Sistema para Identificação de Emoções utilizando Machine Learning

Gabriel da Silva Cardoso dos Santos[1], Reiner Franthesco Perozzo[1]

[1]Curso de Bacharelado em Ciência da Computação – Universidade Franciscana – 97010-032 – Santa Maria – RS - Brasil

gabriel.santos@unifra.edu.br, reiner.perozzo@unifra.br

Abstract. *This paper presents a system that identifies the emotion of the user, realizing emotions recognition by means of a mobile device and emotion API by Azure platform. The data obtained with the API are compared with the answers from a questionnaire, serving as a basis for verification between what the user has defined and what the API identifies through machine learning techniques. In the end, there are the implementation of a system for generating statistical data aimed at the emotions detected by the target audience.*

Resumo. *Este trabalho apresenta um sistema que identifica a emoção do usuário, realizando o reconhecimento de emoções, por meio de um dispositivo móvel e da API de reconhecimento de emoções da plataforma Azure. Os dados obtidos com a API são comparados com respostas de um questionário, servindo como base para a verificação entre o que o usuário definiu e o que a API identifica por meio de técnicas de aprendizado de máquina. Ao final, tem-se a implementação de um sistema para geração de dados estatísticos voltados às emoções detectadas de um público-alvo.*

1. Introdução

A utilização de tecnologias, nas mais diversas áreas de atuação, estão cada vez mais em uso na atualidade. Hoje em dia, é possível visualizar o emprego de *softwares* como ferramentas facilitadoras que automatizam processos e ajudam a corrigir e melhorar os serviços que são prestados aos clientes, por exemplo. Além disso, estabelecimentos que possuem serviços e atendimentos que estão em constante busca por melhorias e diferenciais a serem apresentados para o seu público-alvo, crescem cada vez mais e tornam-se requisitados em um mercado competitivo [Pizzolato 2015].

Nesse sentido, para uma empresa torna-se importante obter dados e conhecimento a respeito da satisfação/sentimento de seu público-alvo, pois uma vez que a empresa tiver conhecimento sobre o que seus clientes pensam a respeito de seus serviços/produtos, existirá uma base para a criação de estratégias, campanhas publicitárias e outros fatores que influenciam no crescimento de uma empresa ou na ampliação de venda de seus serviços ou produtos [Prass 2012]. Da mesma forma, é importante para uma empresa saber o nível de satisfação de seus funcionários em relação o ambiente em que trabalham, pois, com esse conhecimento, a empresa pode atender as melhoras no ambiente de trabalho, dando maior motivação aos funcionários e consequentemente eles poderão gerar mais e melhores resultados [Soares 2014].

A partir disso, este trabalho consiste no desenvolvimento de um sistema capaz de realizar a pesquisa quanto o sentimento que um certo público apresenta quando é colocado um questionamento ou conforme o local de sua aplicação. Dessa forma, a empresa tendo um sistema com essas características, possivelmente apresentará melhorias em seus serviços, produtos e/ou atendimento, além de obter vantagens diante de seus concorrentes.

1.1. Objetivo Geral

O objetivo deste trabalho é o desenvolvimento de um sistema para auxiliar no controle de satisfação de usuários, por meio de reconhecimento facial e aprendizado de máquina. Bem como, o teste, quanto a precisão, de uma *Application Programming Interface* (API) para detecção de emoções a partir de um rosto contido em uma imagem.

1.2. Objetivos Específicos

- Desenvolver uma aplicação móvel para coleta de dados e apresentação de interface gráfica ao usuário;
- Integrar API de reconhecimento de emoções à aplicação móvel;
- Desenvolver um sistema *web* para recebimento e armazenamento de dados para análise e geração de relatório;
- Desenvolver interface gráfica ao usuário na parte *web*;
- Analisar a precisão da API quando comparada com a resposta do próprio usuário quanto sua emoção.

2. Referencial Teórico

A seguir, serão apresentados nas subseções Revisão Bibliográfica e Trabalhos Correlatos, conceitos e trabalhos/soluções relacionados ao tema que foi escolhido para este trabalho. A intenção é contextualizar sobre assuntos que serão abordados no decorrer deste trabalho, auxiliando no entendimento dos conceitos e tecnologias que serão utilizados na proposta.

2.1. Revisão Bibliográfica

Na seguinte subseção, serão abordados os principais temas deste trabalho, contemplando *Machine Learning*, *Knowledge Discovery in Databases* (KDD), *web service*, aplicações móveis e *Microsoft Emotion API*, respectivamente.

2.1.1. Machine Learning

O *Machine Learning*, capacita a aprendizagem da máquina sem a necessidade de ser programada explicitamente. Nesse sentido, esse aprendizado de máquina concentra-se no desenvolvimento de programas que podem acessar dados e, consequentemente, obter conhecimento sobre determinado assunto. Assim, pode ser possível determinar ou sugerir sobre aquilo que está sendo utilizado [Expert System 2017].

O processo, para que ocorra o aprendizado, deve ter como base a observação de padrões em dados para que aos poucos ocorra uma melhora quanto a tomada de decisões sobre determinado assunto. O *Machine Learning* pode fazer uso de diferentes técnicas para atingir o seu propósito, como o aprendizado supervisionado e não supervisionado.

2.1.1.1. Aprendizado supervisionado

O aprendizado supervisionado é uma técnica comumente utilizada para realizar previsões que possivelmente possam ocorrer no futuro. Para essa forma de aprendizado, é necessária a entrada de dados que já possuam resposta. Ao inserir um conjunto de entradas e relativamente suas saídas, as quais devem estar corretas, o algoritmo utilizado para a aprendizagem consegue relacionar e comparar a saída real com as saídas corretas, resultando no encontro de erros [SAS 2015].

2.1.1.2. Aprendizado não supervisionado

Diferente do aprendizado supervisionado, o aprendizado não supervisionado é a técnica utilizada em algoritmos que não possuem uma resposta. A ideia é buscar nos dados que estão sendo analisados a possível resposta para os dados de entrada. Dessa forma, o aprendizado não supervisionado, busca encontrar padrões ocultos ou estruturas características dos dados que foram inseridos como entrada [Mathworks 2016].

2.1.2. Knowledge Discovery in Database (KDD)

Em projetos os quais necessitam armazenar dados para futuras requisições, o KDD se mostra eficiente. Esse processo funciona de maneira a extrair informações, ainda desconhecidas do banco de dados, mas que podem ser de grande relevância. Para Fayyad et al. (1996), o KDD é composto por etapas, mas antes de começá-las, deve-se ter entendimento sobre a área de aplicação e, além disso, identificar o objetivo da aplicação do KDD. Após isso, inicia-se a etapa de seleção, a qual define os dados a serem absorvidos. A etapa de pré-processamento recolhe apenas o que é necessário dos dados. Partindo para a etapa de transformação, busca-se obter as características úteis conforme o objetivo. Na mineração de dados, ocorre a mescla de métodos e análises para obter um padrão de busca. Por fim, na etapa de interpretação os dados que chegaram até essa etapa são interpretados e avaliados.

As próximas subseções, irão descrever a respeito de duas das técnicas (ou algoritmos) utilizadas na descoberta de conhecimento em base de dados. Ambas, frequentemente, relacionadas à mineração de dados.

