**TREMSTATUS**

Bruno César Miazaki Andrade [[1]](#footnote-2) – Fatec Carapicuíba

Vinicius Baptista da Silva [[2]](#footnote-3) – Fatec Carapicuíba

Prof. Dr. Dewar Taylor Carnero Chavez [[3]](#footnote-4) – Fatec Carapicuíba

**RESUMO**

TREMSTATUS, é um aplicativo móvel que possibilita ao utilizador verificar um conjunto de mensagens, categorizadas por meio de mineração de textos, referente à linhas ferroviárias e identificar possíveis transtornos. Este artigo tem como objetivo demonstrar a utilização da Biblioteca Python NLTK para mineração de texto, dado um conjunto de mensagens retiradas do micro blog Twitter, das contas da CPTM, METRÔ e de outros perfis que disponibilizam atualizações sobre as linhas neste serviço, referente às linhas férreas da cidade de São Paulo e da região metropolitana do Estado de São Paulo. O aplicativo será desenvolvido em React Native, um framework desenvolvido com a linguagem de programação web JavaScript, que permitirá ao utilizador ver estas mensagens categorizadas com as seguintes polaridades: Negativo, Neutro e Positivo.

**Palavras-chave:** React Native, NLTK, Mineração de Texto, Twitter.

***ABSTRACT***

*TREMSTATUS is a mobile application that allows the user to check a set of messages, categorized by means of text mining, referring to the railway lines and to identify possible disorders. This article aims to demonstrate the use of the NLTK Python Library for text mining, given a set of messages taken from the micro blog Twitter, the accounts of CPTM, METRÔ and other profiles that provide updates on the lines in this service, referring to the lines railroads of the city of São Paulo and the metropolitan region of the State of São Paulo. The application will be developed in React Native, a framework developed with JavaScript web programming language, which will allow the user to see these messages categorized with the following polarities: Negative, Neutral and Positive.*

***Keywords:*** *React Native, NLTK, Text Mining, Twitter.*

# INTRODUÇÃO

O transporte público sobre linhas férreas auxilia a população a se locomover, mas como em qualquer tipo de transporte, público ou privado, podemos enfrentar problemas técnicos ou de qualquer outra natureza, como eventos naturais, manutenções pontuais ou emergenciais.

Os usuários da Companhia Paulista de Trens Metropolitanos (CPTM) e do Metrô de São Paulo enfrentaram um problema que levou à interrupção momentânea do serviço de transporte público a cada 23 horas e 34 minutos no ano de 2016. (GRAÇA; STOCHERO, 2017).

Logo se faz necessária uma comunicação eficaz entre as partes interessadas, prestadores desses serviços e seus clientes. Para este meio de transporte em questão temos os seguintes meios de comunicação: Portal do prestador de serviços e perfis em redes sociais. Porém esses meios de comunicação podem se tornar ineficazes quando o cliente não tem acesso as redes sociais muito menos ao portal, quando se tem acesso as redes sociais. é necessário acompanhar todos os perfis disponíveis falando sobre as linhas, pois as atualizações dos perfis oficiais podem demorar a serem publicadas. O micro blog Twitter permite a recuperação fácil das mensagens de todos os perfis, oficiais, não-oficiais sobre essas linhas ou até mesmo dos próprios usuários. Tornando viável a criação de um aplicativo que possa concentrar esses dados em apenas um lugar, facilitando a busca por informações.

O objetivo geral deste artigo é apresentar um novo aplicativo mobile criado inicialmente para a plataforma Android, que disponibiliza status sobre serviços ferroviários oferecidos pelas seguintes companhias: CPTM e Metrô e de outras linhas que são controladas por concessões, e analisar por meio do algoritmo Naive Bayes aplicado com NLTK para categorizar esses status. Para tanto, alguns objetivos específicos se fazem necessários: detalhar o processo de criação do aplicativo; demonstrar a recuperação dos status do Twitter; categorizar os status com o algoritmo de Naive Bayes; apresentar os resultados do algoritmo.

Atualmente muitos aplicativos criados buscando a resolução problemas cotidianos, os aplicativos que permitem a comunicação, como o Twitter, são os mais famosos entre os portadores de smartphones. Um novo aplicativo que permite ao usuário acesso às informações sobre os serviços que ele utiliza pode resolver o problema de comunicação entre a prestadora de serviço e o portador do aplicativo?

O Aplicativo foi criado utilizando o framework React Native, a recuperação dos status por meio do Twitter foi desenvolvido com linguagem de programação Python, a categorização desses status foi desenvolvida utilizando a mesma linguagem que a recuperação dos mesmos a fim de facilitar a integração entre o algoritmo de recuperação e o de categorização que utilizou da biblioteca NLTK e do algoritmo Naive Bayes.

O TremStatus foi criado para sanar o problema de comunicação entre as companhias prestadoras de serviços e seus clientes, ao abrir o aplicativo o usuário poderá identificar facilmente se a linha que ele utiliza está passando por problemas técnicos ou de qualquer outra natureza, e os status disponibilizados pelo aplicativo não serão apenas destas companhias como também dos próprios clientes, pois esses status serão coletados do Twitter e de qualquer perfil que estiver falando sobre essas linhas.

# FUNDAMENTAÇÃO TEÓRICA

## React Native

React Native, um projeto open source criado e mantido pelo Facebook e a comunidade de JavaScript, permite ao desenvolvedor criar aplicativos móveis usando apenas JavaScript. Ele usa o mesmo conceito que o React, permitindo compor uma rica interface móvel a partir de componentes.

