HENRIQUE VENEZIANO GUISASOLA
RAFAEL CAMARGOS SANTOS

SMARTAD - FERRAMENTA DE MARKETING MOBILE DIRECIONADO

São Paulo
2018

HENRIQUE VENEZIANO GUISASOLA
RAFAEL CAMARGOS SANTOS

SMARTAD - FERRAMENTA DE MARKETING MOBILE DIRECIONADO

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

São Paulo
2018

HENRIQUE VENEZIANO GUISASOLA
RAFAEL CAMARGOS SANTOS

SMARTAD - FERRAMENTA DE MARKETING MOBILE DIRECIONADO

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

Orientador:

Jorge Luis Risco Becerra

São Paulo
2018

RESUMO

Dentro do universo de marketing e propaganda aumentar a conversão de um anúncio se torna uma necessidade cada vez maior, quanto mais eficiente for uma campanha publicitária, maior a conversão de vendas de determinado produto. Aliado a isso tem-se área de marketing mobile crescendo cada vez mais, e a maioria das plataformas de marketing mobile não utilizam de métodos de sugestão direcionada para mostrar propagandas aos usuários de smartphones.

Este projeto visa melhorar o cenário desse problema, ou seja, desenvolver uma plataforma, utilizando tecnologias de inteligência artificial e algoritmos de buscas otimizadas para sugerir propagandas direcionadas a um determinado usuário a partir das informações de localização e das fotos que o usuário tiradas dentro de aplicativos mobile.

Para isso as fotos dos usuários são classificadas dentro de categorias pré determinadas com uma precisão de 80% e com essas informações, utilizando algoritmos de busca, propagandas que mais se aproximam do perfil do usuário são buscadas e enviadas para o usuário.

Palavras-Chave – marketing mobile, inteligência artificial, propagandas direcionadas, algoritmos de busca

ABSTRACT

Within the universe of marketing and advertising increasing the conversion of an ad becomes an increasing need, the more efficient an advertising campaign, the greater the conversion of sales of a particular product. Associated to this is the mobile marketing area is growing more and more, and most mobile marketing platforms do not use suggestion directed methods to show advertisements to smartphone users.

This project aims to improve the scenario of this problem, that is, to develop a platform, using artificial intelligence technologies and optimized search algorithms to suggest targeted advertisements to a given user using the location information and photos that the user takes within mobile applications.

For this, user photos are classified within predetermined categories with an accuracy of 80% and with this information, using search algorithms, advertisements that most closely approximate the user profile are searched and sent to the user.

Keywords – mobile marketing, artificial intelligence, targeted advertisements, search algorithms

LISTA DE FIGURAS

1 Percentual de pessoas utilizando smartphones no mundo 11

2 Gráfico de iterações do algoritmo K-Means 18

3 Processo de negócio do projeto 32

4 Diagrama de Entidade e Relacionamento 34

5 Diagrama da NIST SP 800-145 35

6 Diagrama da arquitetura em camadas do sistema 37

7 Diagrama da arquitetura de serviços 40

8 Diagrama do fluxo do módulo de classificação 43

9 Diagrama do fluxo do módulo de sugestão 48

10 Fórmula para o cálculo do score de um documento no Elasticsearch 53

11 Tela do feed de propagandas 54

12 Tela de upload de fotos 55

13 Tela de galeria de fotos 57

SUMÁRIO

Parte I: INTRODUÇÃO **9**

1 Introdução **10**

1.1 Objetivo 10

1.2 Motivação 10

1.3 Justificativa 11

1.4 Estrutura do trabalho 12

Parte II: DESENVOLVIMENTO **13**

2 Aspectos Conceituais **14**

2.1 Marketing Digital 14

2.1.1 Segmentação de mercado 14

2.1.2 Marketing Invisível 15

2.2 Inteligência Artificial 15

2.3 Machine Learning 16

2.4 Algoritmos de Classificação 17

2.4.1 K-Means 17

2.5 Deep Learning 18

2.6 Visão Computacional 19

2.7 Cloud Computing 19

2.7.1 IaaS 19

2.7.2 FaaS 20

2.8 Serverless 20

3 Tecnologias Utilizadas **21**

3.1 Elasticsearch 21

3.2 API 21

3.3 HTTP 22

3.4 Javascript 23

3.5 Android 23

3.6 AWS 24

3.6.1 Rekognition 24

3.6.2 Sagemaker 25

3.6.3 Lambda 25

3.6.4 DynamoDB 26

3.6.5 API Gateway 26

3.6.6 S3 26

3.6.7 Elasticsearch 27

3.6.8 EC2 27

4 Metodologia de Trabalho **29**

4.1 Estudos de Inteligência Artificial 29

4.2 Planejamento de arquitetura 29

4.3 Implementação Rekognition 29

4.4 Implementação Elasticsearch 30

4.5 Treinamento do modelo 30

4.6 Criação e integração do aplicativo 30

5 Especificação de Requisitos do Sistema **31**

5.1 Visão Empresa 31

5.2 Visão Informação 33

5.3 Visão Computação 33

5.3.1 Arquitetura de Referência 34
5.3.2 Arquitetura da Visão Computação 36
5.3.3 Aplicação da Clean Architecture 37
5.4 Visão Engenharia 39
5.5 Visão Tecnologia 41
5.5.1 Backend 41
5.5.1.1 Serverless 41
5.5.1.2 Node.js 42
5.5.2 Frontend Mobile 42
5.5.3 Frontend Web 42

6 Implementação **43**

6.1 Módulo de classificação 43
6.1.1 Configurações Rekognition 43
6.1.2 Aquisição de dataset de imagens 44
6.1.3 Refinamento da saída do Rekognition 45
6.1.4 Treinamento do modelo de classificação utilizando o Amazon Sage Maker 46
6.1.5 Estruturação de dados no DynamoDB 47
6.2 Módulo de sugestão 48
6.2.1 Criação da API 49
6.2.2 Configuração do Elasticsearch 49
6.2.3 Criação de propagandas no Elasticsearch 50
6.2.4 Query para busca de propagandas 51
6.3 Módulo de demonstração (Aplicação mobile) 53
6.3.1 Criação do projeto Java 53
6.3.2 Integração 56

7 Considerações Finais **58**

7.1 Conclusões do Projeto de Formatura 58

7.2 Contribuições 59

7.3 Trabalhos Futuros 59

Referências **61**

PARTE I

INTRODUÇÃO

1 INTRODUÇÃO

1.1 Objetivo

O objetivo deste trabalho é, por meio do uso de tecnologias de estado da arte de inteligência artificial e arquitetura de serviços, fornecer para o crescente mercado de marketing digital uma solução de alta eficiência para segmentação de público e veiculação de campanhas de marketing invisível integrada a aplicações já existentes.

Para tal, se propõe desenvolver uma ferramenta de marketing mobile direcionado utilizando visão computacional para reconhecimento de locais e interesses em fotos do usuário alimentando uma inteligência artificial treinada para agrupar as imagens em categorias pré definidas e em seguida, usando ferramentas de busca, sugerir propagandas relevantes aos usuários.

O intuito final do trabalho é fornecer uma ferramenta, através de uma biblioteca para desenvolvimento android, que permite que diversas empresas implementem a nossa ferramentas dentro de seus próprios aplicativos, melhorando a satisfação do usuário e do anunciante.

1.2 Motivação

Na sociedade brasileira atual é muito comum ver pessoas entretidas por seus Smartphones. Seja na rua, no ônibus, nos restaurantes, as pessoas sempre estão com, pelo menos, um Smartphone em sua posse.

No início de 2017, o estudo "Google Consumer Barometer", mostrou que, em 2012, 14% da população brasileira tinha um Smartphone. Em 2016, esse número aumentou para 62%, ou seja, em 5 anos, houve um aumento de 450%.

Atualmente o acesso a Internet é feito principalmente através de aparelhos celulares, a utilização de aplicativos para facilitar a vida das pessoas ou simplesmente como forma

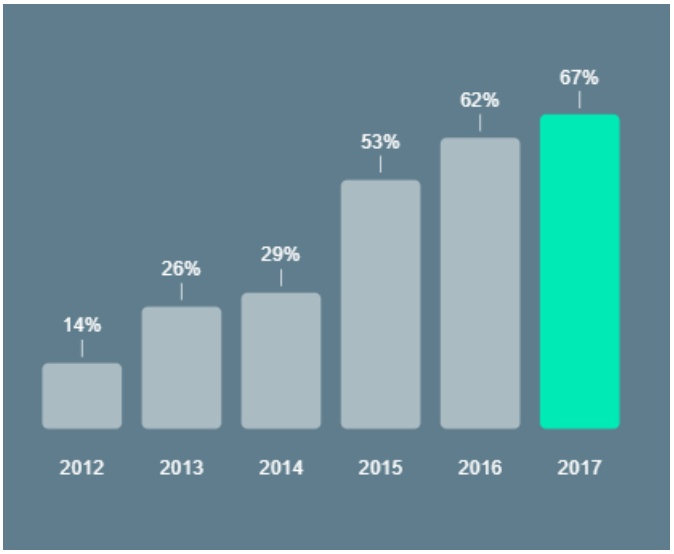

Figura 1: Percentual de pessoas utilizando smartphones no mundo

de entretenimento já se tornou algo cotidiano. Os smartphones já se tornaram parte da vida da população, e assim como outras formas de entretenimento, grandes empresas têm o desejo de explorar essa grande utilização dos smartphones para fazerem anúncios de seus produtos.

É previsto um crescimento no investimento em marketing para smartphones de aproximadamente 22% ao ano, de 2017 até 2023, chegando a marca de um investimento de 102 bilhões de dólares no ano de 2023. Todo esse investimento é normalmente utilizado em plataformas de propaganda genéricas, sem o foco no cliente.

1.3 Justificativa

Com esse cenário de crescimento de marketing para smartphones é fácil justificar a necessidade desse projeto, atualmente existem diversas empresas focadas no desenvolvimento de plataforma para gerar propaganda e conteúdo personalizado. O Google é um dos grandes competidores desse mercado, sendo responsável por boa parte do dinheiro envolvido quando falamos sobre marketing digital.

Mesmo assim, não temos conhecimento de uma plataforma capaz de otimizar essa geração de conteúdo, a maioria das plataformas de marketing utiliza uma quantidade limitada de fatores para determinar o conteúdo que será visualizado pelo cliente final.

A ideia desse projeto é expandir esses limites aplicando conceitos de inteligência artificial como Visão Computacional e Machine Learning, ou seja, com a nossa plataforma será possível fornecer um conteúdo patrocinado de maior relevância para o usuário final,

essa solução é benéfica para os dois lados, dado que grandes empresas que contratam o espaço de propaganda terão a certeza que os usuários atingidos por aquela propaganda tem uma maior chance de adquirir o produto final.

Além disso é uma vantagem para o usuário que quando ver conteúdo patrocinado provavelmente será algo de seu interesse e será relevante para o momento que o usuário visualizar esse conteúdo.

1.4 Estrutura do trabalho

A seguir está descrita a organização do restante da monografia, mostrando o principal objetivo de cada um do capítulos.

O capítulo 2 irá abordar os principais aspectos conceituais abordados no trabalho, tanto do ponto de vista técnico quanto do ponto de vista de produto.

No capítulo 3 serão apresentadas as principais tecnologias utilizadas no projeto, visando apresentar o funcionamento básico da tecnologia e sua aplicação no projeto.

No capítulo 4 será explicada a metodologia usada para implementação do trabalho, contando os principais passos tomados durante o trabalho e como o trabalho foi distribuído.

O capítulo 5 apresentará as especificações técnicas do projeto, utilizando para isso o Reference Model of Open Distributed Processing (RM-ODP).

Já o capítulo 6 irá apresentar toda a implementação do projeto, passando pelos principais problemas encontrados, as formas como o problema foi solucionado e os principais resultados obtidos.

Finalmente o capítulo 7 apresentará as considerações finais do grupo, tanto do ponto vista técnico, pessoal e de produto.

PARTE II

DESENVOLVIMENTO

2 ASPECTOS CONCEITUAIS

2.1 Marketing Digital

Marketing Digital pode ser entendido como a divulgação de produtos e serviços utilizando meios digitais para atingir os consumidores. O objetivo principal a ser atingido é a promoção de marcas por meio das diferentes formas de mídia digital disponíveis.

O conceito tem se expandido na última década, incorporando marketing realizado por meio de telefones celulares por SMS, mídias sociais, marketing de busca etc.

Atualmente, a maioria dos especialistas em marketing acreditam que o meio digital tem características e exigências muito diferentes dos demais canais de marketing. Com a introdução de tecnologias disruptivas como os smartphones e outros devices com conexão a internet, o consenso é que o entendimento do comportamento dos consumidores e seus padrões de uso das plataformas que essas tecnologias proporcionam é um dos fatores mais importantes para o sucesso do marketing digital.

Dentre os novos fatores levantados pelo segmento para otimização de campanhas no contexto digital, esse projeto utiliza dois conceitos em evidência no mercado de marketing: segmentação de mercado e marketing invisível.

2.1.1 Segmentação de mercado

Um conceito bastante discutido, mesmo no contexto de marketing tradicional, a segmentação de público, também chamada de segmentação de mercado, tornou-se fator fundamental no ambiente digital.

Pela forma como o modelo de negócios de veiculação de propagandas se estabeleceu na internet, onde anunciantes pagam pelo alcance de suas divulgações, a otimização de campanhas de marketing invariavelmente passa por um trabalho de segmentação de público que determine qual a fatia de usuários está mais propensa a comprar o seu produto. Para

atingir tal objetivo, aspectos demográficos, geográficos, sociais e econômicos são inferidos a partir de padrões comportamentais dos usuários.

No caso desse projeto, a aplicação desse conceito se dará pela identificação de usuários com categorias de anúncios por meio da classificação das suas fotos. Tal estratégia apresenta a vantagem de se basear em um conjunto de dados, fotos tiradas pelo próprio usuário, que tem alta correlação com padrões de compra e facilita a identificação de interesses dos usuários.

2.1.2 Marketing Invisível

Com a popularização da internet e de tecnologias disruptivas como os smartphones, o mercado de marketing rapidamente buscou ocupar as plataformas utilizadas pelos usuários diariamente. Navegando por redes sociais, portais de notícias e sites de criadores de conteúdo, o público dessas plataformas recebe uma quantidade enorme de campanhas de divulgação de produtos e serviços, muitas vezes irrelevantes para esse usuário.

Dentro desse contexto e buscando evitar uma saturação e perda de relevância do marketing no contexto digital, o conceito de marketing invisível tem ganhado força no mercado. O objetivo é integrar o marketing dentro dessas plataformas de maneira transparente para o usuário, de maneira a tornar a visualização das campanhas o menos disruptiva possível.

Uma estratégia comum e que foi explorada no desenvolvimento desse projeto é a de integrar material promocional nos fluxos de uso das plataformas, por exemplo criando campanhas de marketing que coexistam dentro do feed de notícias do usuário como um post comum, em meio às publicações de outros usuários.