2.1.2.1. Regra de associação

A regra de associação se detém na associação de dados que condizem com uma certa frequência de acontecimentos, com uma base de condições, ou seja, com a ocorrência de outros fatores comumente associados. O desenvolvimento de uma regra de associação se baseia na forma como os dados são associados. Dessa forma, obtendo-se as medidas necessárias, podem ser usados algoritmos como o *Apriori*. Eles que irão definir se as regras geradas são eficientes e confiáveis para seu uso em análises [Pichiliani 2008]. O algoritmo *Apriori* faz a retirada de dados a partir de um conjunto. Assim, são formadas as medidas de suporte e confiança, formando uma regra de associação.

2.1.2.2. Reconhecimento de padrões (RP)

Com o objetivo de procurar atos que ocorrem com certa frequência, o reconhecimento de padrões classifica e descreve aquilo que for relevante para o contexto. Os algoritmos de reconhecimento de padrões buscam a melhor resposta possível para determinada

situação, extraindo características dos dados que estão sendo processados, selecionando as características que tornam o dado único e construindo um classificador [Sá 2000].

A classificação de dados pode ser de forma supervisionada, fazendo uso de algoritmos capazes de classificar com base no que já sabem. A classificação supervisionada geralmente é utilizada em reconhecimento ótico, como detecção e reconhecimento de faces. Já a classificação não supervisionada busca informações que estão escondidas em dados desconhecidos utilizando técnicas de segmentação ou *Clustering*, utilizada na mineração de dados [Mathworks 2013].

2.1.3. Web service

Web service pode ser considerado como um *software* de utilização em múltiplas plataformas, independentemente da linguagem. Além disso, ele é hospedado em algum local endereçável de rede. Um sistema baseado em *web service* pode ser utilizado em integrações com outros sistemas, fazendo a comunicação, por exemplo. Por ser uma tecnologia que abrange diversas plataformas e com diferentes compatibilidades, ele é capaz de emitir e receber dados no formato de alguma das linguagens universais, como *eXtensible Markup Language* (XML) ou JavaScript Object Notation (JSON), por exemplo. O XML, por exemplo, é uma linguagem de marcação para a qual as demais linguagens das outras plataformas podem ser traduzidas.

O *Simple Object Access Protocol* (SOAP), é um dos protocolos responsável por realizar a comunicação entre aplicações por meio do protocolo *Hypertext Transfer Protocol* (HTTP). Quanto a segurança dos *web services*, existem alguns mecanismos como o *Secure Socket Layer* (SSL). O SSL transforma o HTTP em *Hyper Text Transfer Protocol Secure* (HTTPS), ou seja, é um mecanismo de proteção em nível de transporte, cuidando da autenticação, da integridade e da privacidade dos dados que estão sendo transportados pelo serviço.

2.1.4. Aplicações móveis

Aplicações móveis (*apps* ou aplicativos), são *softwares* desenvolvidos para *smartphones* e *tablets* (plataformas móveis). Elas podem ser categorizadas como nativas e híbridas. Os aplicativos nativos são aqueles que foram desenvolvidos para uma plataforma específica, como *apps* desenvolvidos na linguagem *Swift*, rodando apenas em sistemas operacionais (SO) iOS, ou nas linguagens Kotlin ou Java, que funcionam apenas em dispositivos com o SO *Android*. Já os aplicativos híbridos, combinam elementos de aplicativos nativos com os aplicativos de *web*, como um site, por exemplo. Uma exceção ocorre no desenvolvimento de aplicativos para os dispositivos iOS, sendo necessário um computador com o SO equivalente, o macOS. Porém, esse SO funciona apenas nos computadores da *Apple*, tornando o desenvolvimento caro para os padrões brasileiros, comparado à países desenvolvidos.

2.1.5. Microsoft Emotion API

A *Emotion API* é um dos serviços cognitivos desenvolvidos pela *Microsoft* dentro de um conjunto abrangente de diversos serviços em nuvem, chamado *Azure*. Com a *Emotion API*, o sistema onde é aplicada será capaz de detectar as emoções de uma ou mais pessoas, por meio de uma imagem armazenada ou em tempo de captura. Assim, por esse motivo, a *Emotion API* foi escolhida para ser utilizada neste trabalho.

A API funciona com uma imagem contendo pessoas como dado de entrada e retorna um JSON. O JSON retornado contém porcentagens aproximadas quanto a tendência para um conjunto de emoções, entre elas: raiva, desinteresse, aversão, medo, felicidade, neutralidade, tristeza e surpresa, conforme pode ser observado na Figura 15a do Anexo A. Para a detecção do rosto em uma imagem, a *Emotion API* desenha um retângulo sobre a(s) face(s) (na Figura 15b em azul no Anexo A) por meio da API de Detecção Facial, a qual trabalha em conjunto com as demais APIs [Microsoft 2016].

O retorno do JSON é uma matriz com as entradas faciais e as porcentagens para cada emoção. Desta forma, basta o sistema interpretar os resultados e conforme as suas necessidades definir um limite de confiança, ou então tomar como emoção resultante aquela que for interpretada com a maior porcentagem, sabendo que a normalização das porcentagens de cada emoção é dada para resultar em apenas uma. Vale ressaltar que a API aceita imagens no formato *Joint Photographic Experts Group* (JPEG), *Portable Network Graphics* (PNG), *Graphics Interchange Format* (GIF) (apenas o primeiro quadro) e *Bitmap* (BMP), com tamanhos menores que 4 *megabytes* (MB). A API detecta no máximo 64 rostos. Ainda existem alguns desafios por conta do ângulo da imagem, da posição em que a cabeça da pessoa se encontra, da iluminação, ou de emoções como desprezo e nojo.

2.2. Trabalhos Correlatos

Nesta subseção, o objetivo é trazer soluções e trabalhos que apresentam os mesmos conteúdos que foram desenvolvidos neste trabalho. Dessa forma, os seguintes trabalhos correlatos são utilizados como indicadores e base para o trabalho que aqui se descreve.

2.2.1. Cainthus

Cainthus é um *software* que surgiu com o intuito de realizar o reconhecimento facial de vacas para acompanhar seus comportamentos em geral, mas dando foco para a alimentação. O sistema não é voltado apenas para a agropecuária, mas também para a agricultura. No geral, o *Cainthus* monitora a saúde e o bem-estar dos animais e das plantações, sem atrapalhar no cotidiano dos animais e dos trabalhadores. O processo de análise é feito por meio de imagens e funciona com qualquer *hardware* que forneça informações visuais. O *software* é capaz de fazer reconhecimento facial e corporal, verificar se o animal está mancando, analisar as atividades no rebanho e se os animais estão se alimentando de forma correta [Cainthus 2016].

Para o acompanhamento dos rebanhos, câmeras precisam ser instaladas em locais relevantes para tal aplicação. Assim, enquanto o animal fica parado, o sistema inicia, de forma individual, o processo de leitura facial, conforme ilustrado na Figura 16 presente no Anexo B, pegando as principais características presentes na imagem. Após esse processo, a pessoa responsável pelo controle do rebanho pode acompanhar as atividades dos animais.