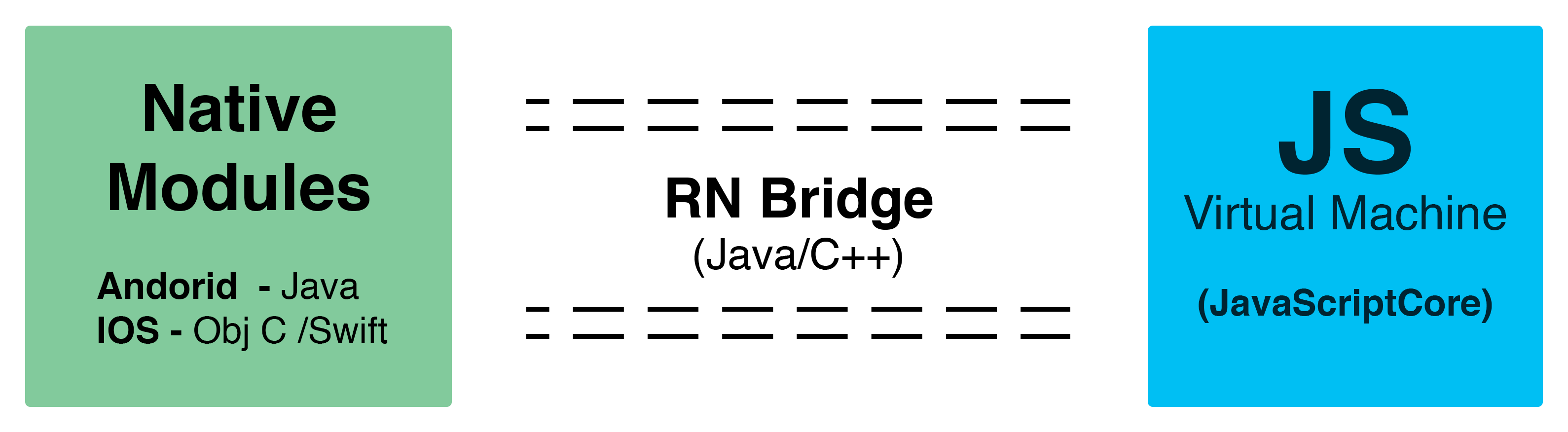
Se o desenvolvedor tiver necessidade de otimizar alguns aspectos do seu aplicativo móvel com códigos nativos, em Objective-C, Java, ou Swift, o React Native consegue combinar estas implementações com os componentes escritos em JavaScript.

Eisenman (2017) argumenta que React Native o framework JavaScript baseado em React, a biblioteca JavaScript criada pelo Facebook para construir interfaces, mas, em vez disso, o seu seguimento é focado na criação de aplicativos mobile. Desenvolvedores de aplicações web podem criar aplicativos semelhantes aos nativos, com todo o conforto desta biblioteca JavaScript. React Native torna a tarefa de escrever código, para ambos os sistemas mais conhecidos pelo mercado vigente, técnica também conhecida como cross-plataform, ser muito simples.

Um aplicativo escrito em React Native utiliza da seguinte arquitetura que o React Native disponibiliza, estes três pilares abaixo são fundamentais para que o canal entre o nativo e o JavaScript funcione corretamente:

1. **Código Nativo/Módulos:** Código escrito em Objective C ou Swift para sistemas iOS, e Java para sistemas Android.
2. **JavaScript VM:** Máquina Virtual JavaScript que roda todo o código JavaScript. Para realizar tal tarefa o React Native faz uso do JavaScriptCore, criado para iOS e que também é utilizado no Android.
3. **React Native Bridge(ponte)**: Escrita em C++/Java que é responsável pela comunicação entre os códigos nativo e JavaScript.

Figura 1 – Arquitetura React Native

Fonte: React Made Native Easy - <https://www.reactnative.guide/assets/images/rn-architecture.png>

Utilizando esta arquitetura é possível criar os aplicativos chamados de cross-plataform, o desenvolvedor não precisa ser especialista em linguagens como Java e Objective-C, ele apenas precisa escrever um código em JavaScript e o React Native cuida do resto para tornar seu aplicativo nativo como se tivesse sido escrito em uma dessas linguagens citadas.

## Natural Language Toolkit - NLTK

NLTK é uma biblioteca escrita em Python que disponibiliza recursos importantes para a criação de programas de Processamento de Linguagem Natural.

Por linguagem natural entendemos a linguagem usada na comunicação entre os humanos, idiomas como o inglês, o espanhol ou o português. Em alguns momentos. O Processamento de Linguagem Natural pode até ser considerado como algo simples quanto a contagem da frequência de palavras, em questão a comparar múltiplos estilos de escrita. Por outro lado, a PLN envolve "assimilar" enunciados humanos completos, no mínimo até o momento de ser capaz de proporcionar soluções úteis a estes. (BIRD; KLEIN; LOPER, 2014)

Esta biblioteca disponibiliza classes básicas para representar dados relevantes para a PLN e interfaces padrões para executar tarefas como a classificação de textos e implementações mais complexas que combinadas tornam possível resolver problemas mais complexos.