2.2 Inteligência Artificial

O conceito de inteligência artificial(IA) é muito abrangente, diversos livros e artigos divergem sobre uma definição concreta, tanta divergência levou a subdivisão da inteligência artificial em diferentes áreas de pesquisa como: processamento de linguagem natural, representação de conhecimento, raciocínio automatizado, visão computacional, robótica e aprendizado de máquina.

Apesar de tantas divergências podemos explicar a IA como a forma de uma máquina pensar ou atuar de forma humana ou racional, ou seja, se uma máquina é capaz de simular

comportamentos humanos como pensamento e atuar de forma a seguir uma lógica racional ou humana podemos dizer que isso é inteligência artificial.

A cultura popular aborda isso de diversas formas, muitas vezes de forma exagerada e de certa forma ainda distante da realidade, mas o conceito por trás é muito interessante como analogia, a inteligência artificial tenta transformar máquinas em humanos, se um robô consegue possuir tantas características humanas, que se torna difícil diferenciar a máquina de um humano podemos dizer que esse robô é realmente uma máquina ou na verdade devemos considerá-lo como um humano?

2.3 Machine Learning

O conceito de aprendizado de máquina pode ser explicado como a criação de um modelo que dado um conjunto de dados de entrada será capaz de fazer uma previsão de outra variável do sistema, ou de uma saída do sistema. O aprendizado de máquina utiliza algumas terminologias que serão bastante comuns durante a apresentação do trabalho:

- Label: Uma label é considerada como a variável dentro do problema que será descoberta, ou seja, o ponto que será previsto pela máquina.
- Feature: São todas as variáveis de entrada do problema, ou seja, todos os dados que devem ser usados para prever o resultado de uma Label.
- Modelo: Um modelo define a relação entre as label e features, e pode ser dividido em duas etapas, o treinamento e a dedução. No treinamento, o modelo conhece o valor tanto das Labels quanto das Features, e com isso é mostrado para o modelo a relação entre as variáveis. Depois vem a etapa de dedução, onde as Labels não são conhecidas, mas o modelo já é capaz de inferir o resultado a partir das Features apresentadas para ele.

Durante o projeto será criado um modelo capaz de sugerir propagandas para um usuário, ou seja, dado a entrada de um perfil de usuário, com várias Features como idade, onde mora, locais que mais frequenta, rotina, etc, será gerada uma Label de qual propaganda mais se aproxima do perfil do usuário.

2.4 Algoritmos de Classificação

Na raiz do modelo de sugestão está um algoritmo de classificação. No caso do sistema apresentado aqui, será necessário determinar um algoritmo que, a partir das características levantadas sobre as fotos analisadas, as classifique em categorias pré-definidas de acordo com os grupos de propagandas que serão exibidas.

Dentro do contexto de aprendizado de máquina, a operação de classificação é uma técnica de aprendizado supervisionado na qual o programa aprende a partir de um conjunto de dados e utiliza esse aprendizado para classificar um novo input dado a ele. Algoritmos utilizando esse princípio são comumente encontrados na literatura para solucionar problemas do tipo: reconhecimento de linguagem natural, reconhecimento de escrita manual, identificação biométrica etc.

Tendo em vista a amplitude de algoritmos que podem ser usados para a classificação, a definição do algoritmo a ser utilizado passou pelo levantamento de vantagens e desvantagens na utilização de algumas das estratégias. Por fim, tendo em vista o tamanho do nosso conjunto de dados de treinamento e capacidade computacional, optamos por utilizar para esse projeto o algoritmo K-Means, descrito abaixo.

2.4.1 K-Means

O algoritmo k-means trabalha com um método iterativo simples para dividir um conjunto de dados de entrada em um número k de clusters, definido pelo usuário. A partir dessa organização, predições podem ser feitas ligando novos pontos de entrada a proximidade a cada um dos clusters.

O conjunto inicial de dados é organizado em um conjunto de vetores de dimensão d, D = xi — i = 1,..., N onde xi ϵ d é o dado de ordem i. Para inicializar o algoritmo, devemos escolher k pontos em d, que serão os centros iniciais dos clusters. Existem algumas técnicas para selecionar esses pontos, incluindo a seleção ao acaso, selecionar a partir da resolução do algoritmo para um conjunto filho do conjunto inicial ou partir da média do conjunto e realizar k desvios em diferentes direções.

A partir de então, o algoritmo itera entre dois passos até a convergência:

- Atribuição dos dados

  Cada ponto do conjunto de entrada é atribuído ao centro de cluster mais próximo. Dessa forma, separamos os dados em subconjuntos.

- Realocação dos centros

  Nessa etapa o centro de cada um dos k clusters é realocado a partir do cálculo da média dos pontos que compõem o subconjunto desse cluster.

O algoritmo converge quando os pontos não mudam mais de clusters. Abaixo temos uma representação visual do funcionamento do algoritmo com três clusters e vetores de duas dimensões:

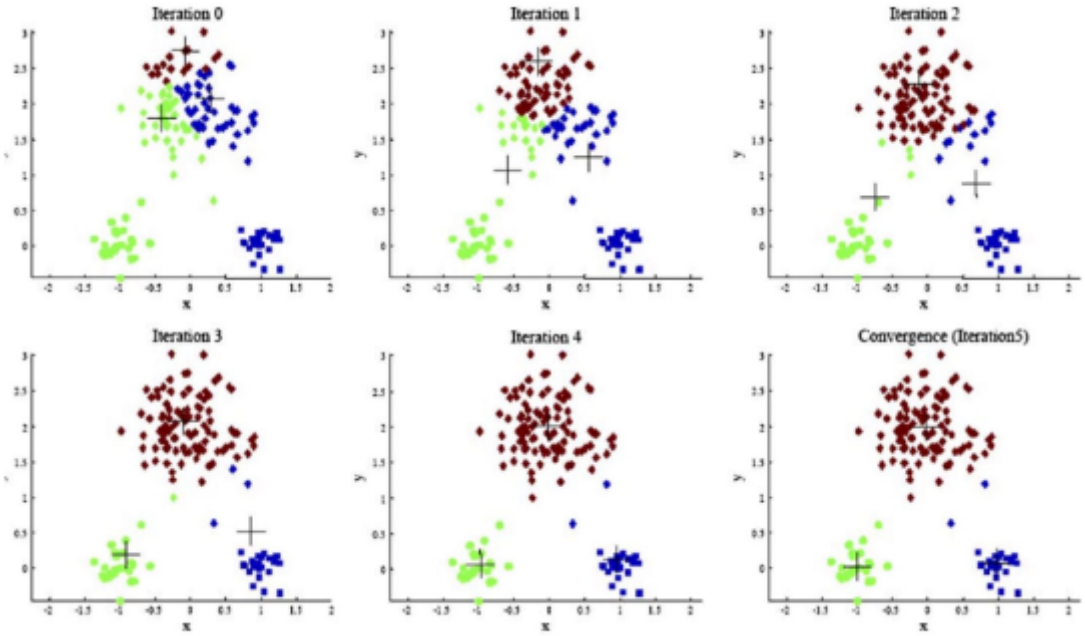

Figura 2: Gráfico de iterações do algoritmo K-Means

2.5 Deep Learning

O Deep Learning, ou aprendizagem profunda, é um ramo de algoritmos da inteligência de máquina, baseado em um conjunto de algoritmos que tentam modelar abstrações de alto nível de dados usando um grafo profundo com várias camadas de processamento. Softwares baseados no conceito de deep-learning tentam reproduzir o comportamento de camadas de neurônios no neocórtex, a parte do cérebro onde o pensamento acontece. Com esse modelo, algoritmos desse tipo aprendem a reconhecer padrões em sons, imagens e outros tipos de dados. A ideia por trás desse conceito de rede neural já existe a algumas décadas, no entanto avanços em formulações matemáticas e poder computacional possibilitaram recentemente que redes neurais mais complexas fossem construídas.

2.6 Visão Computacional

A visão computacional é uma das vertentes do deep learning que estuda como reconstruir, interpretar e compreender uma cena 3D a partir de imagens 2D em termos das propriedades das estruturas presentes na cena. O principal objetivo da visão computacional é modelar, replicar e mais importante exceder a visão humana usando software e hardware em diferentes níveis.

A visão computacional é a construção de significativas descrições de objetos físicos de suas imagens. A saída de visão computacional é uma descrição, uma interpretação ou alguma medida quantitativas das estruturas na cena 3D. Processamento de imagem e reconhecimento de padrões estão entre muitas técnicas de visão computacional empregada para atingir seus objetivos.

Dentro do contexto do projeto, a visão computacional será usada para categorizar as imagens tiradas pelo usuário da aplicação, ou seja, iremos retirar uma descrição do local da foto.

2.7 Cloud Computing

A computação na nuvem pode ser explicada como a entrega sob demanda de poder computacional, armazenamento de dados e diversos serviços via Internet com um custo conforme o uso. As principais vantagens da computação na nuvem são a facilidade de integração dos serviços, a escalabilidade e o custo reduzido devido ao pagamento sobre o uso, assim evitando gastos de poder computacional ocioso.

2.7.1 IaaS

O conceito de IaaS (Infrastructure as a Service) está ligado ao provisionamento de estrutura computacional na forma de um serviço, ou seja, todo o hardware necessário para executar um sistema é fornecido como um sistema.

Quando falamos de computação em nuvem esse é o conceito mais comum aplicado a empresas devido a fácil migração de servidores físicos para servidores na nuvem, o caso de uso mais comum de utilização de servidores na nuvem são o de servidores dedicados para operação de sistemas e sistemas de armazenamento de dados de larga escala.

2.7.2 FaaS

Já o conceito de FaaS (Function as a Service) vem ganhando força nos últimos anos, com o lançamento de serviços especializados nesse conceito. No FaaS o serviço é contrato para a execução de um função e o pagamento é feito apenas sobre o consumo da função individual.

O FaaS trabalha juntamente com o conceito de aplicações stateless, ou seja, sua função deve ser executada sem nenhum estado de servidor, e apenas com dados de entrada. As principais vantagens do FaaS estão nos custos, foi você paga apenas pela utilização e é fácil balancear o uso de diferentes funções, na computação mais tradicional isso é um problema, pois não é fácil dividir um sistema de forma a não sobrecarregar uma parte dele que está sendo mais requisitada.

2.8 Serverless

Apoiado no conceito de FaaS surge o conceito de Serverless, onde o intuito é a operação de um sistema sem servidor, todas as funções do sistema ficam disponíveis através de funções stateless e são invocadas conforme a necessidade, esse tipo de arquitetura requer a utilização de vários serviços em conjunto de forma a criar um rede de funções que representa o sistema.

Os principais provedores de cloud da atualidade oferecem os principais serviços necessários para a implantação de uma arquitetura serverless, e com auxílio de frameworks específicos é fácil criar e manter esses serviços sem grandes necessidades de manutenção acompanhada.

As principais vantagens do serverless estão no custo, pois o valor gasto é apenas o valor utilizado, não existe computação ociosa no serverless, a escalabilidade, garantida pelos diversos serviços inerentes da arquitetura, e a facilidade de manutenção, os serviços garantem o funcionamento do sistema, não é necessário contratação de pessoas para cuidar do servidor e verificar quando é necessário contratar mais poder computacional.

3 TECNOLOGIAS UTILIZADAS

3.1 Elasticsearch

O Elasticsearch é um mecanismo de busca altamente escalável e open-source. Com o Elasticsearch é possível armazenar uma grande quantidade de dados, na ordem de terabytes, e fazer busca em cima desses dados de forma rápida e escalável. O Elasticsearch utiliza uma estrutura com uma API Rest para fazer a comunicação com o Apache Lucene, que é responsável pelo motor de busca.

O Apache Lucene é responsável por armazenar os dados de forma indexada e realizar as buscas desses dados. Ele funciona de forma tão eficiente pois todos os seus dados armazenados são indexados, tornando a busca por termos muito eficiente, ou seja, com ele é possível buscar um conjunto de propagandas que esteja vinculado a determinadas categorias, ou além disso, é possível filtrar essas propagandas de acordo com distancia ou com algum conteúdo específico do anuncio.

O Lucene funciona de forma a separar todos os campos de um determinado documento e indexar essas palavras de forma individual, com isso todos os termos do documento são indexados de forma separada, tornando a busca tão rápida e precisa.

As buscas feitas com esse mecanismo são classificadas através de um score que é calculado baseado na relevância dos resultados. Neste trabalho, o elasticsearch é usado para fazer a busca de propagandas, o cálculo dos scores é feito em cima das categorias pré-definidas e da distância do usuário para o anúncio.

3.2 API

API (Application Programming Interface), como o nome diz, é simplesmente uma interface de comunicação entre várias aplicações. Podemos dizer que existe uma aplicação pai, que é responsável por tratar de alguma lógica específica, e aplicações filhas, que irão

consumir dessa aplicação pai.

Para facilitar o trabalho de desenvolvedores, foi criado o conceito de API, onde a aplicação pai define uma interface de comunicação, ou seja, um protocolo que deve ser seguido por todas as aplicações filhas que desejam usar as funcionalidades da aplicação pai.

Para o desenvolvimento do trabalho foi utilizada um API do tipo REST. REST (REpresentational State Transfer) é uma abstração da arquitetura World Wide Web, é composto por um estilo arquitetural que consiste em um conjunto de restrições a serem aplicados em um API, ou seja, REST nada mais é do que um conjunto de regras que uma API deve seguir, um padrão para facilitar a comunicação entre desenvolvedores.

As principais regras impostas pelo REST são:

- Uso de protocolo HTTP (verbos, accept headers, códigos de estado HTTP, Content-Type) de forma explícita e representativa para se comunicar, usando XML ou JSON como forma de transferência de dados.
- Não possui estado entre essas comunicações.
- Deve facilitar o cache de conteúdo no cliente.
- Deve ter clara definição do que faz parte do cliente e do servidor.

Dada uma API REST é fácil entender o funcionamento do sistema e como utilizá-lo pois a API irá seguir regras pré-estabelecidas e comuns a uma grande quantidade de sistemas.

3.3 HTTP

O HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação, na camada de aplicação do modelo OSI utilizado para transferência de dados na World Wide Web. Para que o protocolo HTTP consiga transferir seus dados pela Web, é necessário que os protocolos TCP e IP (Internet Protocol, Protocolo de Internet) tornem possível a conexão entre clientes e servidores através de sockets TCP/IP.

Para garantir a segurança dentro da rede, foi utilizada uma extensão do protocolo HTTP, o HTTPS. É o protocolo de comunicação encriptado usando Transport Layer Security (TLS) ou Secure Sockets Layer (SSL). A principal motivação do HTTPS é realizar

a autenticação de requisições na rede de forma segura, protegendo os dados de quem faz a requisição e garantindo a integridade de quem processa a requisição.