2.2.2. Controle de acesso inteligente usando Deep Learning

Esse trabalho descreve a aplicação de *Deep Learning* em um sistema de controle de acesso residencial. *Deep Learning* é uma técnica de *Machine Learning*, que busca ensinar computadores a tomar decisões assim como o ser humano faz naturalmente. O treinamento pode ocorrer por meio de dados rotulados e arquitetura de redes neurais

contendo diversas camadas [Mathworks 2017]. Os autores desse trabalho, descrevem que em um sistema de aprendizagem, existem várias camadas neurais criadas por vários neurônios. Então, o sistema proposto por eles aplica quatro camadas quando ocorre a captura do rosto de uma pessoa. Após a captura, a imagem formada é inserida em uma rede neural tendo como valor uma matriz de pixels [Lee 2017].

A partir de então, começa-se a aplicação das camadas, sendo que na primeira camada as informações características que estão contidas nas bordas da imagem serão aprendidas pelo sistema. Na segunda camada, as informações aprendidas anteriormente, serão organizadas de forma especifica em padrões básicos (olhos, nariz, boca e orelhas, conforme definido pelos autores). Na terceira camada, ocorre a combinação de padrões básicos, que são unidos para obter um rosto. Por fim, na quarta camada, será verificado se este rosto é ou não um rosto cadastrado no sistema.

A aplicação desse modelo de *Deep Learning*, pode ser treinado com grandes quantidades de dados, tendo apenas que fazer os ajustes necessários conforme a aplicação. Existem alguns modelos pré-treinados que inclusive podem ser obtidos publicamente [Lee 2017]. O modelo utilizado no artigo foi o VGGNet, treinado com o conjunto de dados do ImageNet. Apesar da pequena quantidade de dados, a taxa de reconhecimento, tanto das imagens que possuíam cães e gatos como as que possuíam humanos, foi relativamente alta, segundo os autores.

2.2.3. NeoFace Watch

Projetado para a integração com sistemas de segurança, o *NeoFace Watch*, desenvolvido pela *NEC IT Soluctions*, observa imagens instantâneas, buscando rostos indicados em uma lista. Ele pode realizar o reconhecimento facial em situações consideradas complicadas como quando existem variantes de envelhecimento, pessoas em movimento e quando a imagem está mal enquadrada [NEC 2013]. O *NeoFace Watch* é um sistema baseado na *web* e para seu funcionamento, o sistema deve estar conectado com uma câmera, que fará a captura das imagens. O algoritmo do *software* é capaz de analisar imagens capturadas no momento ou gravadas, extraindo rostos presentes na imagem e comparando com os rostos presentes em uma lista preenchida pelo usuário. Dessa forma, quando algo for encontrado, é gerado um alerta instantâneo.

2.2.4. Considerações sobre os trabalhos relacionados

Quanto aos trabalhos descritos, pode ser observado o desenvolvimento e o estudo sobre técnicas e métodos que visam a obtenção de informações de forma inteligente. No geral, os trabalhos apresentados desenvolvem um estudo com tecnologias capazes de englobar o aprendizado de padrões, possibilitando seu uso em varias aplicações.

Seus pontos positivos são notáveis, tanto em aplicação como em funcionamento. Pode-se destacar áreas bastante requisitadas como o reconhecimento de padrões, o *Deep Learning*, o KDD, ou seja, o *Machine Learning* em geral, pois essa é a tecnologia que faz os sistemas serem considerados inteligentes e eficazes em suas tarefas. Ainda existem alguns gargalos a serem tratados, pois o reconhecimento facial em humanos não é cem por cento correto, existem dificuldades com o reconhecimento de expressões e os sistemas que estão no mercado possuem um custo elevado.

Dessa forma, as funcionalidades e técnicas usadas nos sistemas, no estudo e na API que foram descritas, se encaixam com a ideia objetiva deste trabalho, que é o

desenvolvimento de um sistema capaz de identificar a satisfação de um público, por meio de técnicas inteligentes do *Machine Learning* e testar a precisão da API comercial.

3. Metodologia

Este trabalho utiliza a metodologia *Feature Driven Development* (FDD), que é uma metodologia de desenvolvimento ágil com a característica de iniciar o desenvolvimento a partir dos requisitos funcionais do sistema. Para isso, o FDD possui cinco processos: o desenvolvimento de um modelo abrangente, a construção da lista de funcionalidades, o planejamento por funcionalidade, o detalhamento por funcionalidade e a construção por funcionalidade [Gomes 2013]. Assim, o FDD se destaca por apresentar resultados a curto prazo, ter um planejamento detalhado, ter a capacidade de se adaptar e ser uma metodologia que se orienta conforme as necessidades do cliente, gerente e desenvolvedor [Heptagon 2017]. Diante disso, a seguir são descritos os requisitos, a lista de funcionalidades e o planejamento por funcionalidade.

3.1. Requisitos

Considerando que os requisitos são o ponto de partida para a metodologia FDD, podem ser observados os requisitos para este trabalho no Apêndice A.

3.2. Lista de funcionalidades

Como destacado anteriormente, as funcionalidades de um sistema têm papel importante no desenvolvimento da metodologia FDD. Dessa forma, a partir dos requisitos do presente trabalho, no Apêndice B, pode ser encontrada a lista de funcionalidades.

3.3. Planejamento por funcionalidade

No que tange o planejamento de cada atividade que foi desenvolvida no decorrer do projeto, cada atividade foi alocada por semanas, estando dentro do período proposto para o desenvolvimento do projeto. A tabela com as atividades desenvolvidas pode ser observada no Apêndice C.

4. Proposta

Dado o cenário apresentado, em que se pode observar o crescimento de áreas tecnológicas que buscam se unir outras áreas de forma inteligente, a presente proposta contempla o desenvolvimento de um sistema capaz de realizar o controle de satisfação sobre determinado assunto, por meio de leitura facial. Em síntese, este trabalho busca a coleta de informações voltadas à satisfação de um público alvo, utilizando técnicas de reconhecimento de padrões.

Dessa forma, este trabalho consiste na implementação de um *software* que está disponível em plataformas computacionais móveis e serviços computacionais na parte *web*. O *software* é capaz de realizar a identificação de emoções do público, por meio de leitura facial, apresentando dados resultantes da leitura facial e das informações preenchidas pelo próprio usuário. Além disso, também é analisada a precisão da *Emotion API* da *Microsoft*. A seguir, na Figura 1, é apresentada uma visão geral da ideia proposta neste trabalho.

Considerando um cenário de utilização do sistema, no qual existe um dispositivo móvel (no caso da Figura 1, um totem de atendimento contendo um *tablet*), o sistema

apresenta um breve questionário ao usuário. Enquanto o usuário termina de responder, a câmera do dispositivo captura uma foto para realizar a detecção de seu rosto e realizar a análise de suas emoções naquele momento.

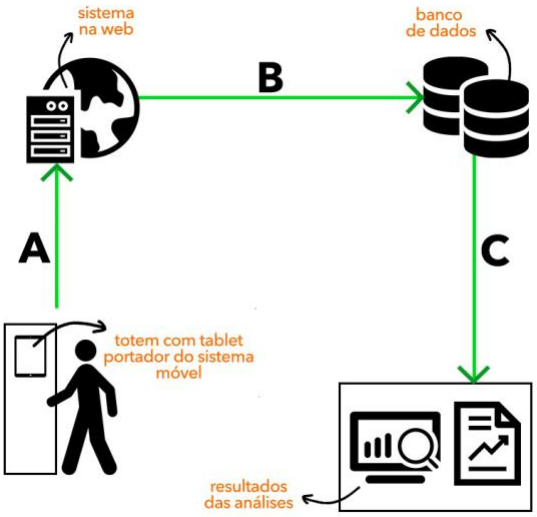


Figura 1. Representação do funcionamento do projeto.