As seguintes classes serão usadas para desenvolver o algoritmo que classifique as palavras:

1. **Corpus**: O método “corpus.stopwords.words” é utiliza porquê retorna um conjunto de palavras, em português, que para executar a tarefa de análise de texto se tornam dispensáveis. Exemplo de stopwords: dos, como, ou, há, nos. Este método retorna mais de 200 palavras consideradas stopwords.
2. **NaiveBayesClassifier**: É o algoritmo utilizado para classificações dos textos, desse algoritmo iremos utilizar duas funções, o treinamento necessário para modelar a nossa classificação e a função de classificação utilizada para retornar o texto que passou pelo algoritmo e foi classificado com algumas das polaridades definidas.

### Mineração de Texto

De acordo com (SERPIÃO PRB. et al., 2010, p.2), é apresentada seguinte definição para mineração de texto:

Estudos apontam que mineração de texto é uma técnica adequada para manipulação automática de grandes volumes de dados, pertencente ao campo da ciência da computação, cientificamente ligada ao desenvolvimento de ferramentas de recuperação automática da informação. O método básico consiste em explorar e identificar termos relevantes em um grupo textual ou documental, bem como estabelecer padrões textuais e desenvolver grupos temáticos de assuntos pela frequência de aparecimento de termos no domínio a ser analisado. (SERPIÃO PRB. et al., 2010, p.2).

A mineração de texto é capaz de resolver um problema causado graças ao desenvolvimento da internet, pois o principal meio de armazenamento de informações é realizado em documentos de texto. Com o uso da mineração agora uma máquina é capaz de interpretar a linguagem natural, o processo informatizado de recuperação de informação com mineração envolve várias matérias, aprendizagem de máquina, mineração de dados, recuperação de informação, estatística e linguagem computacional para transformar a linguagem escrita em algo que o computador consiga entender. (MACHADO. et al., 2010)

Aranha e Passos (2006) argumentam que a mineração de texto não serve de mecanismo para busca, a mineração é utilizada para fazer novas descobertas, enquanto a busca o usuário já sabe do que precisa. Além disso, a mineração se difere da tecnologia de chatbots, pois não é utilizada para simular o comportamento humano.

### Twitter

Segundo Oliveira e Bermejo (2017 *apud* MONTOYO; MARTÍNEZ-BARCO; BALAHUR, 2012, p. 675-679), “o grande volume de dados postados na internet, por meio das mídias sociais, está produzindo mudanças importantes na maneira pela qual as pessoas se comunicam e compartilham conhecimentos e emoções que influenciam o comportamento social, político e econômico em todo o mundo.”

O Twitter será utilizado nesse projeto por ser a rede social mais utilizada pelos prestadores de serviços, CPTM, METRÔ. Os perfis dessas empresas na rede social frequentemente publicam comunicados sobre o estado das linhas, ou outras informações. Outros perfis existentes, de pessoas comuns usuárias de linhas ferroviárias, nessa rede social também publicam sobre o serviço que utilizam diariamente, por muitas vezes informando problemas antes mesmo que as próprias empresas consigam avaliar e confirmar que um problema está ocorrendo. As mensagens utilizadas para alimentar o aplicativo surgiram de conexões realizadas com apis disponibilizadas pela rede social ocupando nossa base de dados para manter o aplicativo sempre atualizado e aumentando a acurácia do algoritmo de análise de texto.

# Procedimentos metodológicos

Trata-se de uma pesquisa exploratória e descritiva. Foi utilizado o método de abordagem indutivo, que parte de um caso único, com uma conexão ascendente, do específico para o geral (LAKATOS; MARCONI, 2009). Trata-se ainda, de uma pesquisa experimental, por meio de um estudo de caso junto à uma rede social, Twitter. E um experimento com o processamento linguagem natural, a partir do uso de dados coletados do Twitter referente a serviços prestados por CPTM e Metrô, após observações dos principais pontos de insatisfação de usuários a questão de disponibilização de informação.

# DESENVOLVIMENTO

* 1. **Coleta dos Dados**

Figura 2 – Fluxo de coleta de mensagens

Fonte: Elaborado pelos autores

A Figura 2 está representando o fluxo necessário para receber as mensagens que são postadas no Twitter, a aplicação de coleta foi criada sobre a linguagem Python, utilizando uma biblioteca chamada Tweepy.

Uma conexão é aberta entre a aplicação e o Twitter, parâmetros são enviados para que a aplicação fique “escutando” a api do Twitter que responde mensagens somente dentro dos filtros aplicados. Exemplo de filtros: Perfil, hashtags, palavras existentes dentro das mensagens.

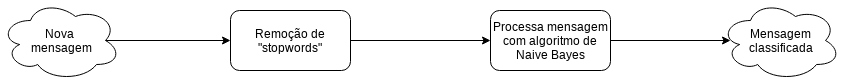
Figura 3 – Fluxo para limpar mensagens

Fonte: Elaborado pelos autores

O fluxo para limpar as mensagens que chegam da api do Twitter, representado na figura 3, é responsável por remover tudo o que não interessa para o próximo passo: Análise de dados e Categorização das frases. É imprescindível retirar as hashtags, remover as urls e tirar marcas de agregação dos perfis “@perfil” este fluxo é executado dentro da aplicação feita para coletar as mensagens, após o Tweepy retornar uma nova mensagem o fluxo descrito acima é iniciado.

* 1. **Mineração de Texto**

Figura 4: Fluxo do processamento de uma nova mensagem

Fonte: Elaborado pelos autores

Após passar por todo o fluxo de coleta dos dados, outra parte do processo é iniciada, a mineração do texto, que ao fim retorna a mensagem classificada com uma das emoções: Negativo, Neutro ou Positivo.