O HTTPS protege a rede de ataques intermediários, ou seja, após uma requisição ser feita se alguém interceptar essa requisição não é possível alterá-la ou ler os dados da mesma, tornando a comunicação entre cliente e servidor segura.

3.4 Javascript

Node.js é um interpretador de código Javascript, open-source, projetado para rodar código escrito em javascript fora do navegador, ou seja, com ele é possível criar um servidor que interpreta código javascript, sendo possível criar uma abordagem server-side para uma linguagem originalmente criada como client-side.

Javascript é um linguagem de programação interpretada, foi inicialmente introduzida como uma forma de executar scripts dentro de navegadores web, assim era possível que sites estáticos executassem trechos de código com lógicas para realizar ações mais complexas como animações específicas, requisições para servidores e etc.

Uma das maiores reclamações dos desenvolvedores Web em relação ao Javascript é a falta de organização e tipagem dentro da linguagem. Isso levou a necessidade de utilizar outras bibliotecas e ferramentas para auxiliar os desenvolvedores a controlar melhor seus códigos. Uma dessas bibliotecas é o Typescript, criado pela Microsoft, ela permite a utilização de diversas ferramentas para o desenvolvimento que não são nativas ao javascript, como a utilização de interfaces, classes, tipos de variável bem definidos.

Entre várias características do framework podemos destacar como principais: operação em uma única thread, suporte para operações assíncronas, grande otimização para melhorar escalabilidade e vazão, suporte para os principais sistemas operacionais e um gerenciador de dependências integrado com a plataforma.

3.5 Android

O Android SDK é a ferramenta usada para criação de aplicações mobile para Android, ele é usado para compilar código escritos em Java para serem rodados na ART e Dalvik(máquinas virtuais personalizadas para rodar dentro de dispositivos Android).

Java é uma linguagem de programação interpretada orientada a objetos. Diferente

das linguagens de programação convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é interpretado por uma máquina virtual (Java Virtual Machine, mais conhecida pela sua abreviação JVM).

Java é uma das linguagens utilizadas para a programação de aplicações mobile para Android Embora as ferramentas do SDK possam ser usadas por linha de comando, no trabalho foi usado o software para desenvolvimento Android Studio, que compila e executa os projetos Android através de emuladores ou devices reais.

3.6 AWS

Para a realização do trabalho será utilizado os serviços de cloud da AWS (Amazon Web Services), a AWS é uma das pioneiras na construção de arquiteturas serverless, além disso a AWS oferece serviços de treinamento de Machine Learning e serviços de análise de imagem, sendo uma solução completa para os problemas de infraestrutura do trabalho.

A seguir serão apresentados os serviços utilizados no trabalho, quais suas principais funcionalidades, suas limitações e seus usos dentro do trabalho

3.6.1 Rekognition

Serviço de análise imagens e vídeo da AWS, o Rekognition possui três principais funcionalidades: detecção de objetos, reconhecimento facial, reconhecimento de texto em imagens.

Para o trabalho será utilizada apenas a funcionalidade de detecção de objetos, com esse recurso é possível identificar diversas tags vinculadas a uma imagem, ou seja, são identificados todos os elementos relevantes em uma imagem com uma porcentagem de confiabilidade vinculada.

A principal limitação do Rekognition está na precisão do seu algoritmo, como não é algo controlado pelo grupo, não conseguimos aprimorar o algoritmo de Deep Learning do serviço para alcançar melhores resultados ou resultados mais específicos para o problema que queremos resolver. Além disso o serviço possui uma limitação de conexões simultâneas, onde é apenas permitido a análise de 50 imagens por segundo.

3.6.2 Sagemaker

O Sagemaker é a plataforma gerenciada da AWS para criação, treinamento e implementação de modelos de Machine Learning. Podemos dividir o Sagemaker em três principais recursos:

- Notebook: Local para criação dos modelos, permite você criar algoritmos, organizar dados, ativar o treinamento do modelo e verificar resultados do treinamento.
- Training: Recebe um input de um notebook e aloca uma máquina para o treinamento do modelo, a máquina é alocada apenas durante o treinamento, evitando o uso de recursos ociosos.
- Endpoints: Após o treinamento, é gerado um endpoint para realização de previsões, ou seja, uma API é criada e disponibilizada na forma de um endpoint HTTP, esse endpoint recebe um input apropriado e faz a previsão de resultado a partir do modelo pré-treinado.

3.6.3 Lambda

A Lambda é o serviço da AWS para a execução de código sem a provisão de um servidor dedicado para essa tarefa, como comentado o AWS Lambda é um dos pilares da arquitetura serverless, além de ter sido um dos primeiros serviços criados para essa finalidade entre todos os provedores de Cloud do mundo.

A lambda funciona com uma configuração simples, onde é necessário fazer o upload de um código em algumas linguagem suportada pelo serviço, o Lambda é compatível com Node.js (JavaScript), Python, Java (compatível com Java 8), C (.NET Core) e Go. Além disso é necessário especificar o gatilho da sua função, ou seja, o evento que irá invocar a execução do seu código.

As principais limitações do Lambda estão vinculadas as configurações do serviço, a função executa com um memória RAM pré-definida, ou seja, é função do cliente estipular quanta memória será usada e isso irá influenciar diretamente no custo e no tempo de execução, mas esse valores são pré-definidos e podem não ser suficientes para determinadas aplicações.

Além disso a lambda possui uma limitação de timeout, ou seja, a execução de funções Lambda podem durar no máximo 15 minutos, o que pode ser um limitante dependendo da forma como a função for escrita.

3.6.4 DynamoDB

O DynamoDB é um banco de dados não relacional que fornece performance confiável em qualquer escala. Assim como os principais serviços associados a uma arquitetura serverless não o DynamoDB não está localizado em um servidor dedicado, a distribuição de dados é operada pela própria AWS, permitindo que desenvolvedores definam um limite de capacidade para utilização do banco e a AWS garanta a capacidade estipulada.

Uma das grandes vantagens do DynamoDB é a sua confiabilidade, e sua escalabilidade, permitindo o desenvolvimento de aplicações de grande porte sem a preocupação com alocação de recursos para dados.

3.6.5 API Gateway

O API Gateway é o serviço de gerenciamento de APIs da Amazon, esse serviço provê a geração de endpoints customizados e configuráveis que são usados como acionadores de eventos da Lambda, ou seja, o endpoint possui as suas configuração HTTP (método, url, porta, domínio, etc) e umas vez que esse endpoint é chamado uma função Lambda é invocada e todos os parâmetros passados pelo endpoints são direcionados para a Lambda.

Esse funcionamento permite a criação de APIs completas para o gerenciamento de aplicações web e mobile mantendo a segurança provisionada pelo protocolo HTTPS e todos os padrões de comunicação estipulados pelo protocolo HTTP.

3.6.6 S3

O S3(Simple Storage Service) é o serviço de armazenamento de dados em larga escala da AWS, é um dos serviços de storage mais confiáveis do mercado e um dos 'carros chefe' da empresa, possui uma resiliência de 99,999999999% a um custo muito acessível.

O S3 é dividido em buckets, quem funcionam como grandes pastas, onde cada bucket possui uma configuração e acesso próprio, o serviço é usado como storage de todas as fotos da nossa aplicação, tanto imagens de usuários quanto as imagens usadas para o treinamento do nosso modelo de Machine Learning, o S3 também é usado para o armazenamento dos códigos das nossas funções Lambda e para o armazenamento das configurações de treinamento do Sagemaker.

Outra utilidade do S3 é a de ser usado como fonte de eventos para disparar funções Lambda, esse serviço pode ser configurado para quando houver um upload em determi-

nado bucket de um arquivo de determinado tipo, ou com determinado nome, uma função Lambda é acionada e o endereço onde tal arquivo foi salvo é enviado como parâmetro para o Lambda.

3.6.7 Elasticsearch

Como comentado anteriormente o Elasticsearch é um algoritmo open source para a realização de buscas, mas isso permite apenas a utilização do algoritmo, nenhuma infraestrutura é fornecida, é necessário que o algoritmo seja instalado e configurado em um servidor, isso traz diversas consequências, como o gerenciamento de fluxo de dados dentro da máquina, algo que pode facilmente se tornar muito complicado quando uma aplicação escala rapidamente.

A AWS fornece um serviço gerenciamento de Elasticsearch, chamado AWS Elasticsearch, ele permite a configuração de um servidor com o algoritmo já instalado e com um controle de acesso e de rede através de políticas de acesso. Esse serviço também permite a mudança de máquina alocada, permitindo a migração para servidores menores ou maiores de forma rápida e segura, sem a perda de dados ou de disponibilidade.

A principal desvantagem desse serviço é que ele não opera como os outros serviços utilizados no projeto, onde o custo de operação é apenas o utilizado, nesse caso você contrata uma máquina de tamanho variado e paga pelos recursos em tempo integral, sendo eles ociosos ou não.

3.6.8 EC2

O EC2 (Elastic Compute Cloud) é o serviço de disponibilização de capacidade computacional segura e redimensionável na nuvem, é o principal produto da AWS, e podemos resumi-lo como: servidores na nuvem, com esse serviço você possui acesso quase total a uma máquina de tamanho configurável e pode fazer qualquer tipo de operação dentro dele, o caso de uso mais comum para esse serviço é fazer a hospedagem de servidores e de bancos de dados, sendo uma solução completa para a criação do backend de um projeto de qualquer tamanho.

Empresas do mundo todo utilizam o EC2 como principal fonte de hospedagem de seus servidores, dada a fácil configuração de seus servidores e pelo fato de ser facilmente redimensionável. Além disso com a ajuda de outros serviços é possível fazer a distribuição de diversas máquinas para a operação em conjunto, permitindo que a quantidade de

servidores disponibilizado para uma aplicação cresça infinitamente, onde o único limitante é o tamanho dos data centers provisionados pela AWS.

Esse serviço foi usado apenas para testes dos algoritmos de Elasticsearch em um ambiente real, fora isso o serviço foi usado apenas de forma indireta, tanto o AWS Elasticsearch quanto o AWS Sagemaker utilizam máquinas do EC2 específicas e com configurações especiais para otimização de seus processos.

4 METODOLOGIA DE TRABALHO

O trabalho foi divido em X categorias, onde a realização das mesmas aconteceu de forma paralela devido a independÊncia de completude de uma categoria para a realização de outras.

4.1 Estudos de Inteligência Artificial

Assim que o tema do trabalho foi definido identificamos a necessidade de dedicar tempo de estudo para a principal tecnologia envolvida no trabalho, para isso o grupo realizou cursos de machine learning, associado a leitura de diversos artigos relacionados a machine learning, deep learning e visão computacional.

Também utilizamos as aulas da matéria PCS 3838 e o livro Artificial Intelligence: A Modern Approach como base teórica de inteligência artificial para o desenvolvimento do trabalho.

4.2 Planejamento de arquitetura

Com os conhecimentos necessários consolidados fizemos um estudo de como seria organizada a arquitetura no projeto, como seriam abordados diversos conceitos e uma grande quantidade de tecnologias seria usada o grupo optou por utilizar os serviços da AWS como base do trabalho e com isso modelamos como cada serviço seria usada de forma a atender as necessidades do grupo.

4.3 Implementação Rekognition

O primeiro serviço a ser integrado foi o serviço do AWS Rekognition, ele seria o ponto de entrada da nossa aplicação, pois com ele teríamos uma classificação inicial de nossas

imagens, por isso optamos por começar a implementação por ele e entender suas principais funcionalidade e limitações para adequar o resto do trabalho aos outputs do Rekognition.

4.4 Implementação Elasticsearch

A necessidade da implementação do Elasticsearch surgiu durante o planejamento da arquitetura, o que necessitou em mais um período de estudos sobre o algoritmo, e através de cursos online e leitura de documentações sobre o algoritmo. Em seguida foram realizadas provas de conceito do funcionamento e da instalação de um algoritmo de elasticsearch dentro computadores pessoais e dentro de máquinas da AWS.

4.5 Treinamento do modelo

O treinamento e aperfeiçoamento do mesmo durou quase toda a implementação do trabalho, foi necessário a aquisição de datasets de imagens que atendessem as categorias de propaganda que tínhamos interesse. Além disso tivemos etapas de estudo e testes do serviço de machine learning da AWS e de modificações do algoritmo para atenderem as necessidades do projeto.

4.6 Criação e integração do aplicativo

A última etapa do projeto consistiu na criação da aplicação mobile que seria usada para demonstração e validação do projeto. Além disso nessa etapa foi feita a integração de todos os serviços e foram feitos os testes de validação do funcionamento do projeto.

5 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

Para a especificação do sistema será utilizado o modelo proposto pelo Reference Model of Open Distributed Processing (RM-ODP). Esse modelo divide a arquitetura do projeto em cinco visões, onde cada visão foca em descrever o sistema de um ponto de vista diferente com o objetivo de criar uma arquitetura mínima do projeto, sendo capaz de descrever todo o projeto.

5.1 Visão Empresa

Como apresentado anteriormente o mercado mobile vem crescendo de forma assombrosa, a mudança de tecnologia predominante, do computador pessoal para os smartphones, já é realidade. Pessoas deixam de comprar um computador de mesa ou um notebook para comprar um smartphone de última geração, e o mercado de marketing já está reagindo a essa mudança.

O mercado de marketing e propaganda é um dos mercados que mais movimenta a economia mundial, bilhões de dólares são investidos anualmente com propagandas publicitárias com o intuito de impulsionar as vendas de determinado produto. Atualmente empresas não competem para fazer o melhor produto, mas sim a melhor propaganda, pois é isso que provavelmente irá fazer o produto vender mais.

Com essa necessidade estão associados dois principais problemas: Como fazer propagandas que atraem o cliente para comprar o produto e como fazer que os clientes vejam essa propaganda. O principal objetivo do projeto é resolver esse segundo problema, ou seja, como entregar as propagandas certas para os clientes certos.

O SmartAd entra como um facilitador entre 3 personagens, sendo eles os clientes finais, os anunciantes e os desenvolvedores de aplicações de redes sociais. O nosso projeto conecta esses três personagens através de uma plataforma específica para cada um, nesse

modelo o anunciante não precisa se preocupar em quais lugares a sua propaganda será exibida, ele apenas define quantas visualizações direcionadas ele deseja e paga de acordo com esse número.

Para os clientes nada muda, eles não precisam baixar uma nova aplicação para ver essas propagandas, elas serão inseridas dentro dos aplicativos de rede social do usuário de forma direcionada. Para os desenvolvedores de aplicações de rede social é necessária a integração com a plataforma de mobile desenvolvida no projeto, e a forma de remuneração para essas aplicações é proporcional a quantas propagandas os seus usuários estão visualizando, ou seja, quanto mais usuários na aplicação, mais remuneração a rede social irá ganhar.

A figura 3 compreende como a Visão Empresa compreende o negócio do projeto, explicitando como os personagens interagem com o sistema.