O processo A da Figura 1, representa o envio das respostas preenchidas no questionário para o sistema *web*. O sistema *web* é responsável pela geração de dados estatísticos e apresentação dos resultados para serem feitas as análises e comparações daqueles dados que já estão armazenados no banco de dados. Esse BD recebe as informações que o compõem após o término das operações realizadas no sistema *web*, sendo representado pelo processo B.

Assim, com uma base formada, o sistema possibilita a geração de dados estatísticos para o usuário (representado no processo C). Também são apresentadas as diferenças entre o que o próprio usuário definiu como sua emoção e o que a API apresenta como emoção detectada na mesma pessoa. Podendo assim haver as comparações desses dados e verificação da precisão da API para tal aplicação.

4.1. Projeto

Nesta subseção é apresentado o diagrama de classes para ilustrar, os componentes do sistema e as suas associações. Na Figura 14 do Apêndice D, é possível observar a existência das classes “EmotionAPI”, “Data”, “Connection”, “DataBase”, “GUI” e “Reports”, que são responsáveis por realizar de forma geral a interação do sistema com o usuário e a disponibilização de estatísticas.

Por meio dessas classes, o sistema apresenta um breve questionário de dois campos para coletar a emoção e a nota para essa emoção, porém indicadas pelo próprio usuário, ao final é realizada a detecção da emoção por meio da captura facial. A classe “EmotionAPI” representa as funcionalidades que a API da *Microsoft* apresenta para este trabalho, que seria a detecção do rosto e da emoção. Também busca pelos pontos direito e esquerdo do rosto e o quão acima ou abaixo estão do centro da boca. Essa classe trabalha a questão do uso da câmera de forma oculta por meio da API *HiddenCamera*.

A classe “Connection” realiza o envio dos dados obtidos pelo aplicativo móvel que estão presentes na classe “Data” e envia para o sistema *web*.

Partindo para parte *web* desse trabalho, os dados enviados pela classe “Connection” são recebidos na classe “DataBase”, que além de recebe-los, realiza a conexão com o BD e a inserção destes dados recebidos. A interface presente no sistema *web* é gerado pela classe “GUI”, a qual faz uso da API para geração do gráfico (Chart.js) e busca no banco de dados pelas informações que precisa para ser composta. Por fim, a classe “Reports” é responsável por gerar o relatório de pessoas que passaram pelo sistema e apresentar dados estatísticos, assim como o mesmo gráfico presente na classe “GUI”, porém para a data em que o relatório foi gerado.

4.2. Implementação

Esta subseção apresenta a implementação do projeto, em que as classes e as suas principais funções são expostas. Uma visão do funcionamento do sistema é apresentada pelo diagrama de atividade presente no Apêndice E.

4.2.1. Aplicação Móvel

As classes que compõem este trabalho em sua forma móvel, são as classes “EmotionAPI” e “Connection” (Figura 14 do Apêndice D). De modo geral, estas duas classes são responsáveis pela detecção da emoção e pelo envio dos dados obtidos para a aplicação *web*. A aplicação móvel, utiliza o ambiente de desenvolvimento *Android Studio*, com as linguagens XML e Java, para *layout* (Figura 2) e desenvolvimento respectivamente.

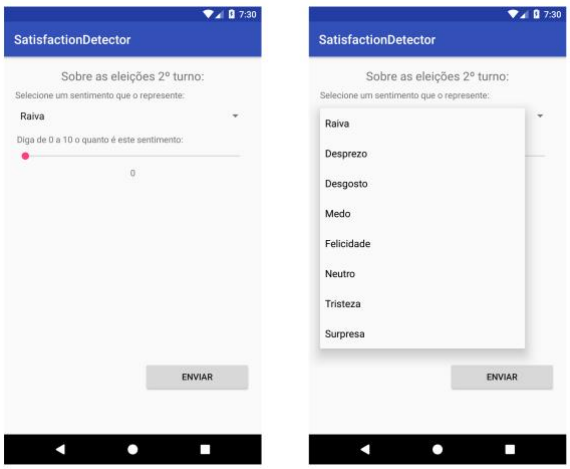


Figura 2. Interface aplicação móvel.

A classe “EmotionAPI” é responsável pela captação da emoção selecionada pelo próprio usuário e uma nota para tal emoção, isso tudo em relação a um determinado assunto. Além disso, essa classe possui a principal função da API, a qual realiza o processo de captação e definição da emoção expressa pelo rosto presente na imagem.

Antes da detecção de emoção, é necessária a detecção da face presente na imagem. Para essa finalidade, é utilizado o método “faceServiceClient.detect” (Figura 3), que assim com os demais métodos da *Emotion API* da *Microsoft*, está pré-definido na biblioteca de Detecção Facial “com.microsoft.projectoxford:face:1.4.3”. Este método

realiza uma requisição via POST à *Uniform Resource Locator* (URL) definida na variável "apiEndpoint" contendo em seu cabeçalho a chave de assinatura da *Emotion API*, definida em uma variável chamada "subscriptionKey".

```
@SuppressLint("DefaultLocale")
@Override
protected Face[] doInBackground(InputStream... params) {
    try {
        publishProgress( ...values: "Detectando...");
        Face[] result = faceServiceClient.detect(
                params[0],
                b: true,         // returnFaceId
                b1: true,        // returnFaceLandmarks
                new FaceServiceClient.FaceAttributeType[] {
                        FaceServiceClient.FaceAttributeType.Emotion
                }
        );
        if (result == null){
            publishProgress( ...values: "Detecção Finalizada. Nada detectado.");
            return null;
        }
        publishProgress(String.format("Detecção Finalizada. %d face(s) detectada(s)", result.length));
        return result;
    } catch (Exception e) {
        exceptionMessage = String.format("Falha na detecção: %s", e.getMessage());
        return null;
    }
}
```

Figura 3. Trecho do código presente na função "detectAndFrame" da classe "EmotionAPI".

Em chamadas bem-sucedidas (*Response 200*), é retornada uma matriz de faces, contendo os valores "faceId", "faceRectangle", "faceLandmarks" e "faceAttributes", os quais, exceto "faceRectangle", necessitam ter valor *true* nos parâmetros de chamada do método "returnFaceId" e "returnFaceLandmarks" respectivamente, enquanto que para o parâmetro "returnFaceAttributes" é necessária uma nova chamada ao método "FaceServiceClient.FaceAttributeType" especificando o tipo de atributo, no caso emoção. Os valores necessários para este trabalho estão contidos em "faceLandmarks" que apresentam os pontos X e Y dos cantos direito e esquerdo da boca ("MouthRight" e "MouthLeft"), e também os valores presentes em "faceAttributes" que são alguns atributos, entre eles, o que está sendo utilizado é o atributo *Emotion*.