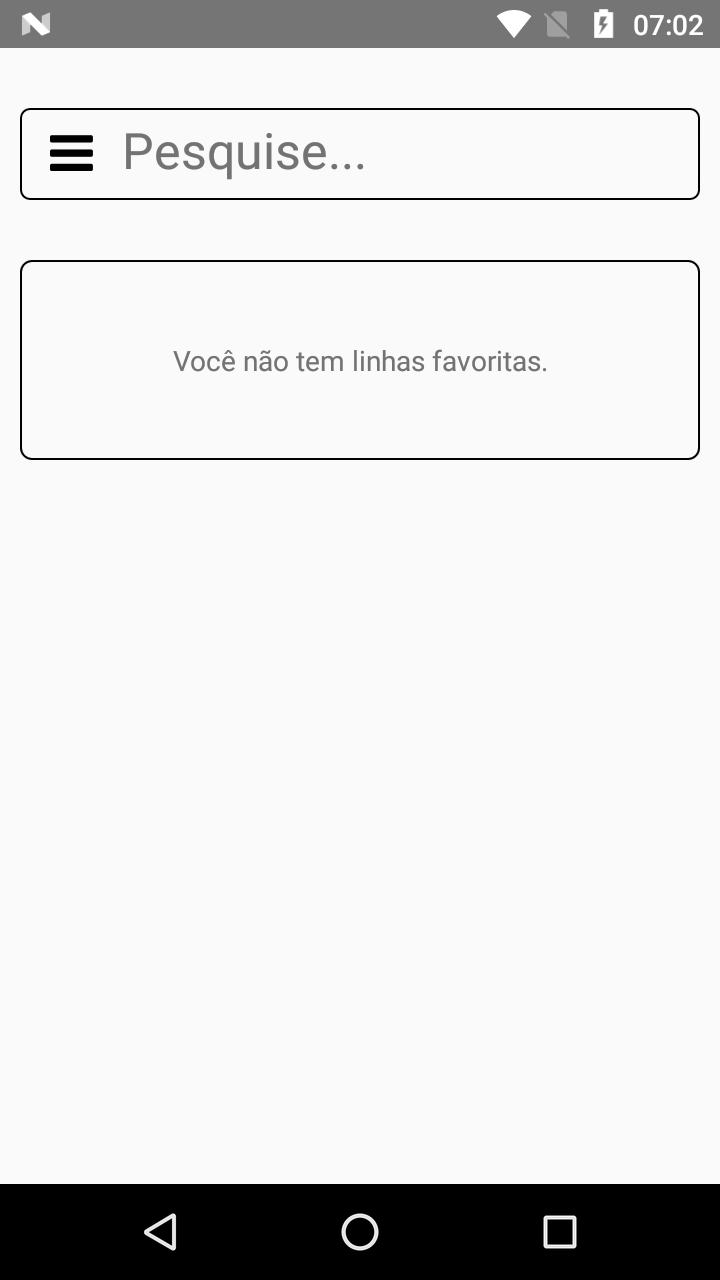
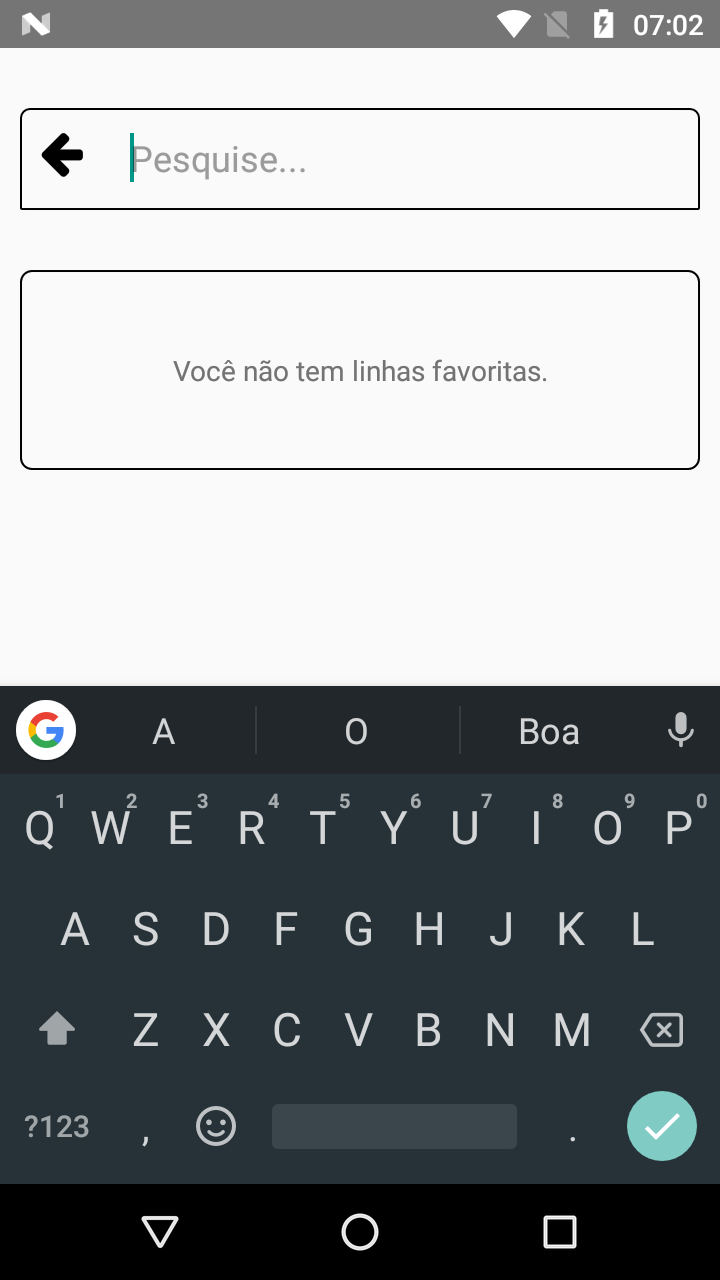
O primeiro passo logo após a mensagem entrar no novo fluxo é a remoção de “stopwords” palavras desnecessárias para a mineração. A seguir entra em ação o algoritmo de Naive Bayes, analisando a frase a partir do conhecimento já adquirido em processos anteriores e ao terminar o processo ele retorna a mensagem classificada.