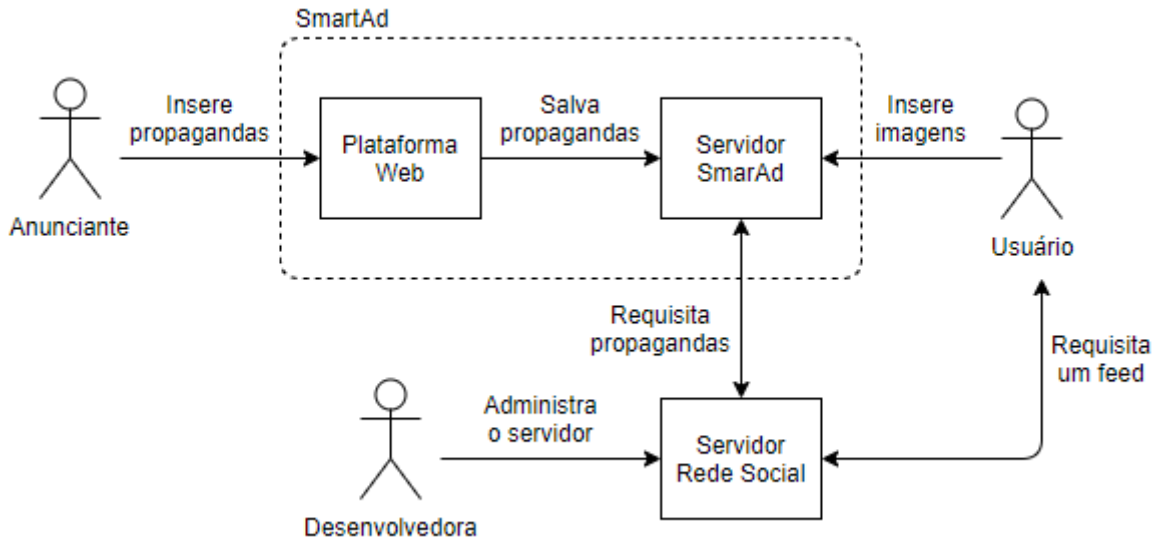

Figura 3: Processo de negócio do projeto

No diagrama podemos ver a interação entre os 3 personagens citados anteriormente e como cada um deles participa dentro do sistema de forma diferente. O Anunciante irá primariamente utilizar a plataforma Web, todo o fluxo de caixa do negócio depende desse fluxo, uma vez que, dentro dessa plataforma Web, o anunciante criará os anúncios e fará o pagamento por essa divulgação.

A partir desse fluxo a nossa empresa consegue operar, nesse ponto entra o outro participante do sistema, os Parceiros, tão essenciais quanto os Anunciantes, uma vez que o produto desses parceiros será o local de divulgação das plataformas, portanto a

existência de parceiros com grande volume de usuários garante que os Anunciantes terão suas propagandas exibidas aos usuários finais.

Esse personagem age através da integração da biblioteca mobile desenvolvida no projeto, seria necessário algum tipo de plataforma onde os parceiros poderiam se cadastrar, e com esse cadastro, a nossa empresa faria o liberamento do acesso a biblioteca e daria o suporte para a instalação da mesma, como benefício, esses parceiros ganham uma parte do valor arrecadado com o pagamentos das propagandas.

Finalmente temos o nosso personagem final, o Usuário, o nosso produto tem pouco impacto visível para esse personagem, uma vez que ele não vê o nosso produto diretamente, mas sim consome ele através das propagandas exibidas pelos aplicativos nos nossos Parceiros, além de ser esse personagem que alimenta o nosso banco de dados com informações sobre seu próprio perfil.

5.2 Visão Informação

A Visão Informação deve representar as informações que o sistema necessita para resolver o problema, portanto essa visão deve exemplificar toda o fluxo de informação para que uma propaganda seja sugerida.

Para isso o mais importante é mostrar o fluxo para a geração do perfil de um usuário, ou seja, como a partir de uma foto tirada pelo usuário, será criado um conjunto de informações que definem o perfil do usuário, e que posteriormente serão usadas na busca de propagandas.

Para mostrar esse fluxo foi utilizado um Diagrama Entidade-Relacionamento (DER), indicado na figura 4, que mostra o fluxo de informação a partir do momento que uma foto é tirada, a partir disso são extraídos dois tipos de informação, os metadados da imagem, que serão utilizados como forma de identificação da imagem, e as categorias provenientes do serviço de análise de imagem, essas categorias servem então de input para o modelo de aprendizado de máquina que gera um cluster associado a imagem, finalmente esse conjunto de informações, metadados e cluster, são salvos em um banco de dados.

5.3 Visão Computação

A visão computacional descreve como o sistema interage entre cada um dos módulos, portanto, com esse documento podemos entender o relacionamento entre as camadas do

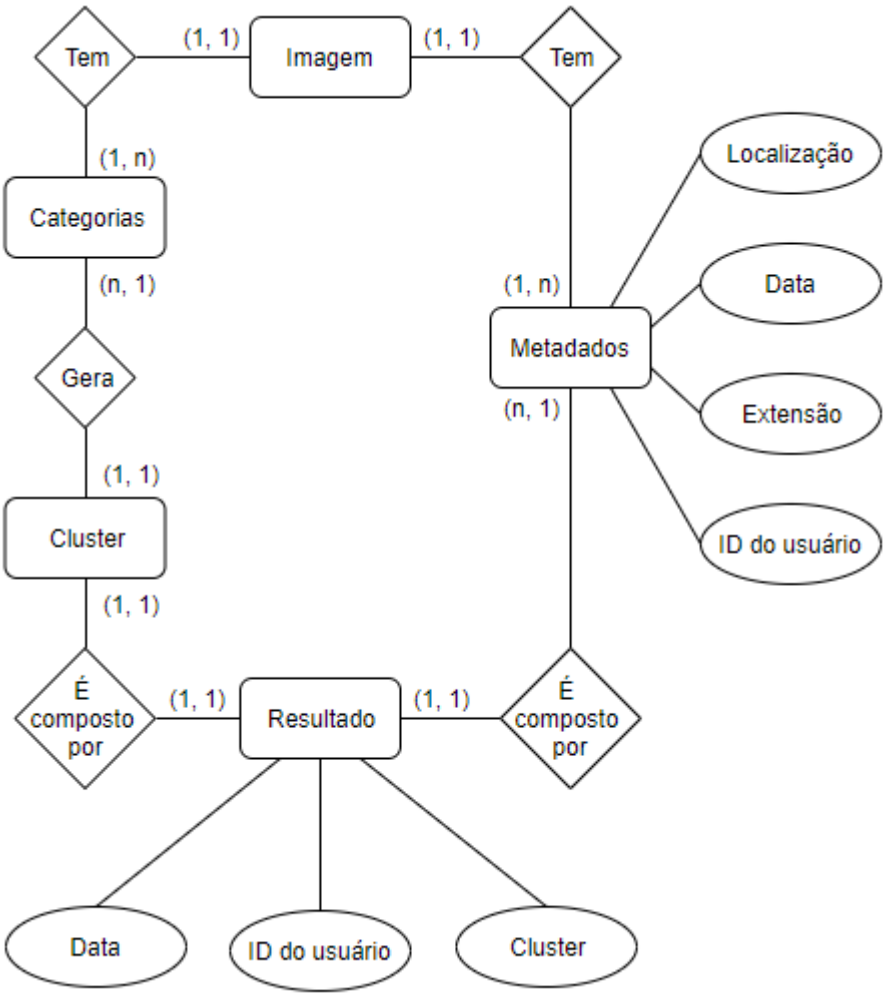

Figura 4: Diagrama de Entidade e Relacionamento

projeto, qual a responsabilidade de cada camada, e como funciona o interfaceamento entre essas partes do sistema.

5.3.1 Arquitetura de Referência

A arquitetura do projeto foi montada seguindo como referência os padrões definidos pelo NIST (National Institute of Standards and Technology). Para o trabalho foram considerados dois artigos específicos, um que diz respeito a computação Cloud e outro que referencia a segurança em aplicações Cloud.

O NIST SP 800-145 define o que é a computação na nuvem, e quais padrões devem ser seguidos para atender as especificações da arquitetura, o artigo define as principais características que o sistema deve possuir para ser considerado um sistema Cloud. as características essenciais do artigo podem ser vistas no diagrama da figura 5.

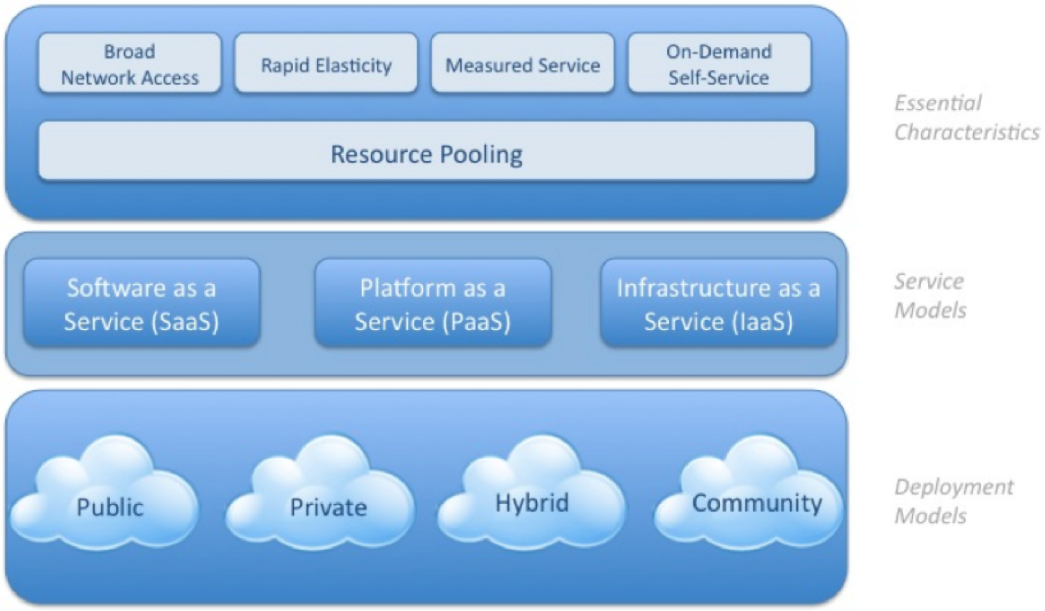

Figura 5: Diagrama da NIST SP 800-145

O outro artigo que foi usado como base foi o NIST SP 800-53 que explicita padrões de segurança a serem seguidos por sistemas de forma que os dados de usuários estejam seguros e não possa haver vazamento de dados. Esse artigo estipula regras arquiteturais que o sistema deve possuir para garantir a segurança dos usuários dentro do sistema.

O sistema desenvolvido para o projeto segue as especificações definidas pelos dois artigo, para o NIST SP 800-53, a provedora de Cloud AWS, utilizada para o projeto, segue as referências de segurança implementando protocolos de VPN entre suas camadas para garantir o acesso apenas de usuários autorizados, além do uso de policies de segurança para comunicação entre serviços da provedora.

As regras NIST SP 800-145 são garantidas também pela AWS em conjunto com a arquitetura definida pelo projeto, a utilização de uma arquitetura de três camadas garante a segurança dos dados de forma a apenas usuários autenticados serem capazes de fazer modificações sensíveis ao sistema.

Abordando as características do NIST 800-145 com a estrutura do projeto podemos associar várias características com o projeto, do ponto de vista de Deployment Models, o projeto trabalhou com uma abordagem Híbrida, isso se deu devido a necessidade de alguns micro serviços serem privados, como o treinamento do modelo de machine learning, e outros micro serviços, como a api criada para sugestão de propagandas e a api para criação de propagandas, serem de domínio público, e podem ser acessados independemente do host de origem.

Do lado de Service Models, apesar de boa parte do projeto operar com conceitos FaaS, que não foram considerados na NIST 800-145, vários serviços utilizados pelo projeto se apoiam no conceito de IaaS, um dos mais conhecidos serviços de IaaS do mundo é a EC2, serviço de computação dedicada da AWS, se entrarmos mais no detalhe de alguns serviços como Sagemaker e Elasticsearch, podemos ver que na verdade todo o processamento é operado em uma máquina da própria EC2, customizada para resolver um problema específico.

Sobre as Essential Characteristics, podemos levantar alguns pontos interessantes, o nosso projeto opera com uma disponibilidade de mais de 99%, garantido pelo AWS Lambda, esse serviço possui essa propriedade devido a característica distribuída do próprio serviço, ou seja, o projeto não está vinculado a uma máquina, mas sim a um hub de máquinas da AWS dedicado a isso.

Essa característica, associado a facilidade de escalar dentro da Cloud, garante o crescimento do serviço oferecido pelo projeto de forma segura e natural, tendo como respaldo esses números, conseguimos garantir uma eficiência de quase 100% a qualquer parceiro que quiser integrar o nosso serviço em sua plataforma, ou dar uma certeza a qualquer anunciante que o serviço de anúncio contratado será entregue ao usuário final.

5.3.2 Arquitetura da Visão Computação

Como mencionado, o projeto utiliza uma arquitetura de três camadas para realizar a comunicação dos serviços e aplicações do projeto, isso garante que a responsabilidade de cada camada esteja bem definida e os acessos de cada camada sejam restritos de forma a impedir de usuários mal intencionados consigam danificar o sistema de alguma forma.