As posições em que os cantos direto e esquerdo da boca se encontram (pontos X e Y de cada canto) são essenciais para saber o quão acima ou abaixo do eixo 0 (centro da boca) os cantos estão. Para isso, é utilizada a função "mouthInformation", que recebe como parâmetro a face detectada e a partir dela com o método "faceLandmarks" são obtidos os pontos X e Y dos cantos direito e esquerdo da boca, e também os pontos Y onde se encontram o topo do lábio superior e a base do lábio inferior. Estes valores são armazenados em variáveis para a manipulação no cálculo do ponto médio da boca (eixo 0 citado anteriormente). Tendo o ponto médio da boca é possível dizer quanto acima ou abaixo os cantos da boca estão do meio, por isso é trabalhado apenas o Y dos pontos.

```
public void sendHttp(String emotion,
                     float porc_emocao,
                     String emocaoForm,
                     String notaEmocaoForm,
                     float boca_dirX,
                     float boca_esqX,
                     float info_boca_dir,
                     float info_boca_esq,
                     String data, Context context) {

    WebView web = new WebView(context);
    web.loadUrl(host + "?emocao=" + emotion
            + "&porc_emocao=" + porc_emocao
            + "&emocaoForm=" + emocaoForm
            + "&notaEmocaoForm=" + notaEmocaoForm
            + "&boca_dirX=" + boca_dirX
            + "&boca_esqX=" + boca_esqX
            + "&info_boca_dir=" + info_boca_dir
            + "&info_boca_esq=" + info_boca_esq
            + "&data=" + data);
}
```

Figura 4. Código da função "sendHttp" da classe "Connection".

A classe “Connection” transmite todos os dados coletados na aplicação móvel para a aplicação *web*. Ela possui uma variável do tipo *string* contendo o endereço para onde serão enviados os atributos recebidos. O método chamado “sendHttp” (Figura 4), recebe como parâmetros, dados que serão recebidos pela aplicação *web* e então armazenados no BD. Entre os dados, está a emoção informada pelo próprio usuário, uma nota para tal emoção, a emoção detectada pela API, o percentual de certeza quanto a emoção, o ponto X dos cantos direito e esquerdo, o quão acima ou abaixo estão os cantos da boca, a data em que a detecção foi realizada e o contexto (quanto a aplicação móvel Android) de onde foram retirados os dados, porém este ultimo não é enviado para a aplicação *web*. Então é chamada uma WebView Android para realizar o envio HTTP para o endereço definido na variável global da classe. O envio é feito por meio do método “WebView.loadUrl”, que ocorre pelo método de envio HTTP chamado GET. Por essa razão, os dados que o método “sendHttp” recebe por parâmetro, são enviados à aplicação *web* em forma de variáveis na própria URL.

4.2.2. Aplicação Web

A seguir, nesta subseção serão apresentadas as classes presentes na aplicação *web* desse trabalho. As classes “DataBase”, “GUI” e “Reports” são responsáveis por receber os dados da aplicação móvel, conectar-se com o BD e então armazena-los, apresenta-los de forma estatística e prepará-los para a geração de eventuais relatórios. Estas são classes desenvolvidas na linguagem *PHP: Hypertext Preprocessor* (PHP). A aplicação em si, apresenta em seu *dashboard* o gráfico que demonstra os pontos direito e esquerdo da boca quanto o centro, e a tabela contendo os dados obtidos com a API, como os pontos presentes no gráfico (Figuras 5 e 6). Para isso, foi utilizado o *framework* Bootstrap, o qual trata-se de um *framework web front-end* de código aberto e trabalha usando *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript.

Figura 5. Gráfico da interface web.

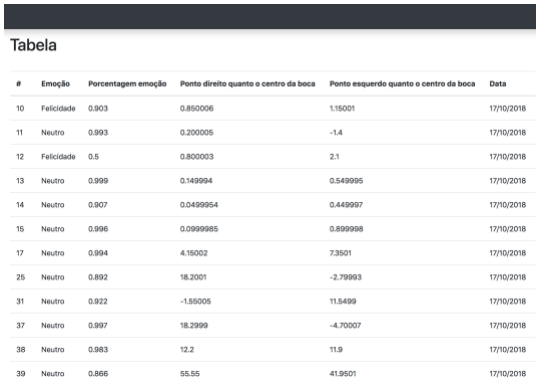


Figura 6. Tabela da interface web.

A classe “DataBase” faz a recepção dos dados vindos da aplicação móvel e os coloca em variáveis para facilitar na manipulação. Possui uma função para conexão com o BD e uma para inserir os dados no banco de dados MySQL para consultas no momento da apresentação de informações e geração de relatórios. A conexão e manipulação de dados no BD é dada por meio do *PHP Data Objects* (PDO), que é uma das formas de se trabalhar o PHP com os bancos de dados. Desta forma, são passados o endereço, nome, usuário e senha do banco de dados no método de conexão do PDO. Com a conexão aberta, representada pelo método “connectionDb”, é possível realizar a

inserção de dados no BD. A conexão é posta em uma variável chamada "conn", a qual é utilizada na chamada do método PDO "*prepare*" (Figura 7) que recebe a sintaxe *Structured Query Language* (SQL) referente a inserção de dados (*INSERT*) no caso desta classe. E então com o método "*execute*" é realizada a requisição ao BD.

```
$sql = $conn->prepare('INSERT INTO face (emocao, porcentagem_emocao, emocaoForm,
notaEmocaoForm, boca_dirX, boca_esqX, info_boca_dir, info_boca_esq, data) VALUES (:emocao,
:porc_emocao, :emocaoForm, :notaEmocaoForm, :boca_dirX, :boca_esqX, :info_boca_dir,
:info_boca_esq, :data)');
$sql->execute(array(
   ':emocao' => $emotion,
   ':porc_emocao' => $percEmotion,
   ':emocaoForm' => $emotionForm,
   ':notaEmocaoForm' => $noteEmotionForm,
   ':boca_dirX' => $mouthRightX,
   ':boca_esqX' => $mouthLeftX,
   ':info_boca_dir' => $infoMouthRight,
   ':info_boca_esq' => $infoMouthLeft,
   ':data' => $date
));
```

Figura 7. Método "prepare" do PDO recebendo sintaxe SQL para inserção no BD.

Na classe "GUI", a qual se refere ao arquivo "index.php", são realizados o tratamento e a consulta ao BD para apresentar os dados obtidos com a aplicação móvel. Esta é uma classe que trabalha tanto com a linguagem HTML como a linguagem PHP, uma vez que o *layout* da aplicação é construído a partir do HTML e as consultas ao BD são feitas por meio da linguagem PHP também fazendo uso do PDO. Neste momento, também é utilizado o método "connectionDb" como o da classe "DataBase", porém nessa classe é necessário realizar apenas a seleção (*SELECT*) dos dados desejados com o método "selectToTable" e "selectToChart", mudando a sintaxe SQL no método "*prepare*" chamado para a variável "conn" dessa classe (Figura 8).

```
$stmt = $conn->prepare("SELECT boca_dirX, info_boca_dir, boca_esqX, info_boca_esq FROM
face WHERE emocao = 'Felicidade' AND boca_dirX != 0");
$stmt->execute();
$mouthHappy = $stmt -> fetchAll();
```

Figura 8. Processo para captação dos dados buscados no BD.