* 1. **Aplicativo**

O aplicativo móvel TremStatus foi desenvolvido com o framework JavaScript React Native, que possibilita ao desenvolvedor criar aplicativos móveis para as plataformas Android e iOS. No momento ele funcionará apenas no Android, pois é inviável instalar em um dispositivo móvel iOS sendo indispensável o uso de um Mac OS para executar tal tarefa.

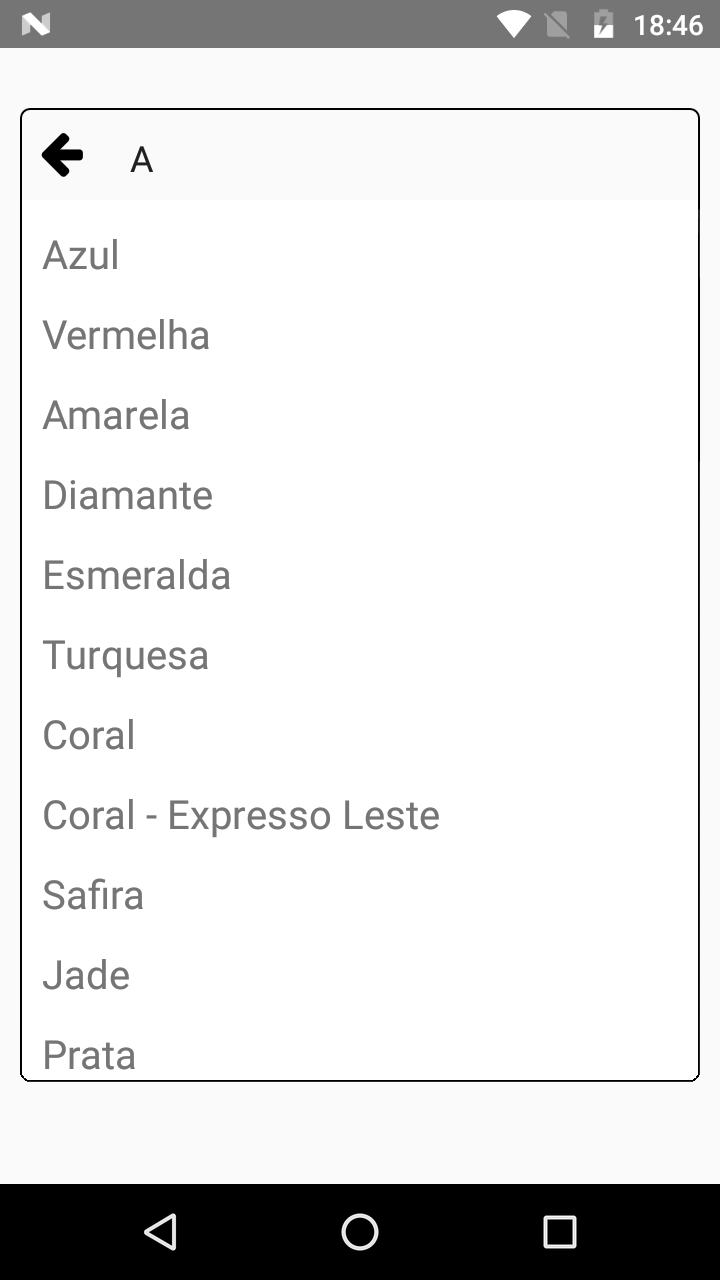
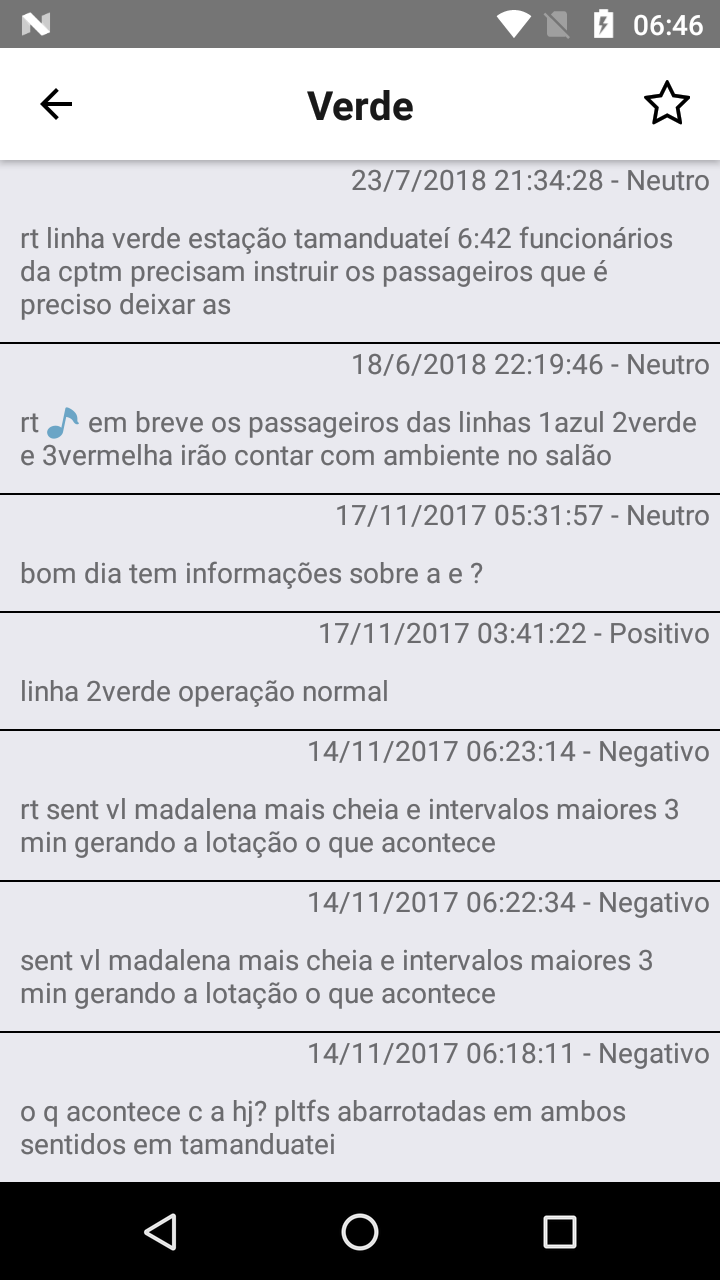
A seguir as telas disponíveis no aplicativo:

