 dezembro 2021.

GRAND-VIEW-RESEARCH. *U.S Online Dating Market*. 2020. Disponível em: ⟨https://www.grandviewresearch.com/industry-analysis/online-dating-market-report⟩. Acesso em: 10 novembro 2022.

IBM CORPORATION. *IBM Global AI Adoption Index 2022*. 2022. Disponível em: ⟨https://www.ibm.com/downloads/cas/GVAGA3JP⟩.

LI, J. *M2M Day 90— How I used Artificial Intelligence to automate Tinder*. 2018. Disponível em: ⟨https://towardsdatascience.com/m2m-day-89-how-i-used-artificial-intelligence-to-automate-tinder-ced91b947e53⟩. Acesso em: 22 junho 2021.

LININGTON, P. F. *Building Enterprise Systems with ODP: An introduction to open distributed processing*. [S.l.]: CRC Press, 2012.

LIU, S. *Personalized Recommendations at Tinder*. 2017. Disponível em: ⟨https://www.slideshare.net/SessionsEvents/dr-steve-liu-chief-scientist-tinder-at-mlconf-sf-2017⟩. Acesso em: 13 abril 2021.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 22, n. 10, p. 1345–1359, 2009.

Tinder Group. *Powering Tinder® — The Method Behind Our Matching*. 2019. Disponível em: ⟨https://www.tinderpressroom.com/powering-tinder-r-the-method-behind-our-matching⟩. Acesso em: 13 abril 2021.

WIERINGA, R. J. *Design science methodology for information systems and software engineering*. [S.l.]: Springer, 2014.

WIRTH, R.; HIPP, J. Crisp-dm: Towards a standard process model for data mining. In: SPRINGER-VERLAG LONDON, UK. *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. [S.l.], 2000. v. 1.

ZHENG, A. *Evaluating machine learning models*. [S.l.]: O'Reilly Media, Inc., 2015.