A tela que faz a apresentação dos dados, expõem um gráfico mostrando as regiões de maior acumulo dos cantos da boca quanto a sua emoção e uma tabela mostrando alguns dados vindos do BD. Para a apresentação do gráfico, está sendo utilizada uma biblioteca JavaScript, chamada Chart.js. Esta biblioteca é de código aberto e permite a ilustração de diversos tipos de gráficos (no caso deste trabalho é utilizado o gráfico *Bubble*) por meio da *tag* HTML "canvas". Para apresentação dos dados no gráfico, é realizado um "*SELECT*", representado pelo método "selectToChart" que faz a seleção das emoções felicidade, neutro, triste e raiva. Assim, os resultados da busca são colocados um a um dentro de um vetor por meio do método PHP "fetchAll". Desta forma, é possível percorrer o vetor e então mostrar cada dado conforme sua posição no vetor um a um. No fim, é exibido o gráfico que é gerado conforme os dados recebidos da consulta. Quanto a tabela, para serem apresentados os dados, a forma é a mesma, porém são realizados os "*SELECTs*" necessários para obter as informações desejadas, representados pelo método "selectToTable".

O código JavaScript que constrói o gráfico, chama o método "Chart", o qual recebe o id da *tag* HTML onde será exibido o gráfico resultante e um objeto que indica o tipo de gráfico, uma matriz de dados como o nome, cor e outra matriz contendo os pontos X e Y e o raio que a bolha receberá (Figura 9), e as opções de apresentação, como titulo do gráfico e sua escala na página *web*.

```
label: ["Felicidade - Ponto direito boca"],
backgroundColor: "rgba(255,221,50,0.2)",
borderColor: "rgba(255,221,50,1)",
data: [
   <?php
   foreach ($mouthHappy as $row) {
      echo "{
         x: ".$row["boca_dirX"].",
         y: ".$row["info_boca_dir"].",
         r: 15
      },";
   }
   ?>
]
```

Figura 9. Trecho do código JavaScript para construção do gráfico.

Por fim, a classe "Reports" funciona como as anteriores quanto a conexão (método "connectionDb") e consulta ao banco de dados (método "selectToChart"). Porém, ao ser chamada, é passada uma data por método POST. Essa data é responsável pelo filtro de data ao gerar o relatório, ou seja, a data selecionada é a data em que a quantidade de pessoa, as emoções e as médias de porcentagens foram obtidas. Desta forma, o "*SELECT*" apresentados nos métodos "selectToChart" e "generateReport" é o mesmo, acrescentando apenas o "AND" da sintaxe SQL para a data e também fazendo o uso do método "AVG" no caso do "generateReport", o qual também pertence a sintaxe SQL e é utilizado para calcular a média das porcentagens, como pode ser observado na Figura 10 para a consulta para a emoção "Felicidade". O funcionamento do botão "Gerar relatório" se dá por meio da função "window.print()" do JavaScript.

```
$average = $conn->prepare("SELECT AVG(porcentagem_emocao) FROM face WHERE emocao =
'Felicidade' AND data = '$date'");
$average->execute();
$percHappy = $average -> fetchAll();
$averageHappy = 0;
foreach( $percHappy as $row ) {
   $averageHappy = $row["AVG(porcentagem_emocao)"];
}
$averageHappy = $averageHappy * 100;
```

Figura 10. Trecho de código do método "generateReport" da classe "Reports".

5. Cenário de avaliação e testes

Após a fase de desenvolvimento, foi construído um cenário para avaliação e testes do sistema proposto. A identificação de um cenário para aplicação do sistema foi realizada visando os requisitos funcionais presentes no projeto, ou seja, o cenário precisava demonstrar aplicação para as funcionalidades do sistema e verificar a precisão da *Emotion API*. Desta forma, o sistema proposto foi testado nas áreas comuns da Universidade Franciscana (UFN).

Para isso, foi levado em conta o momento politico do país, o qual se encontrava em fase de preparações para o segundo turno quanto a decisão da presidência do Brasil, contendo assim maior gama para as opiniões e relativamente emoções. Assim, por meio de um *smartphone* com a aplicação móvel do sistema, o usuário entrevistado informava a emoção que estava sendo sentida em relação ao segundo turno das eleições e logo em seguida era dada uma nota para essa emoção. Enquanto é dada a nota à emoção, é realizada a captura da face por meio da câmera frontal do *smartphone*, gerando a informação de emoção e porcentagem da emoção extraídas da face. Foram coletados dados de 100 pessoas em um intervalo de 11 dias.

6. Resultados e discussões

Com o teste do sistema e eventualmente da API, alguns fatores foram possíveis notar. Muitas variáveis no ambiente afetam os resultados na coleta de dados, uma vez que quanto mais imóvel estiver o dispositivo em um ponto, mais precisa será a API e os dados estatísticos gerados a partir dela e do questionário. Por esta questão e por conta da conexão instável com a internet, alguns resultados vindos da *Emotion API* foram descartados por terem sido dados como nulos. Porém, na Figura 11, pode ser observada a comparação do que a API detectou e das respostas vindas dos usuários entrevistados.

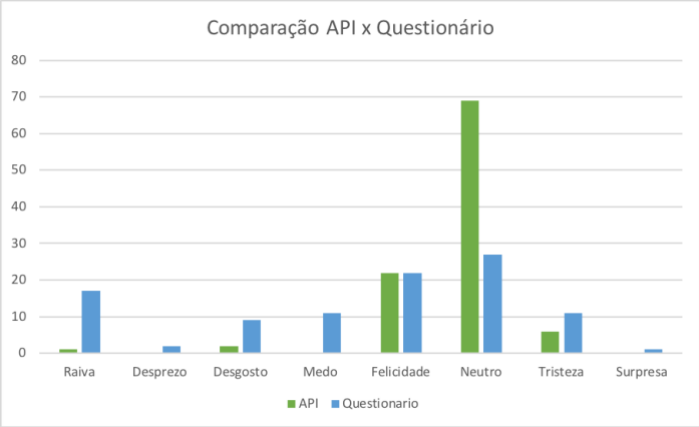


Figura 11. Gráfico da relação de emoções identificadas pela API e as respostas vindas do questionário.

Na tabela abaixo, as emoções de porcentagem zeradas demonstram que a API não detectou a emoção que o usuário respondeu no questionário. É possível notar que a *Emotion API* é bastante precisa para as emoções "Raiva" e "Felicidade", tendo uma pequena diferença na "Tristeza" e uma diferença considerável para o "Neutro". As demais emoções mostram-se mais complicadas de serem detectadas pela API, sendo que estas têm suas expressões faciais um pouco mais complicadas de serem definidas pelo algoritmo da *Microsoft*. Isso se dá também, pela quantidade de pontos a serem considerados no rosto detectado na imagem capturada.

Emoção	API	Questionário
Raiva	91,1%	90%
Desprezo	0%	0%
Desgosto	0%	0%
Medo	0%	0%
Felicidade	92%	91%
Neutro	91,6%	70%
Tristeza	80,8%	96,6%
Surpresa	0%	0%

Das 100 pessoas entrevistadas, 34 tiveram as emoções detectadas pela API compatíveis com suas respostas no questionário, sendo a maioria de emoção "Neutro". Também, pode ser notado que a emoção "Neutro" foi a mais detectada pela API quando

o usuário respondia no questionário outra emoção diferente da detectada pela API. Dos 47 registros de emoção "Neutro" que eram incompatíveis com os dados vindos do questionário, 13 das emoções respondidas pelos usuários eram "Raiva", tendo 79,2% de certeza por parte do entrevistado e 82,7% de precisão da API para emoção "Neutro".