Figura 5: Home Figura 6: Pesquisa

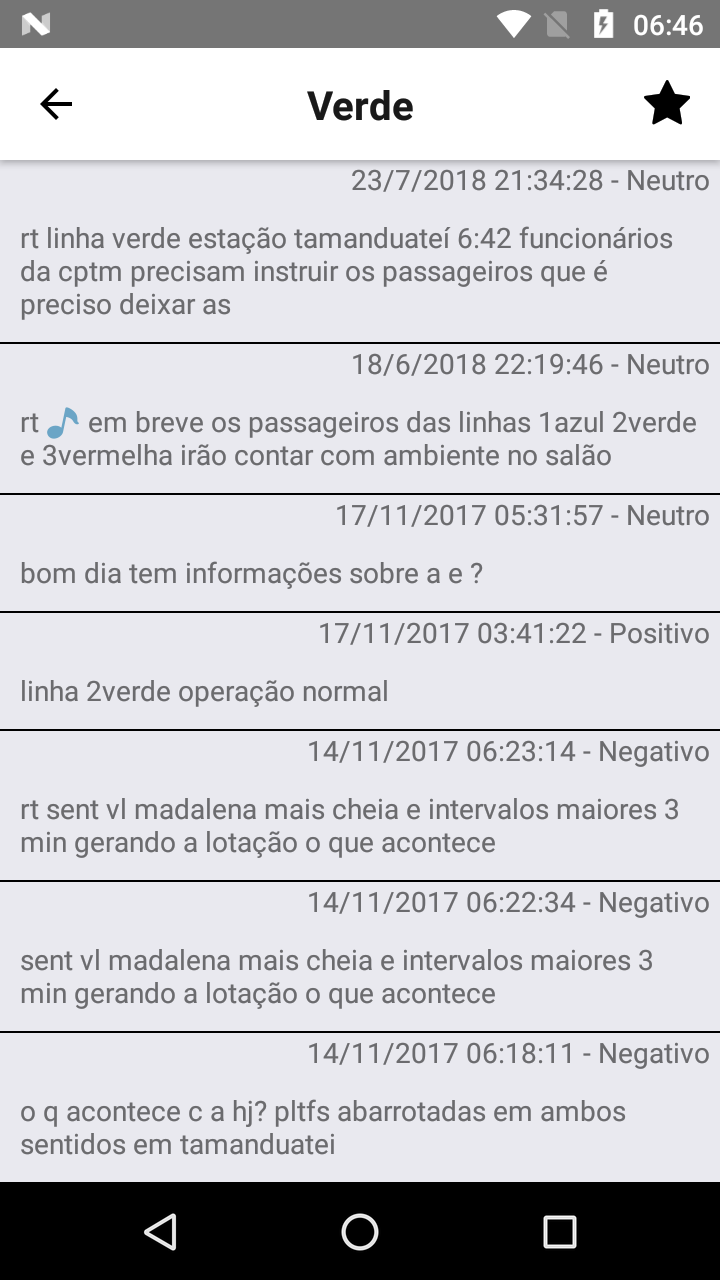
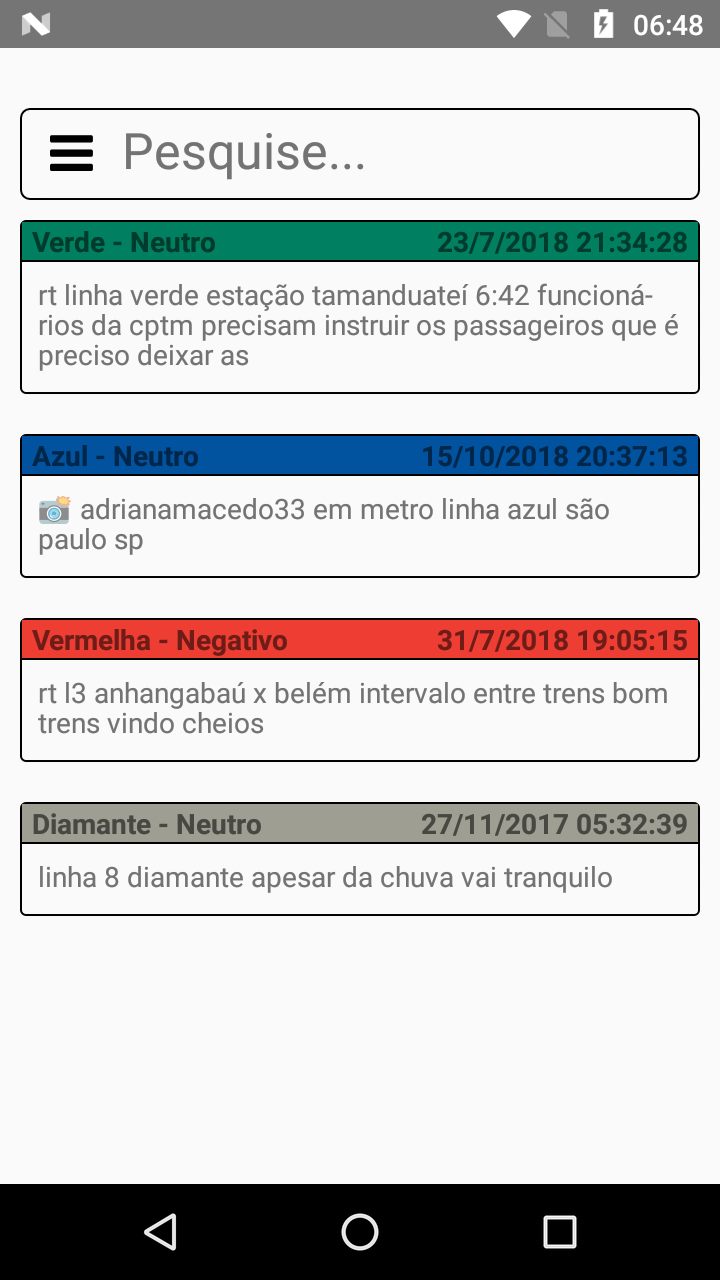
Fonte: Elaborado pelos autores Fonte: Elaborado pelos autores