A utilização dessa arquitetura também auxilia o projeto no sentido de organizar a estrutura de código e garantir que cada camada tenha uma responsabilidade de software bem definida, ou seja, quando um problema surgir, ou uma nova implementação for feita, a organização sistema garante que as lógicas do projeto possam ser reaproveitados ou que a modificação em uma camada da aplicação não irá afetar diretamente o funcionamento do sistema. As três camadas, que compõe o sistema estão descritas a seguir:

- Camada de Aplicação: Camada de apresentação do sistema, responsável por mostrar telas, coletar dados do usuário, validar esse dados e se comunicar com a camada de negócio.
- Camada de Negócios: Responsável por contar todas as regras de negócio da aplicação,

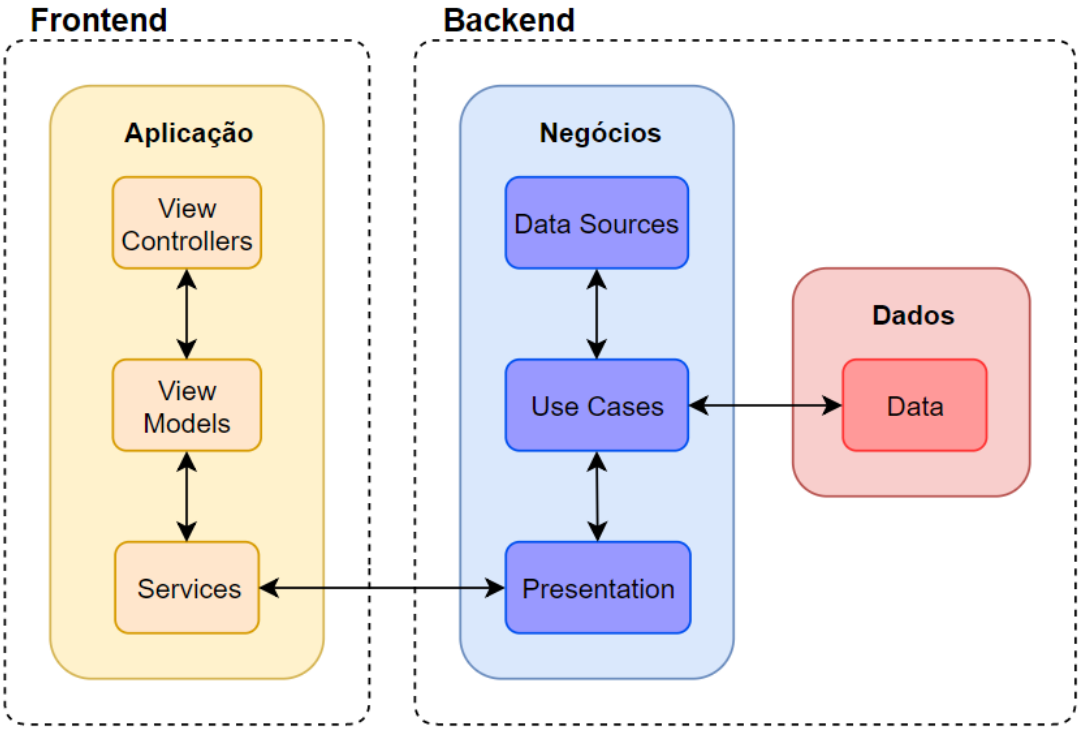

Figura 6: Diagrama da arquitetura em camadas do sistema

essa camada trata de dados sensíveis e deve ser protegida, sendo capaz de acessá-la apenas por meio de interfaces bem definidas.

- Camada de Dados: Trata a persistência de dados e consulta desses dados, camada mais sensível da aplicação, uma vez que alterar essa camada impacta a veracidade dos dados da aplicação.

5.3.3 Aplicação da Clean Architecture

Para a arquitetura de software do backend foi usado o conceito de clean architecture. Essa arquitetura prega a independência de banco de dados, independência de frameworks e independência de agentes externos, ou seja, cada camada da arquitetura é independente e pode ser substituída sem alterar o funciona do sistema como um todo.

Outro conceito da Clean Architecture é o uso de interfaces para a comunicação entre camada. as interfaces definidas são um guia para cada camada, e considerando que as interfaces não mudem, cada camada pode mudar a forma como trata um operação, ou um serviço pode mudar, sem afetar o funcionamento do sistema. A seguir estão apresentadas as camadas utilizadas no projeto, sua função e os principais módulos contidos em cada camada.

- presentation: Essa camada pouco difere pelo definido pela arquitetura, a camada contém os entry points de todas as funções Lambda que serão utilizadas no projeto, essa camada é dividida em módulos que englobam todas as funções referentes a um fluxo específico da aplicação. Essa camada também é responsável pela validação dos dados de entrada, ou seja, qualquer parâmetro enviado para a função lambda é validado nesse primeiro momento e caso seja necessário a execução da função Lambda já é interrompida e uma mensagem de erro é enviado informando que houveram erros na validação de dados Nessa camada temos dois módulos principais: suggestion e classification, o módulo de suggestion é responsável por contém todas as funções de entrada referentes a busca e adição de propagandas dentro da aplicação. O módulo de classification é responsável por conter todas as funções de classificação de imagens, criação de perfil, e treinamento do modelo de machine learning.

- core: Essa camada engloba dois módulos principais: use-cases e data-sources, o módulo de use-cases é responsável por conter todas os arquivos referentes a lógica de negócios do projeto, neste módulo estão presentes funcionalidades como o parse de dados para atender as especificações do projeto, lógicas de paginação, etc. Essa camada pode ser considerada como o coração da função, ela é responsável por organizar toda a lógica da função e também é responsável por montar a saída da função e tratar eventuais erros de lógica e de serviço. O outro módulo presente nessa camada são os data-sources, esse módulo contém todas as interfaces de comunicação entre use-cases e serviços gerais, também contém a definição dos modelos de comunicação entre as camadas. De forma resumida este módulo define quais serviços podem ser chamados, que parâmetros devem ser passados para a função e o que ela irá devolver.

- data: Camada engloba todas as definições vinculadas com dados, nesse trabalho foi divida em três módulos: tables, dictionaries e services. O módulo de tables contém as definições de todas as tabelas do banco de dados, apesar de estar trabalhando com banco de dados não relacional, essas definições são importantes, pois permitem que a segurança dos dados que estão sendo armazenados não seja comprometida e impede que usuários maliciosos tentem inserir dados estranhos para a aplicação. O módulo de dictionaries contém arquivos gerados durante o treinamento do modelo que contém uma lista de todas as tags utilizadas durante o treinamento, esses arquivos foram salvos dentro do projeto pois não representavam uma grande quantidade de dados, mas caso fosse guardado em algum sistema de storage poderia acarretar em uma adição de latência indesejada na execução de algumas funções. Finalmente, o módulo de services contém todos os serviços utilizados dentro do projeto, cada

classe deste módulo implementa pelo menos um data-source e segue as suas definições implementado suas funções, permitindo que os serviços sejam facilmente trocas contando que elas sigam as definições estipuladas pelos data-source. Esse módulo contém as integrações com todos os serviços externos a aplicação (rekognition, elasticsearch, s3, etc..), implementado as principais usabilidades do serviços, um exemplo é que para o serviços de DynamoDB estão implementadas as principais funcionalidades de CRUD de banco de dados.

5.4 Visão Engenharia

A visão engenharia trata a distribuição de sistemas do projeto, ou seja, como o sistema se comunica entre cliente e servidor, para a realização do projeto foi utilizada uma infraestrutura de servidor cloud, utilizando como provedor a AWS. Esses serviços trabalham em conjunto para atender as necessidades de infraestrutura do projeto e simular uma estrutura de servidor clássico.

É importante ressaltar que toda a comunicação entre cliente e servidor foi feita através de uma API REST, utilizando como protocolo de comunicação o HTTP e toda a troca de dados foi feita utilizando JSON.

O projeto englobou um total de sete serviços da AWS, apresentados em três fluxos principais. A seguir serão explicados os três fluxos e como os serviços se ligam para atender todas as especificações do projeto. É importante ressaltar que os três fluxos apresentam funcionalidades da aplicação e não englobam o processo de treinamento do modelo, que foi feito através da execução de scripts diversos e será abordada durante a explicação do item 6.1.

O primeiro fluxo principal do projeto é iniciado quando um usuário tira uma foto dentro do aplicativo mobile, a partir desse momento começam as integrações com a AWS, é então feito upload da foto do usuário dentro do S3, a partir disso o upload de um arquivo específico em um determinado bucket no S3 aciona uma função Lambda, essa função valida o endereço da imagem e o id do usuário, enviado através de metadados da imagem e aciona um use-case responsável por fazer a análise da imagem, esse use-case então aciona o Rekognition enviando como parâmetro a foto em questão, com as tags de saída do Rekognition é acionado o endpoint do Sagemaker para realizar a previsão de qual cluster essa imagem se enquadra. Com a resposta do Sagemaker os dados da imagem, id do usuário e dados de cluster são salvos no DynamoDB.

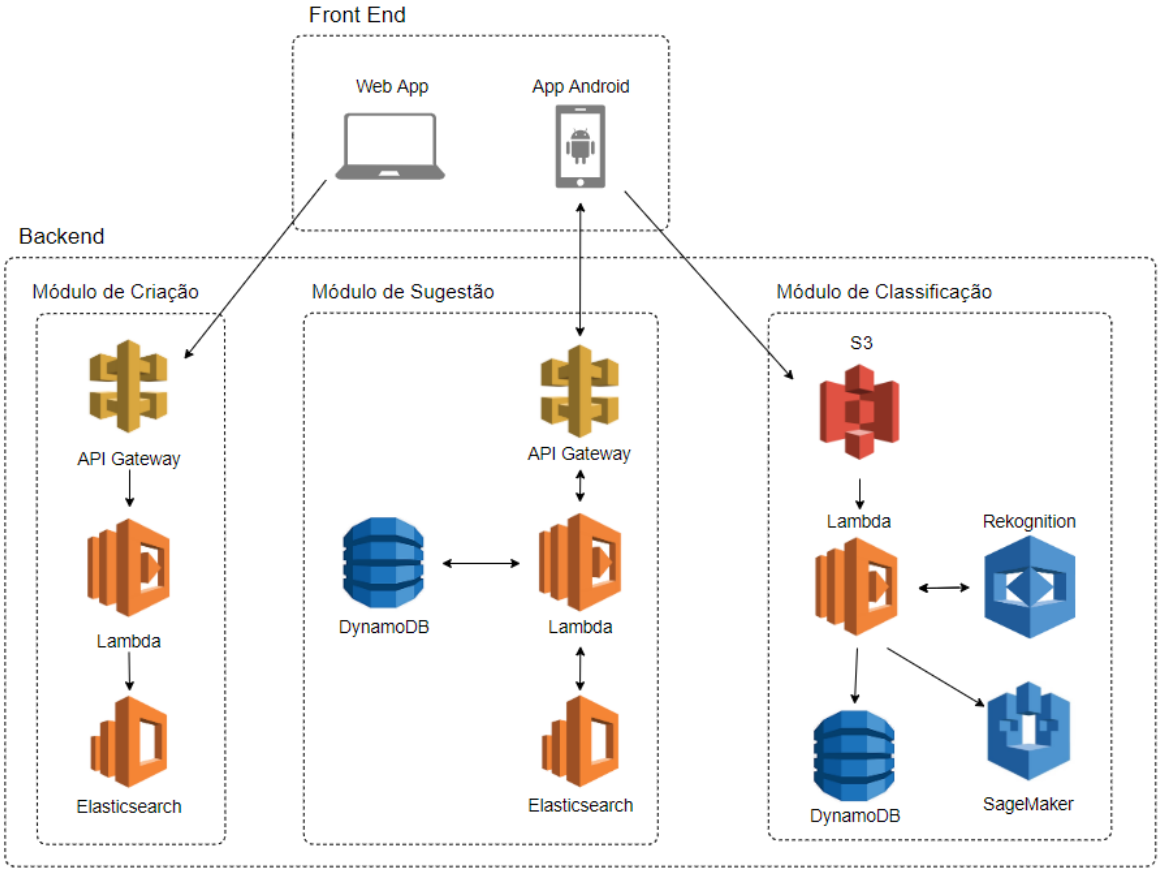

Figura 7: Diagrama da arquitetura de serviços

O segundo fluxo principal do projeto acontece quando o usuário acessa o seu feed, nesse momento a aplicação mobile faz uma requisição HTTPS, essa requisição é validada pelo API Gateway que redireciona os parâmetros enviados para uma função Lambda, essa função é responsável primeiramente por validar os dados enviados, em seguida um use-case é acionado para buscar as propagandas relevantes para o usuário, este use-case primeiro recupera um parcela de informações do Dynamo, contendo as informações das fotos tiradas pelo usuário, com isso é montado um modelo de perfil do usuário que é enviado para realizar um busca dentro do elasticsearch, essa busca leva em consideração a localização e as preferências do usuário, o retorno dessa busca são as propagandas que são então retornadas para o usuário como resposta dessa requisição HTTPS.

O terceiro fluxo engloba o processo de inserir uma propaganda dentro do sistema, apesar de ser o fluxo mais simples é o que traz mais valor para aplicação, dado que é o local em que ocorrerá o pagamento para inserir propagandas no sistema. Similar ao segundo fluxo, dentro de uma plataforma Web é feita uma requisição HTTPS, enviando para o servidor as informações da propagandas e as informações de pagamento, isso é então validado do lado do servidor e finalmente as informações são salvas dentro do Elasticsearch.

5.5 Visão Tecnologia

A Visão Tecnologia do projeto deixa claro as implementações do sistema, ou seja, quais as tecnologias utilizadas para o desenvolvimento do software do projeto. A implementação do projeto pode ser descrita com uma divisão em três partes: Desenvolvimento do Backend, Desenvolvimento do Frontend Mobile, Desenvolvimento do Frontend Web.

5.5.1 Backend

5.5.1.1 Serverless

Como mencionado no item 2.7, para o projeto foi utilizada uma arquitetura serverless, onde os serviços são instalados na nuvem de forma separada e acionados entre si. O principal framework utilizado para gerenciar aplicações desse tipo se chama Serverless Framework, ele permite que através de um arquivo de configurações escrito na linguagem YAML todos os serviços sejam implantados no provedor de nuvem escolhido através de um comando executado na linha de comando.

Nessa seção serão abordados as configurações que englobam o processo de deploy e de execução de um projeto serverless.

- provider: esse trecho de configurações informa as principais informações de provedor de cloud e as configurações necessárias para deploy dessas informações. Os principais parâmetros definidos aqui são o provedor (AWS), linguagem de execução das funções lambda, ambiente de desenvolvimento (dev), perfil de credenciais para execução do deploy e permissões de quais serviços da AWS as funções lambda irão acessar.

- plugins: define os plugins adicionais usados para execução do comandos do serverless. Os plugins usados foram: serverless-webpack utilizado para o transpile e build do código a ser executado, serverless-prune para remover versões antigas de código que são armazenadas pelo Lambda e serverless-offline que permite a execução do projeto de forma emulada, onde as funções ficam disponíveis para execução sem a necessidade de um deploy na AWS.

- resources: definição de todos os recursos a serem criados pelo serverless além das funções lambda e de seus eventos, por exemplo, com esses recursos podem definir um modelo de tabela para ser criada dentro do DynamoDB junto com o deploy da aplicação.

- functions: definição de todas as funções lambda a serem implantadas na AWS assim como a definição de todos os eventos que acionam essas funções lambdas

5.5.1.2 Node.js

O AWS Lambda permite executar código em algumas linguagens, para o projeto foi utilizado o Node.js, na sua versão 8.10, que é um interpretador de código Javascript, portanto todo backend foi escrito em javascript, com auxílio de algumas bibliotecas open-source como typescript (tipagem dentro do Javascript), class-validator (usado para validação de inputs), aws-sdk (biblioteca com sdk de todos os serviços da AWS), request-promise (biblioteca para gerenciamento de requisições HTTP), entre outras.

5.5.2 Frontend Mobile

Para o desenvolvimento do Frontend mobile, foi utilizada a linguagem nativa do android, Java, com auxílio do Android SDK, esse aplicativo funciona como um módulo de demonstração, onde é possível visualizar um feed de propagandas, tirar fotos e ver a sua galeria de fotos.