7. Conclusão

Este trabalho tomou como base conceitos e técnicas aplicados em sistemas inteligentes, bem como as ideias apresentadas nos trabalhos correlatos. Desse modo, o desenvolvimento de um sistema para controle de satisfação, integrado com a *Emotion API* da *Microsoft*, se atenta ao desempenho da própria API, verificando sua precisão ao ser comparada com as respostas vindas dos próprios usuários. Também, é apresentada uma ferramenta importante para estudo e consulta voltados ao sentimento do público em determinado ambiente ou situação, possibilitando ainda a melhoria de atendimentos ou no desenvolvimento de diversos setores em uma empresa, por exemplo.

No que tange a precisão da API, só foi possível verificá-la em 34 das 100 pessoas entrevistas. Porém, do que foi detectado e respondido de acordo, foram obtidos percentuais próximos, o que mostra a precisão da API para tais emoções. A maior discrepância foi em relação ao sentimento "Neutro", como por exemplo, quando a API detectava como "Neutro" emoções que o usuário entrevistado afirmava ser "Raiva". Nestes momentos foi possível observar que quando a expressão facial não tem certos pontos do rosto destacados, a API detecta o sentimento como "Neutro". Esse fator foi predominante na pesquisa realizada e, portanto, mostrou imprecisão da API na maioria das emoções respondidas pelos entrevistados.

Desta forma, como contribuição, este trabalho oferece uma ferramenta que integra tecnologias, auxilia em diversos cenários de aplicação e verifica quanto a precisão a API da *Microsoft* para detecção de emoções. Como trabalhos futuros, é interessante pensar no desenvolvimento de um algoritmo que realiza a detecção de emoções utilizando técnicas de *Machine Learning*, tomando como base de aprendizado, dados que já foram previamente extraídos. Assim, seria possível o desenvolvimento e apresentação de uma ferramenta com maior precisão e independente de qualquer API para detecção de emoções.

Referências

Cainthus. (2016), "Our technology - Machine vision software". Acesso em abril de 2018. Disponível em: <https://www.cainthus.com/>

Fayyad, Usama, Piatetsky-Shapiro, Gregory, Smyth, Padhraic. (1996), "From Data Mining to Knowledge Discovery in Databases". Acesso em abril de 2018. Disponível em: https://www.aaai.org/ojs/index.php/aimagazine/article/viewFile/1230/1131>

Gomes, Fabio. (2013), "Introdução ao FDD - Feature Driven Development". Acesso em abril de 2018. Disponível em: <https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971>

Heptagon. (2017), "O que é FDD?". Acesso em abril de 2018. Disponível em: <http://heptagon.com.br/fdd/fdd-oque/>

Lee, Shih-Hsiung, Yang, Chu-Sing. (2017), “An intelligent home access control system using deep neural network”. Acesso em março de 2018. Disponível em: <http://ieeexplore.ieee.org/document/7991105/>

Mathworks. (2013), “Using pattern recognition for object detection, classification, and computer vision segmentation”. Acesso em abril de 2018. Disponível em: <https://www.mathworks.com/discovery/pattern-recognition.html>

Mathworks. (2016), “What is Machine Learning?”. Acesso em março de 2018. Disponível em: <https://www.mathworks.com/discovery/machine-learning.html>

Mathworks. (2017), “What is Deep Learning?”. Acesso em junho de 2018. Disponível em: <https://www.mathworks.com/discovery/deep-learning.html>

Microsoft. (2016), “API de Detecção de Emoções”. Acesso em abril de 2018. Disponível em: <https://azure.microsoft.com/pt-br/services/cognitive-services/emotion/>

NEC. (2013), “NeoFace Watch”. Acesso em março de 2018. Disponível em: <https://www.nec.com/en/global/solutions/safety/face_recognition/NeoFaceWatch.html?>

Pichiliani, Mauro. (2008), “Data Mining na Prática: Regras de Associação”. Acesso em abril de 2018. Disponível em: <https://imasters.com.br/artigo/7853/sql-server/data-mining-na-pratica-regras-de-associacao/?trace=1519021197&source=single>

Pizzolato, Rafael. (2015), “7 motivos para implantar tecnologia nas empresas”. Acesso em maio de 2018. Disponível em: <https://blog.starti.com.br/implantar-tecnologia-nas-empresas/>

Prass, Fernando Sarturi. (2012), “Um visão geral sobre as fases do Knowledge Discovery in Databases (KDD)”. Acesso em abril de 2018. Disponível em: <http://fp2.com.br/blog/index.php/2012/um-visao-geral-sobre-fases-kdd/>

Sá, Joaquim P. M. (2000), “Reconhecimento de Padrões”. Acesso em abril de 2018. Disponível em: <https://web.fe.up.pt/~jmsa/recpad/index.htm>

SAS. (2015), “Machine Learning – O que é e por que é importante?”. Acesso em março de 2018. Disponível em: <https://www.sas.com/pt_br/insights/analytics/machine-learning.html>

Soares, Antonio Carlos. (2014), “8 ferramentas de controle de performance”. Acesso em maio de 2018. Disponível em: <https://endeavor.org.br/ferramentas-controle-performance/>

System, Expert. (2017), “What is Machine Learning? A definition”. Acesso em março de 2018. Disponível em: <http://www.expertsystem.com/machine-learning-definition/>

Apêndice A

Neste apêndice, podem ser observados os seguintes requisitos do projeto:

RF.1 – Descrição: Detectar emoções

O sistema deve ser capaz de detectar emoções expressas em um rosto detectado em uma imagem, por meio da *Emotion API* da *Microsoft*.		
Relevância: Essencial	**Complexidade:** Médio	**Rastreabilidade:** N/A
RF.2 – Descrição: Apresentar questionário		
O sistema deve apresentar um questionário voltado para a identificação da emoção contendo perguntas relevantes para o ambiente de pesquisa.		
Relevância: Essencial	**Complexidade:** Médio	**Rastreabilidade:** N/A
RF.3 – Descrição: Apresentar painel de controle		
O sistema deve disponibilizar ao usuário um painel de controle (por meio de um sistema na *web*).		
Relevância: Essencial	**Complexidade:** Médio	**Rastreabilidade:** N/A
RF.4 – Descrição: Gerar relatório		
O sistema deve disponibilizar a opção de gerar relatório no painel de controle, contendo dados sobre as consultas de satisfação realizadas em um certo período.		
Relevância: Essencial	**Complexidade:** Médio	**Rastreabilidade:** N/A
RF.5 – Descrição: Apresentar gráfico		
O sistema deve apresentar um gráfico no painel de controle, contendo a variação dos cantos direito e esquerdo da boca.		
Relevância: Essencial	**Complexidade:** Médio	**Rastreabilidade:** N/A
RF.6 – Descrição: Armazenar dados		
O sistema deve ser capaz de armazenar a emoção do usuário, as respostas do questionário e a data em um banco de dados para futuras requisições.		
Relevância: Essencial	**Complexidade:** Médio	**Rastreabilidade:** N/A

Apêndice B

A seguir é apresentada a lista contendo as funcionalidades que o sistema contempla:

- Detecção de emoções por meio de imagem facial;
- Questionário sobre emoção do usuário;
- Painel de controle para usuário;
- Apresentação do gráfico de pontos relevantes da boca;
- Geração de relatório;
- Armazenamento em um banco de dados.