A ideia desse aplicativo é demonstrar o funcionamento do projeto, uma vez que essa funcionalidades descritas aqui já estariam implementadas dentro de outros aplicativos de rede social, e o nosso projeto entraria como uma biblioteca a ser utilizada por esses aplicativos, integrando dentro dos seus aplicativos já existentes.

5.5.3 Frontend Web

Para o desenvolvimento do Frontend Web, foi utilizada a biblioteca React. Esse módulo é responsável por ser uma plataforma onde anunciantes poderão criar propagandas, pagar por uma quantidade de visualizações, escolhidas por esse anunciante, e salvar esses dados de forma que a propaganda seja encontrada pelo sistema.

6 IMPLEMENTAÇÃO

Nesse tópico serão apresentados os principais módulos que compõe o sistema, como eles foram implementados, as principais dificuldades encontradas e os resultados alcançados.

6.1 Módulo de classificação

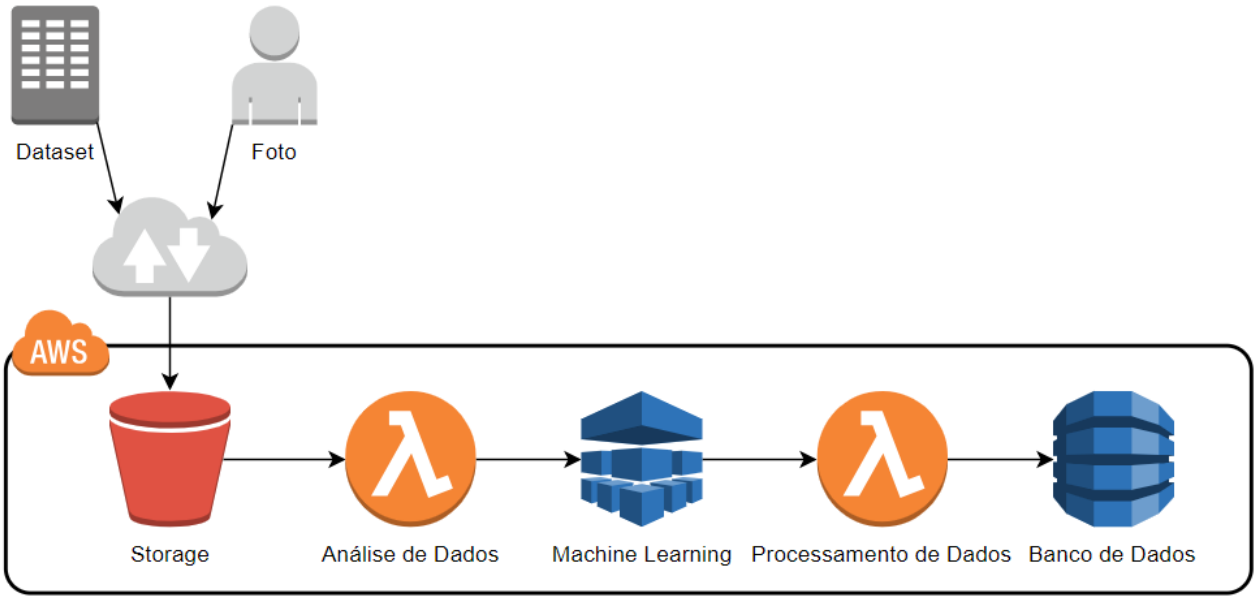

Figura 8: Diagrama do fluxo do módulo de classificação

6.1.1 Configurações Rekognition

O Rekognition não requer nenhuma configuração especial, é um serviço que pode ser utilizado diretamente pela SDK da AWS é necessário apenas a autenticação para associar a chamada do SDK com uma conta específica da AWS. O serviço cobra de forma indiscriminada, ou seja, as chamadas da SDK não estão associadas a um projeto ou domínio específico, mas apenas a conta vinculada.

A principal limitação encontrada pelo grupo durante o trabalho com o Rekognition foi o fato de ele suportar um máximo de 50 requisições por segundo, o que reduziu a eficiência de análise, inicialmente toda a análise de dados pelo rekognition era feita de forma paralela, ou seja, todas as imagens eram enviadas para o S3, o que disparava lambdas que faziam a análise das imagens e jogando os resultados em arquivos de texto de volta para o S3.

Para um pequeno volume de imagens esse procedimento não deu problemas, mas quando trabalhamos com uma quantidade de imagens na casa dos milhares o Rekognition começou a travar a execução da análise de imagens, o que forçou o grupo a reduzir a quantidade de imagens enviadas para o S3 com o intuito de reduzir o fluxo de dados requisitados ao Rekognition.

6.1.2 Aquisição de dataset de imagens

Um dos primeiros desafios encontrados foi o de montar um dataset de imagens que atendesse os requisitos do grupo, esse processo levou alguns meses que incluíram definição de categorias relevantes, busca de dados relevantes e todo processo de upload de dados na infraestrutura da nuvem.

Assim que começamos a trabalhar com o rekognition o grupo decidiu algumas categorias de interesse inicial, essas categorias seriam as categorias chave na hora de mostrar uma propaganda, ou seja, cada imagem tirada por um usuário seria classificada dentro da categorias definidas pelo grupo e na hora de sugerir uma propaganda seria consultado todo o histórico de fotos tiradas pelo usuário e com as classificações vinculadas seria sugeria um conjunto de propagandas organizadas pela relevância.

Nesse primeiro momento definimos um total de seis categorias: comida, esportes, viagem, beleza, saúde e animais domésticos. As categorias foram definidas através de pesquisa de opinião direta com colegas de faculdade, trabalho e familiares de forma informal, com o intuito de entender quais os tipos de propaganda mais comuns em redes sociais e quais os tipos mais comuns de fotos colocadas em redes sociais.

Com as categorias definidas o primeiro experimento do grupo foi adquirir um dataset de forma manual, ou seja, acessamos sites de imagens sem copyright, fizemos download dessas imagens e usamos elas como dataset inicial, nesse primeiro momento foi agrupado um total de 600 imagens (100 para cada categoria).

Nos primeiros testes percebemos algumas dificuldades que seriam muito difíceis de se-

rem superadas sem utilizar de um poder computacional muito maior e um dataset também muito maior, da forma como estávamos fazendo a classificação algumas categorias eram misturadas em uma mesma classificação, os detalhes sobre os resultados do treinamento serão abordados na seção 5.2.X. Como resultado decidimos por diminuir a quantidade de categorias, dado que o principal foco do trabalho é validar a ideia do produto por trás da implementação, com isso reduzimos para quatro categorias (comida, esportes, viagem e animais domésticos), mesmo assim as categorias de esportes e viagem acabavam se misturando, devido a dificuldade de definir quando uma foto é de viagem ou de esporte.

Como solução decidimos fazer duas principais modificações no projeto, a primeira é mudar as categorias novamente, utilizando objetos mais facilmente identificados e aumentar a quantidade de imagens do dataset.

Para as categorias resolvemos retirar as categorias que mais causavam conflito e adicionar outras duas categorias relevantes para o trabalho, com isso ficamos com as seguintes categorias: comida, animais domésticos, carros, bebês.

Em relação ao aumento da quantidade de imagens, seria inviável continuar com a estratégia de seleção de imagens manualmente, para resolver esse problema utilizamos datasets open source de imagens específicos para o treinamento de modelos de deep learning. Utilizando o dataset de imagens do ImageNet(Disponível em http://www.image-net.org/¿. Acesso em: 21 nov. 2018.) conseguimos agrupar um total de 4000 imagens (1000 para cada categoria).

Para o resto do desenvolvimento do trabalho, esse foi o dataset usado, contendo 4000 imagens de comida, animais domésticos, carros e bebês.

6.1.3 Refinamento da saída do Rekognition

O serviço do Rekognition utilizado chama-se detectLabels, com eles é possível identificar labels dentro de imagens e é atribuído uma confiabilidade para a label identificada, inicialmente todas as imagens do dataset eram analisadas utilizando o serviço de detectLabels e a saída dele era utilizada diretamente como entrada para o modelo de machine learning, após alguns testes percebemos que isso nos fornecia resultados errados.

O primeiro problema dessa abordagem é que o Rekognition retorna toda e qualquer label encontrada, não importando o nível de confiabilidade encontrado, portanto muito imagens retornavam labels que quando verificado nem se encontravam na imagem, portanto para garantir que utilizaríamos apenas labels que realmente se encontravam na

imagem, definimos um nível mínimo de confiabilidade, e antes de passar para o modelo de machine learning as labels que não atendessem esse nível mínimo eram filtradas.

Além disso encontramos outro problema, algumas labels encontradas, apesar de fazerem parte de imagem, e terem uma confiabilidade alta não eram o foco principal da imagem, mas quando esses valores eram passados para o modelo de machine learning ele entendia que essas labels indesejadas na verdade formavam uma nova categoria.

Para resolver esse problema, tivemos um trabalho de identificar as labels mais problemáticas e que não tinham uma categoria bem definida e mais uma vez fizemos umas filtragem das labels antes de enviar esses dados para o modelo de machine learning.

6.1.4 Treinamento do modelo de classificação utilizando o Amazon Sage Maker

Tendo o dataset de labels geradas a partir do processamento das 4000 imagens, passamos a implementar o algoritmo escolhido, K-Means, na plataforma do Sage Maker para possibilitar o treinamento do nosso modelo de aprendizado de máquina.

O primeiro passo foi configurar e provisionar um bloco de anotações Jupyter para a execução do tratamento necessário para inputar o dataset no algoritmo para treinamento.

Os cadernos Jupyter são uma tecnologia open source para edição de texto online que suportam a execução de código em 40 linguagens diferentes. Para a implementação do algoritmo do projeto foi escolhido Python. Para a execução do bloco de notas, provisionamos uma máquina na infraestrutura da AWS, pelo serviço EC2, e fizemos o deploy da aplicação pelo console do Sage Maker.

Com o caderno Jupyter em execução, acessamos a interface gráfica e importamos o código desenvolvido em Python para importação, tratamento dos dados e execução do treinamento do modelo. As etapas para o treinamento do modelo, portanto, se deram na seguinte ordem:

- Importação de dataset a partir de arquivo de texto salvo do S3:

  Como resultado do processo de análise da base de fotos categorizada, geramos um documento de texto contendo as labels inferidas a partir de cada foto. Tal arquivo foi armazenado no S3 e do ambiente do caderno Jupyter, estando disponível para os demais processos lerem o arquivo da memória.

- Leitura do arquivo de dataset e configuração do algoritmo:

Com o arquivo importado em memória na máquina rodando o caderno Jupyter, executamos um script para leitura do arquivo e tratamento do mesmo para uma matriz de floats. Em seguida, fazemos a importação do algoritmo KMeans, disponibilizado na sdk nativa do Sage Maker. Para a configuração do algoritmo, precisamos definir qual o dimensionamento de máquina queremos disponibilizar para o treinamento do modelo, qual o valor de K a ser aplicado, 4 no caso do projeto e caminhos para os outputs gerados.

- Alimentação do algoritmo com o dataset e treinamento do modelo:

  Com o algoritmo inicializado com o ambiente de treinamento, alimentamos o mesmo com o dataset tratado, em formato de matriz. Isso dispara o processo de treinamento do modelo, disponibilizando a infraestrutura definida na configuração do algoritmo e rodando o mesmo com o dataset fornecido.

- Deploy de endpoint de inferência:

  Em seguida, ao final do processo de treinamento iniciado acima, nosso script realiza o deploy do endpoint de predição, determinando qual o tamanho e tipo da instância de máquina EC2 que queremos disponibilizar para tratamento das inferências. A partir desse ponto, o Sage Maker faz o provisionamento de recursos para o deploy do endpoint de inferências e temos a nossa disposição um endpoint REST para realizar as consultas no modelo.

6.1.5 Estruturação de dados no DynamoDB

Com a saída do modelo treinado e com todas as informações referentes a foto tirado pelo usuário é necessário salvar esses dados para consultas futuras pelo módulo de sugestão, para isso os dados foram guardados em um banco de dados não relacional da AWS o DynamoDB.

Esse banco, apesar de ser não relacional, possui algumas peculiaridades, ao contrário de banco não relacionais tradicional, toda tabela possui uma chave primária e opcionalmente uma chave secundário (primary key e sort key respectivamente), o conjunto de chaves é chamado de partition key, e essa chave deve ser única. Esse é um comportamento comum em banco relacionais, mas foi introduzido no DynamoDB com o intuito de melhorar a performance e eficiência das buscas, dados que toda partition key de uma tabela tabela contém um index para agilizar as buscas.

Os dados do resultado do modelo são organizados da seguinte forma:

- primary key: id do usuário que tirou a foto, com isso é fácil em outro momento resgatar todas as fotos que um usuário tirou.
- sort key: horário que a foto foi tirada, isso permite organizar as buscas de forma rápida, ou seja, o próprio Dynamo é capaz de retornar os dados ordenados pela sort key, com isso podemos resgatar as entradas do banco inseridas mais recentemente dado que serão as informações mais relevantes (quanto mais nova a foto tirada pelo usuário provavelmente é algo que ele tem mais interesse no momento).
- outros dados: como estamos trabalhando com um banco não relacional os outros dados não são necessárias estruturados, e por sua vez não contém um index para a busca, eles são apenas retornados junto com as informações de primary key e sort key como dados do documento. As informações guardadas são o cluster mais próximo da imagem e a distância para esse cluster.

6.2 Módulo de sugestão

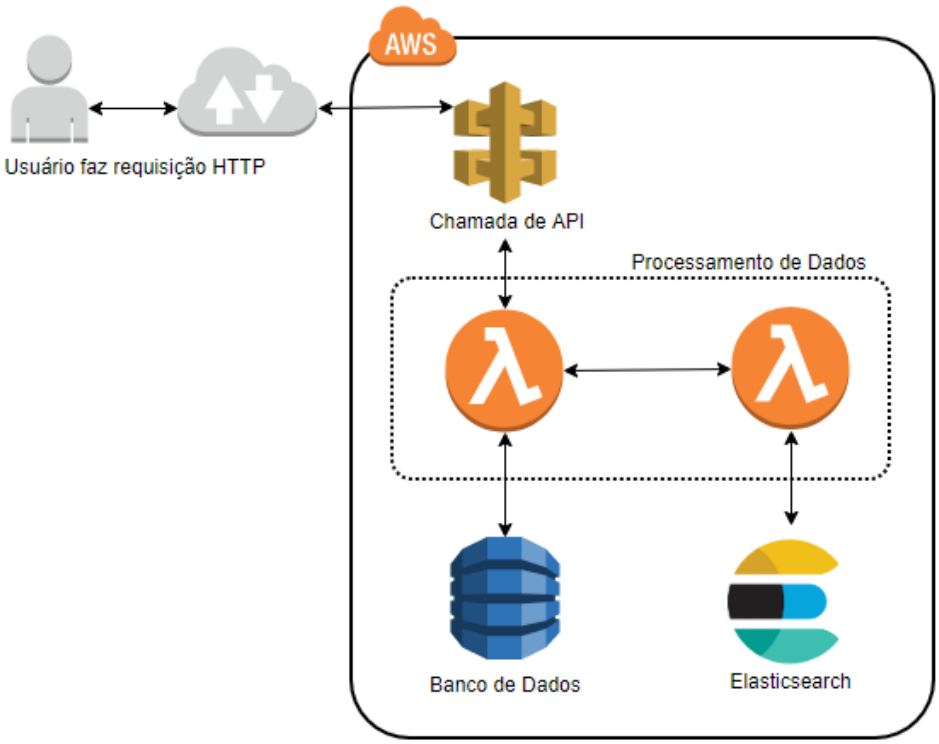

Figura 9: Diagrama do fluxo do módulo de sugestão

6.2.1 Criação da API

Para o módulo de sugestão foi necessário montar uma estrutura de API com endpoints para integração externa, dado que a ideia é que esse endpoint seja integrado dentro de outros aplicativos já existentes.

Foram criados dois endpoints HTTP para esse módulo, um deles usado por um administrador do sistema, que fará a criação das propagandas dentro do sistema e outro endpoints que retornará as propagandas mais relevantes para um determinado usuário.

Inicialmente a ideia era usar mais um algoritmo de machine learning para recomendar propagandas, ou seja, a partir de uma dataset de propagandas e de um perfil de entrada uma grande n de propagandas seria retornada, após alguns testes decidimos utilizar o algoritmo do Elasticsearch para fazer a parte de busca de propagandas.

A ideia de implementar o elasticsearch surgiu após verificar a dificuldade de implementar um algoritmo de machine learning de forma eficiente, a ideia inicial seria utilizar o algoritmo tanto na classificação quanto na recomendação, mas como a precisão do algoritmo poderia ficar prejudicada e a aquisição de um dataset que atendesse nossas necessidades seria difícil, decidimos seguir por essa outra abordagem.

A principal vantagem do elasticsearch para essa aplicação é a sua escalabilidade e o poder de caching associado às suas buscas, ou seja, buscas subsequentes com os mesmos parâmetros tendem a serem mais rápidas, e busca com mesmos inputs são ainda mais rápidas.

Outra vantagem é a facilidade de modificar os parâmetros da busca, os pesos que cada fator terão durante a busca são facilmente modificados e trazendo flexibilidade para a aplicação. Além disso o trabalho de ajustes da query foi muito menor quando comparado aos ajustes do modelo, pois não era necessário retreinar o modelo, apenas modificar alguns parâmetros e uma nova busca já estava pronta.

6.2.2 Configuração do Elasticsearch

Inicialmente o elasticsearch foi implementado localmente nos computadores pessoais do grupo, nesse período alguns testes foram realizados de forma a validar o funcionamento do algoritmo e testar alguns parâmetro de configuração do serviço.

O elasticsearch agrupa os dados em estruturas chamadas clusters, cada cluster possui uma quantidade n de nodes especificados pelo usuário. Além disso cada cluster pode

contém vários indexes, um index o equivalente a uma tabela de um banco de dados relacional, e cada index é alocado em shards, que são distribuídos dentro de nodes. Além disso shards podem ser replicados dentro de nodes diferentes de forma a garantir a integridade dos dados e evitar falhas na aquisição de dados.

Para esse projeto foi utilizado 1 cluster contendo 2 nodes, dentro dele foi criado um index para o armazenamento de propagandas, esse index possui um total de 5 shards, replicados dentro do node oposto a seu node original.

Com essa configuração inicial decidimos que seria necessário integrar o elasticsearch na forma de um serviço dentro da arquitetura serverless, o primeiro passo foi o de instalar o algoritmo, igual fizemos localmente, em uma máquina da EC2, e então iríamos acessar nosso cluster acessando a máquina através de uma porta aberta na rede da máquina da EC2. Essa ideia se mostrou mais complicada do que o esperado, dado que seriam necessárias configurações de redes complexas e seria necessário utilizar outro serviços para gerenciar o tráfego dentro dessa máquina.

Decidimos então utilizar o serviço da AWS específico de elasticsearch, esse serviço permitiu a abstração de várias configurações de uma máquina da EC2 além de ser um serviço que já gerencia o fluxo de rede dentro da máquina, permitindo que o grupo focasse apenas no desenvolvimento da aplicação em si.

Um vez que o software foi instalado foram necessárias então configurações de software, o AWS Elasticsearch fornece um endpoint do domínio que podemos fazer qualquer tipo de requisição para o nosso cluster, mas é necessário assinar essa requisição com as credenciais da conta AWS usada para configuração do serviço, para montar essa estrutura foram usadas duas bibliotecas, a primeira foi a biblioteca de client para javascript da própria Elasticsearch, ela possui uma abstração de todas as queries a serem realizadas, evitando a necessidade de uma verbosidade desnecessária na hora de montar as queries de busca. A outra biblioteca, chamada http-aws-es, foi utilizada para fazer a assinatura das requisições.

6.2.3 Criação de propagandas no Elasticsearch

Como comentado anteriormente o primeiro endpoint desse módulo diz respeito a adição de propagandas dentro do elasticsearch, esse endpoint foi feito de forma bem simples, onde o seu funcionamento simplesmente recebe dados de através do corpo da requisição, válida esses dados e em seguida salva eles dentro do elasticsearch.

Estes documentos irão conter várias informações relevantes tanto para a busca quanto

informações que serão mostradas para o usuário final, a lista de campos de cada documento está descrita a seguir:

- text: texto de descrição da propaganda, no trabalho foi usado apenas como um dado a ser mostrado ao usuário, mas poderia ser sido incorporado na busca, ou seja, poderíamos buscar propagandas contendo a palavra "Feijoada", assim seriam retornadas propagandas relevantes para o usuário e que contém a palavra Feijoada.
- imageUrl: imagem da propaganda, inicialmente seria a imagem analisada para realizar a classificação, mas se tornou apenas um dado a ser mostrado para o usuário.
- company: Empresa criadora da propaganda, mais uma vez pode ser usada como parâmetro da busca, para o projeto será usado apenas como um dado a ser mostrado.
- companyLogo: Logo da empresa criadora da propaganda.
- location: Contém as coordenadas de relevância para a propaganda, podem ser as coordenadas do estabelecimento ou apenas coordenadas de um ponto relevante para auxiliar a busca(Av. Berrini, Av. Paulista, etc).
- categories: Lista contendo um fator de relevância para cada uma das categorias, propagandas tem disponível um total de 100% de relevância a ser distribuído entre as categorias de forma a nichar o público desejado a ser atingido pela propaganda. Por exemplo, eu posso criar um propaganda de um novo drive thru, para isso eu vou criar um propaganda dentro do smart ad que tenho relevância de 80% em 'comida' e 20% de relevância em 'carros', assim eu irei atingir principalmente pessoas com um perfil voltado para comida e irei atingir em menor número pessoas com interesse em carros.

6.2.4 Query para busca de propagandas

Com os dados inseridos dentro do elasticsearch foi necessário fazer o segundo endpoint do módulo, esse endpoint será responsável por recuperar o perfil do usuário do DynamoDB e fazer uma busca dentro do elasticsearch, retornando assim as propagandas mais relevantes para um determinado usuário.

O primeiro passo desse endpoint foi o de validar os dados de entrada, esse endpoint possui duas entradas, o id do usuário que está requisitando as propagandas e a localização atual do usuário.

Em seguida as informações das últimas 50 fotos tiradas por um usuário são resgatadas do banco de dados, com essas informações é montado um perfil do usuário, seguindo a mesma lógica apresentada na criação de propagandas, usuário possui um total de 100% de relevância para o seu perfil, se 30 das 50 fotos forem de carros, e 20 das 50 fotos forem de comida esse usuário terá um perfil com fator 0.6 para carro e 0.4 para comida.

Com esses parâmetros definidos é feita uma busca no elasticsearch, em cima de toda a base de dados de propagandas, utilizando uma funcionalidade do elasticsearch de Function Score Query, essa funcionalidade permite personalizar a forma com que os documentos são classificados na busca, é importante ressaltar que as buscas no elasticsearch por padrão são ordenadas pelo seu score, esse valor é calculado pelo próprio elasticsearch usando métricas pré-definidas, mas esse cálculo pode ser customizado.

Para o projeto a busca foi realizada usando como parâmetros que influenciam a busca as categorias (comida, pets, carros, bebês) e a localização do usuário. Todos os parâmetros entram no cálculo do score, quanto mais próximo o perfil do usuário do perfil da propaganda e quanto mais perto a localização atual do usuário em relação a localização definida para a propaganda maior será o score atribuído a propaganda.

Para esse cálculo de proximidade foi utilizada uma função de decaimento sobre o ponto de referência, ou seja, existe um ponto central (localização do usuário ou fator atribuído a uma categoria do perfil do usuário) e partir dele é aplicada uma função de decaimento que influencia no valor do score, esse valor sempre se inicia em 1 e quanto mais longe do ponto de referência menor esse score fica, no limite, tendendo a zero.

O Elasticsearch disponibiliza o uso de 3 funções para tratar o decaimento: linear, exponencial e gaussiana. A função que mais se adequou aos testes feitos pelo grupo foi a função de gauss, dado que a resposta, ou seja, o decaimento da função, é a mais rápida, mas ainda assim é uma função que nunca chega ao zero.

Para cada parâmetro foi aplicado um função de decaimento, com isso foram obtidos 5 valores parciais de score, com isso esses valores são multiplicados e finalmente obtemos o valor final do score.

Para cada função são passados 4 parâmetros: o parâmetro que está sendo avaliado, origin que corresponde ao ponto de referência, scale que diz a distância do ponto de referência em que o dado atinge um determinado score e decay que corresponde ao valor de score no ponto de scale. Com esses parâmetros o elasticsearch calcula o valor de score para todos os documentos seguindo a seguinte fórmula:

$$\sigma^2 = -scale^2/(2 \cdot ln(decay))$$

$$S(doc) = \exp\left(-\frac{\max(0, |fieldvalue_{doc} - origin| - offset)^2}{2\sigma^2}\right)$$

Figura 10: Fórmula para o cálculo do score de um documento no Elasticsearch

Para o projeto todas as categorias utilizaram os mesmos parâmetros, sendo eles scale=0.5 e decay=0.1, para a localização foram utilizados os parâmetros de scale=25km e decay=0.1. Esses parâmetros foram definidos de forma empírica, ajustando os parâmetros de forma a obter os melhores resultados possíveis para alguns conjuntos de usuários definidos pelo grupo. Não houve grande esforço para definir os melhores parâmetros da busca dado que esses valores são altamente customizados, para determinadas aplicações a distância pode ter uma importância maior, ou uma área maior deve ser coberta para cada propaganda, por isso o foco foi em deixar esses valores facilmente customizados para cada aplicação e não necessariamente ter um foco em obter os melhores resultados para um caso de uso específico.

6.3 Módulo de demonstração (Aplicação mobile)

6.3.1 Criação do projeto Java

Para a criação do frontend mobile de demonstração, foi criado um projeto Android em Java utilizando o Android Studio 3.3.0. A partir do boilerplate de projeto fornecido pela IDE, se deu a elaboração das interfaces de 3 funcionalidades em estrutura de abas:

- Feed de propagandas:

  Para essa aba, foi elaborada uma estrutura de lista de postagens, seguindo uma modelagem simples de propaganda com: Logo da empresa, nome da empresa, imagem principal da postagem, descrição da postagem. Para exibir a lista, foi utilizada uma estrutura de RecyclerView vertical, com uso de um Adapter customizado com estratégia de ViewHolder para reaproveitamento das células renderizadas.

  Para renderização das imagens, nessa tela e nas demais, foi utilizada a biblioteca open source Picasso. Além de auxiliar na reciclagem de ImageViews e gerenciamento de memória, a biblioteca auxiliou download de imagens a partir de urls,

Figura 11: Tela do feed de propagandas

implementando estratégias de caching e redimensionamento das imagens. A lista de postagens a ser mostrada para cada usuário foi obtida de um endpoint REST como descrito abaixo no tópico 6.3.2.

- Upload de fotos:

  Essa interface tem o papel de integrar com os serviços nativos de câmera e galeria de imagens para permitir ao usuário selecionar a foto que será postada. Para isso foi utilizada a biblioteca open source PickImage. Esse biblioteca utiliza uma interface da Activity do android que permite que Activities se comuniquem, realizando uma operação e retornando um resultado para a Activity de origem. Dessa forma, após o usuário fazer a seleção a partir da galeria de fotos ou tirar uma nova foto pela câmera nosso sistema é capaz de identificar que a ação foi tomada e recuperar o

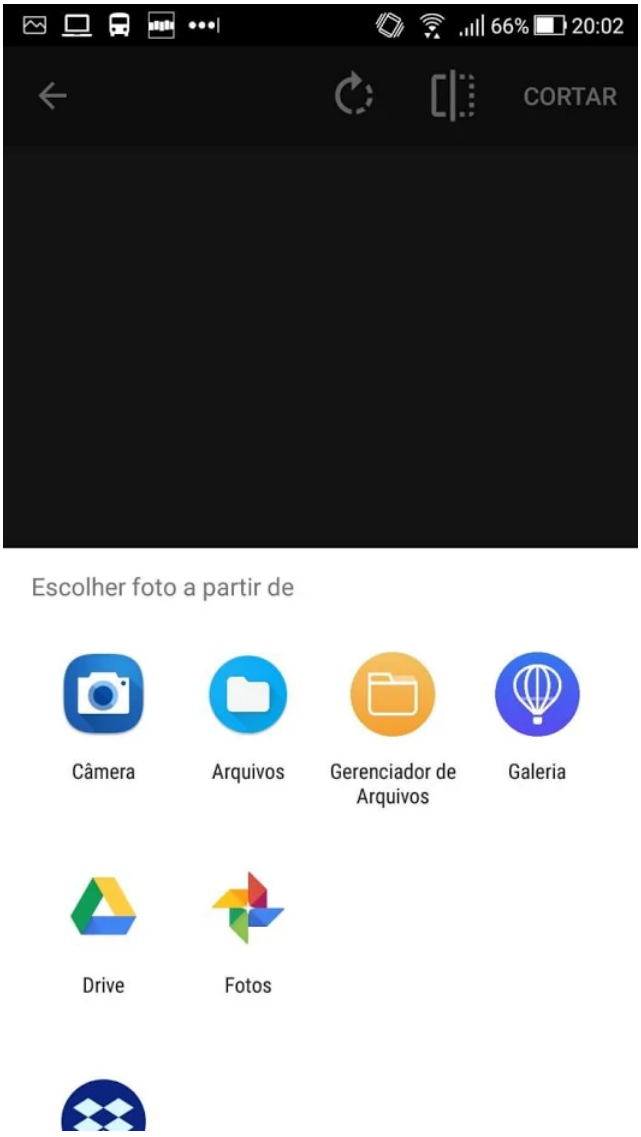

Figura 12: Tela de upload de fotos

caminho para o arquivo dentro do armazenamento interno do device, sendo capaz de renderizar o arquivo na tela.