Apêndice C

Neste apêndice, pode ser observada a tabela representando a etapa de planejamento por funcionalidade da metodologia FDD:

Funcionalidade	Agosto				Setembro			
	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
Detecção de emoções (API)								
Questionário de satisfação								
Dashboard								
Gráfico de pontos relevantes								
Geração de relatório								
Armazenamento								
Funcionalidade	**Outubro**							
	Semana 1	**Semana 2**	**Semana 3**	**Semana 4**				
Detecção de emoções (API)								
Questionário de satisfação								
Dashboard								
Gráfico de pontos relevantes								
Geração de relatório								
Armazenamento								

Figura 12. Planejamento por funcionalidades representado por um cronograma.

Apêndice D

É apresentado o seguinte diagrama de atividade para a subseção Implementação:

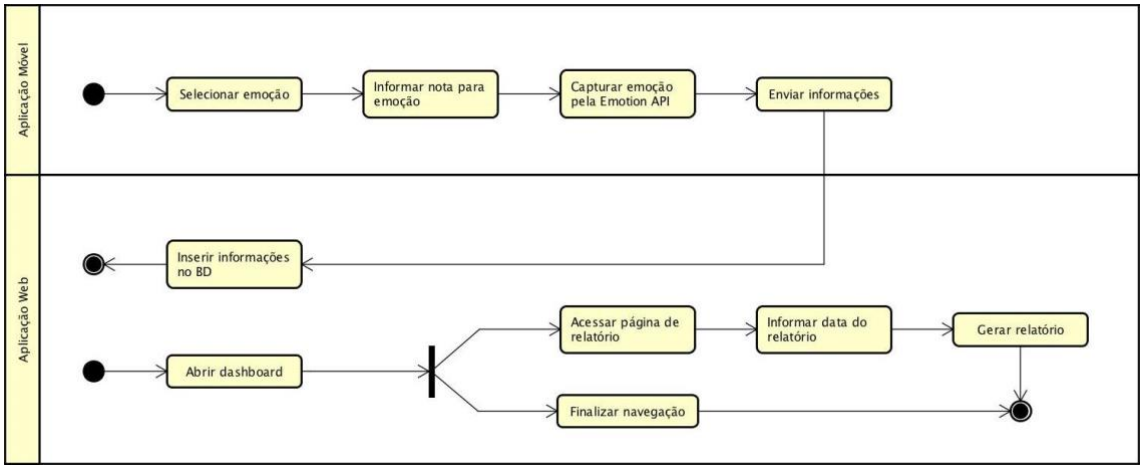


Figura 13. Diagrama de Atividade do projeto.

Apêndice E

A seguir, diagrama de classes dividida na aplicação móvel e aplicação *web*:

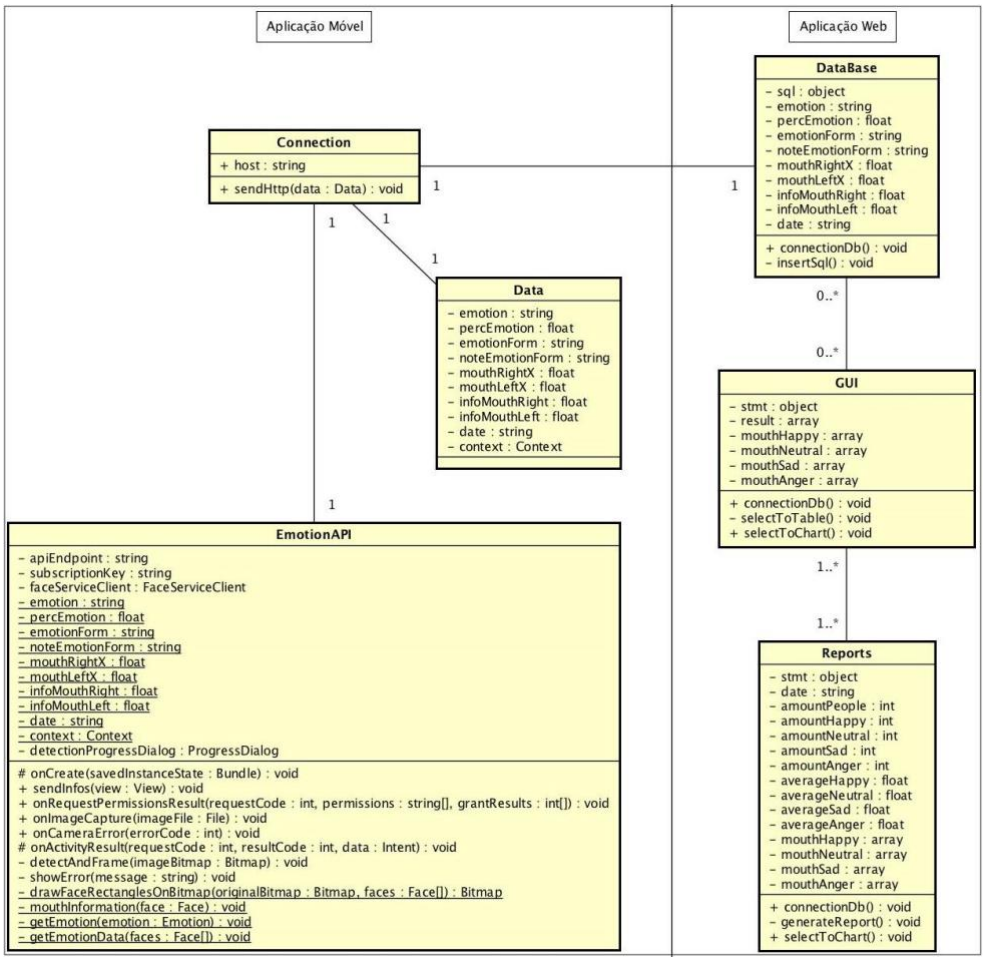


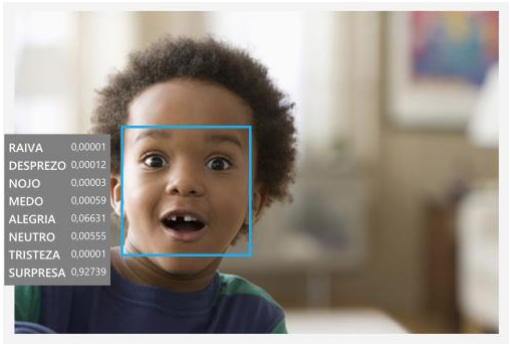
Figura 14. Diagrama de Classes do projeto.

Anexo A

Neste anexo, está presente a imagem contendo parte (a) e (b) do funcionamento da *Emotion API* da *Microsoft*:


```
JSON:
[
 {
   "faceRectangle": {
     "top": 141,
     "left": 130,
     "width": 162,
     "height": 162
   },
   "scores": {
     "anger": 9.29041E-06,
     "contempt": 0.000118981574,
     "disgust": 3.15619363E-05,
     "fear": 0.000589638,
     "happiness": 0.06630674,
     "neutral": 0.00555004273,
     "sadness": 7.44669524E-06,
     "surprise": 0.9273863
   }
 }
]
```


(a)



(b)

Figura 15. JSON (a) retornado do rosto (b).

Fonte: Microsoft, 2016

Anexo B

Aqui é apresentada uma imagem do funcionamento do trabalho correlato Cainthus:

Figura 16. Sistema *Cainthus* em funcionamento.

Fonte: Cainthus, 2018