Em seguida, para trazer uma experiência de usuário mais próxima das redes sociais, foi incorporada a biblioteca de código livre Android Image Cropper que fornece suporte para a implementação de fluxos de corte e edição de imagens. O arquivo recebido da interação do usuário com os recursos nativos do android de galeria e câmera é fornecido como fonte da imagem a ser editada, para então ser realizado o upload do arquivo editado.

A partir da imagem final escolhida e editada pelo usuário, dois processos em série são executados pelo sistema. Para possibilitar a análise da foto registrada e a composição do perfil do usuário que vai alimentar o feed de propagandas, é realizado o upload

da mídia gerada no backend. A implementação adotada para isso é a de upload do arquivo no S3 por meio da biblioteca de suporte da Amazon para java. A biblioteca oferece recursos avançados para o upload de mídias, realizando as operações em thread separada, com uma estratégia de stream por socket. Como resultado da operação de upload, a sdk fornece a url do arquivo no bucket do S3.

Em seguida, para possibilitar a exibição da imagem na tela de galeria de fotos, se utilizou do sistema de persistência local, a classe de SharedPreferences. A classe nativa do android, permite que aplicações realizem persistência local de objetos simples (Strings, ints, hashmaps etc) por meio de um sistema de chave-valor. Dessa maneira, a cada nova foto cadastrada, um hashmap de urls é atualizado com a url da nova foto.

- Galeria de fotos cadastradas:

  Por fim, foi implementada uma interface para visualização das fotos cadastradas pelo usuário. Para tal, foi utilizada uma estrutura de GridView, de 3 colunas, com uma estrutura simples de célula, apenas com a foto. Para alimentar o Adapter desse grid de imagens, extraímos do armazenamento local pela classe SharedPreferences a lista de urls das imagens cadastradas.

6.3.2 Integração

Além do upload de mídias, realizado pela biblioteca de suporte da AWS para Java, como descrito acima, a aplicação também interage com o backend para obter a lista de posts a ser exibida no feed personalizado do usuário. Para tal, foi implementada uma estrutura de integração utilizando a biblioteca Retrofit. A biblioteca de comunicação HTTP, apresenta a vantagem de ser fortemente tipada, auxiliando no correto processamento de respostas REST e facilitando o processamento de respostas em JSON para objetos estruturados do Java.

Para identificação única do usuário, no momento em que o mesmo faz o primeiro uso da aplicação, é gerado um identificador único universal (UUID) que é salvo na classe android de persistência local SharedPreferences. A partir desse momento, as interações do usuário com a plataforma são identificadas com esse UUID, sendo elas o upload de mídias para análise e a consumação do endpoint de lista de propagandas.

Para obter o feed personalizado, a aplicação realiza uma chamada de protocolo POST com body em JSON contendo dois parâmetros: o UUID do usuário e as coordenadas de

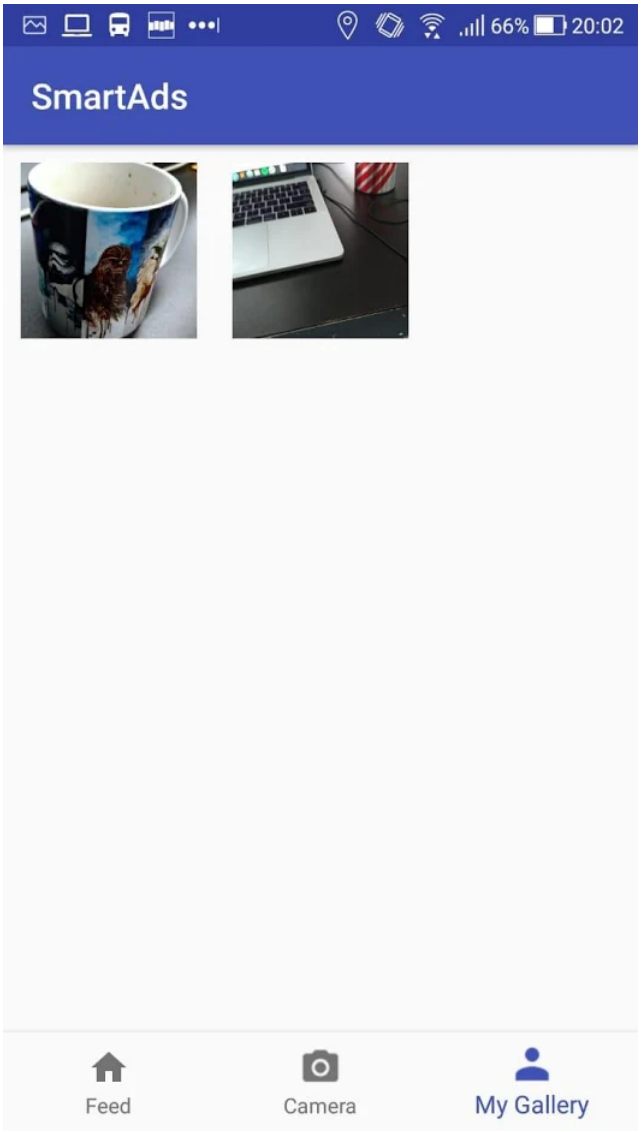

Figura 13: Tela de galeria de fotos

localização do device. Para a obtenção da localização do device, foi implementado um serviço de gerenciamento de localização utilizando a API nativa de serviços de localização da Google.

7 CONSIDERAÇÕES FINAIS

7.1 Conclusões do Projeto de Formatura

O projeto se demonstrou muito relevante para a formação acadêmica dos integrantes do grupo, dado que com a realização do mesmo foi possível aprender uma série de novas tecnologias e conceitos. Foi o primeiro contato do grupo com inteligência artifical de forma geral, e com o trabalho o grupo foi capaz de treinar um modelo de machine learning e utilizá-lo de forma integrada a um sistema complexo.

Além disso todo o processo de montagem da arquitetura, utilizando conceitos de arquitetura na nuvem modernos, além do uso de arquiteturas de software diferentes, que se apoiam em conceitos de arquitetura consolidados. Todo o processo de aprendizado desses conceitos foi muito produtivo para a formação profissional dos membros do grupo.

Em relação aos resultados do projeto, o resultado foi satisfatório, utilizando o algoritmo K-Means foi possível obter uma acurácia na previsão de uma categoria para uma dada imagem de 80%, o que atende as expectativas do projeto, pois o perfil do usuário é montado a partir de uma grande amostragem de imagem, portanto com essa precisão foi possível obter resultados satisfatórios para a sugestão de propagandas.

Apesar de não obtermos uma precisão tão alta, chegando próximo dos 100%, para a demonstração do projeto isso se mostrou suficiente e foi capaz de validar a ideia do produto desenvolvido, como a ideia é transformar isso em uma plataforma a ser comercializada, é fácil perceber que a necessidade de customização e modificação serão necessárias, portanto como uma prova de validação da ideia original, e tendo em vista os objetivos do projeto, podemos concluir que os resultados foram satisfatórios.

7.2 Contribuições

O projeto trouxe um novo paradigma para o marketing mobile, atualmente a maioria das empresas de marketing digital utilizam de métodos tradicionais para fazerem a divulgação de produtos, utilizam de força bruta para mostrar uma grande quantidade de anúncio para uma grande quantidade de pessoas na esperança de um pequeno percentual de usuários se interessar pelo produto e realmente efetuar uma compra.

A maior empresa desse ramo, o Google, utiliza o histórico de acessos de um usuário para realizar a sugestão de propagandas, mas isso não é muito efetivo levando em consideração que muitas vezes uma pessoa pode acessar um site sem querer, ou apenas por curiosidade e isso já fica marcado no rastro digital da pessoa e é usado para sugerir propagandas que podem não ser do interesse do usuário.

O sistema do SmartAd entra como uma inovação na forma como são divulgados os anúncios, levando em consideração a grande adesão de usuários a aplicativos de rede social, que muitas vezes possuem funcionalidades de foto dentro de seus aplicativos, o nosso projeto junta essa grande disponibilidade de dados, que possuem muito mais relevância para o usuário para criar um perfil e com isso sugerir propagandas aos usuários desses aplicativos.

Além disso a nossa solução se torna economicamente mais viável, devido ao menor número de visualizações de cada propaganda, mas com um direcionamento mais preciso, possibilitando que anunciantes divulguem menos seus anúncios, mas para os usuários certos, aumentando a conversão de vendas e reduzindo o custo de anunciar.

7.3 Trabalhos Futuros

Existem algumas melhorias que poderiam ser implementadas no projeto de forma a deixá-lo mais completo e de fato transformá-lo em um produto comercializável.

- Retreinar o modelo: Da forma como o projeto foi realizado, a classificação de imagens está restrita a um conjunto de categorias definida pelo treinamento do modelo, qualquer imagem que não se adeque as categorias pré-definidas é descartada, essas imagens poderiam ser utilizadas para melhorar a acurácia do modelo, ou seja, o próprio modelo se realimenta e melhora a sua eficiência.
- Adicionar mais categorias: O trabalho de treinamento do modelo se mostrou muito

complicado devido a dificuldade em encontrar um dataset ideal e adequar os parâmetros do algoritmo de forma a obter um modelo que atenda as necessidades do projeto, isso levou a um número final de categorias pequeno (apenas quatro categorias), em um momento futuro mais categorias poderiam ser adicionadas.

- Criar um sdk para integração externa: Apesar do protótipo desenvolvido pelo grupo ser excelente para demonstração do produto, ele não é útil para integração do nosso produto em outros aplicativos, seria necessária a criação de um sdk que faria todo o interfaceamento com o nosso sistema e que pudesse ser integrado em aplicações

- Sistema de cobrança: Finalmente, para que o negócio possa operar e ser sustentável, seria necessário criar um sistema de pagamento onde a inserção de uma propaganda dentro do sistema gera um pagamento a ser realizado para o anunciante, além ser necessário uma plataforma para que o aplicativos parceiros possam receber dinheiro pelas propagandas exibidas.

REFERÊNCIAS

[1] GOOGLE. **Consumer Barometer**. Disponível em: <https://www.consumerbarometer.com/en/>. 03 nov. 2018.

[2] IBGE. **Pesquisa Nacional por Amostra de Domicílios Contínua**. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv101543.pdf>. Acesso em: 03 nov. 2018.

[3] RUSSEL, N. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 2nd. ed., 2003.

[4] GOOGLE. **Machine Learning Crash Course**. Disponível em: <https://developers.google.com/machine-learning/crash-course/>. Acesso em: 03 nov. 2018.

[5] WU, X. KUMAR, V. **Top 10 algorithms in data mining**. Disponível em: <https://link.springer.com/article/10.1007%2Fs10115-007-0114-2>. Acesso em: 10 nov. 2018.

[6] HOF, R. D. **Deep Learning**. Disponível em: <https://www.technologyreview.com/s/513696/deep-learning/>. Acesso em: 10 nov. 2018.

[7] ROBERTS, M. **Serverless Architectures**. Disponível em: <https://martinfowler.com/articles/serverless.html>. Acesso em: 03 nov. 2018.

[8] ELASTIC CO. **Elasticsearch Reference [6.5]**. Disponível em: <https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html>. Acesso em: 03 nov. 2018.

[9] CAELUM. **O que é Java**. Disponível em: <https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/#java>. Acesso em: 03 nov. 2018.

[10] AMAZON WEB SERVICES. **What Is Amazon Rekognition?** Disponível em: <https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html>. Acesso em: 04 nov. 2018.

[11] AMAZON WEB SERVICES. **O que é Amazon DynamoDB?** Disponível em: <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>. Acesso em: 04 nov. 2018.

[12] AMAZON WEB SERVICES. **Em que consiste o Amazon Elasticsearch Service?** Disponível em: <https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/what-is-amazon-elasticsearch-service.html>. Acesso em: 04 nov. 2018.

[13] MELL, P. GRANCE, T. **The NIST Definition of Cloud Computing**. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>. Acesso em: 01 dez. 2018.

[14] BLANK, R. GALLAGHER, P. **NIST Special Publication 800-53**. Disponível em: <https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-53r4.pdf>. Acesso em: 01 dez. 2018.

[15] AMAZON WEB SERVICES. **NIST 800-53 Standardized Architecture on the AWS Cloud**. Disponível em: <https://aws.amazon.com/pt/about-aws/whats-new/2016/01/nist-800-53-standardized-architecture-on-the-aws-cloud-quick-start-reference-deployment/>. Acesso em: 01 dez. 2018.

[16] MARTIN, R. C. **The Clean Architecture**. Disponível em: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>. Acesso em: 02 dez. 2018.

[17] SERVERLESS. **Serverless Documentation**. Disponível em: <https://serverless.com/framework/docs/>. Acesso em: 02 dez. 2018.

[18] AMAZON WEB SERVICES. **Algoritmo k-means**. Disponível em: <https://docs.aws.amazon.com/pt_br/sagemaker/latest/dg/k-means.html>. Acesso em: 10 nov. 2018.

[19] ANDERSEN, B. **Complete Guide to Elasticsearch**. Disponível em: <https://www.udemy.com/elasticsearch-complete-guide/learn/v4/content>. Acesso em: 10 nov. 2018.

[20] DEVELOPERS ANDROID. **Interface do usuário**. Disponível em: <https://developer.android.com/guide/topics/ui/>. Acesso em: 19 nov. 2018.

[21] DEVELOPERS ANDROID. **API reference**. Disponível em: <https://developer.android.com/reference/>. Acesso em: 19 nov. 2018.

[22] SQUARE. **Package com.squareup.picasso**. Disponível em: <http://square.github.io/picasso/2.x/picasso/>. Acesso em: 19 nov. 2018.

[23] VANSUITA, J. **PickImage**. Disponível em: <https://github.com/jrvansuita/PickImage>. Acesso em: 19 nov. 2018.

[24] DEVELOPERS ANDROID. **Interação com outros aplicativos**. Disponível em: <https://developer.android.com/training/basics/intents/>. Acesso em: 19 nov. 2018.

[25] HUB, A. **Android Image Cropper**. Disponível em: <https://github.com/ArthurHub>. Acesso em: 19 nov. 2018.

[26] AMAZON WEB SERVICES. **AWS SDK for Android**. Disponível em: <https://github.com/aws-amplify/aws-sdk-android>. Acesso em: 20 nov. 2018.

[27] DEVELOPERS ANDROID. **App data and files**. Disponível em: <https://developer.android.com/guide/topics/data/>. Acesso em: 20 nov. 2018.

[28] SQUARE. **Retrofit**. Disponível em: <https://square.github.io/retrofit/>. Acesso em: 20 nov. 2018.

[29] LEACH, P. MEALLING, M. SALZ, R. **A Universally Unique IDentifier (UUID) URN Namespace**. Disponível em: <https://tools.ietf.org/html/rfc4122>. Acesso em: 20 nov. 2018.