Figura 7: Filtro da busca Figura 8: Linha

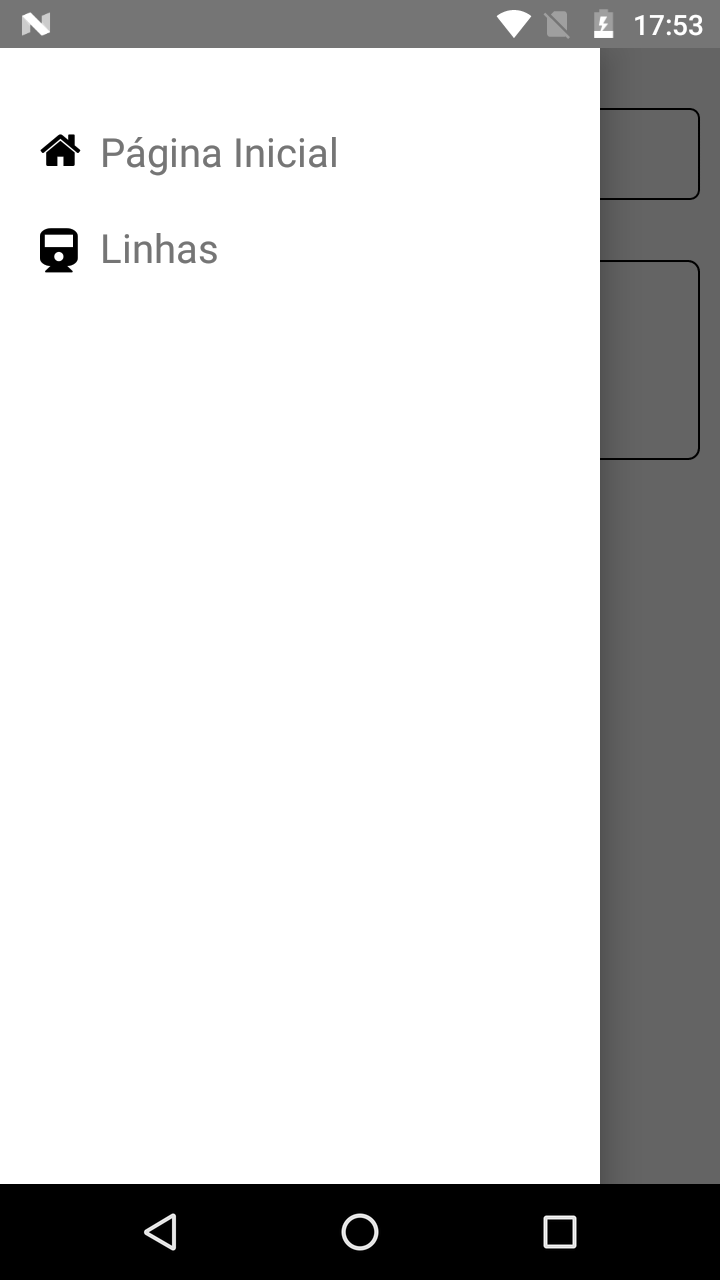
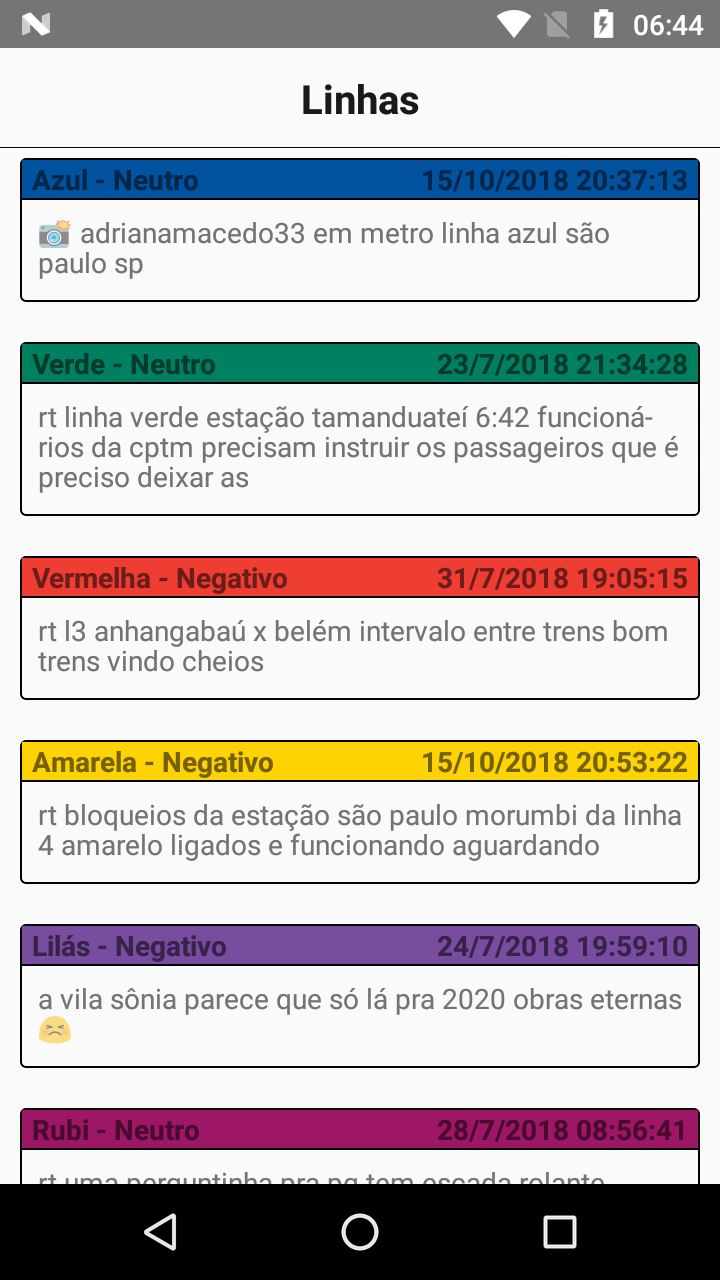
Fonte: Elaborado pelos autores Fonte: Elaborado pelos autores

Figura 9: Linha favorita Figura 10: Linhas favoritas

Fonte: Elaborado pelos autores Fonte: Elaborado pelos autores

Figura 11: Menu Figura 12: Linhas

Fonte: Elaborado pelos autores Fonte: Elaborado pelos autores

À tela inicial do sistema, Figura 5, o portador pode ver suas linhas favoritas, Figura 10, e também pode buscar as linhas, Figura 6 e Figura 7. Ao clicar em uma linha o portador verá a tela com mensagens sobre ela, Figura 7, nesta mesma tela é possível marcar a linha como favorita, evitando refazer essa pesquisa ao entrar no aplicativo novamente. Existe um menu lateral no aplicativo, que abre ao clicar no botão dentro do box de pesquisa, Figura 6, abrindo o menu, Figura 11, clicando na opção “Linhas” o portador é redirecionado a uma nova tela, Figura 12, contento as últimas mensagens sobre todas as linhas disponíveis no aplicativo.

# RESULTADOS E DISCUSSÃO

Após uma análise dos meios de comunicação atualmente utilizados pelas empresas prestadores do serviço abordado nesse artigo, transporte sobre trilhos, foi verificado a necessidade de melhorias nestes meios de comunicação. Desta forma, visando melhorar a comunicação entre as partes o aplicativo móvel TremStatus foi desenvolvido possibilitando que o usuário desses serviços consiga acesso às informações de forma mais prática. O TremStatus não utiliza apenas informações dos perfis das empresas, ele também contém mensagens de qualquer usuário da rede social, assim possibilitando maior fluxo de mensagens alimentando o aplicativo e trazendo notícias atualizadas sobre: Trens, Metrôs ou Mono trilhos

# CONSIDERAÇÕES FINAIS

Conclui-se que é de suma importância que haja uma melhora na comunicação entre cliente e prestador de serviço, pois no cenário atual os clientes tendem a ter dificuldade para encontrar informações atualizadas sobre o estado do transporte que utiliza.

O diferencial da proposta deste projeto, com relação ao problema de comunicação, está no agrupamento das informações, sobre todas as linhas disponíveis e de qualquer usuário que fale sobre a mesma.

**REFERÊNCIAS**

FACEBOOK. ***Build native mobile apps using JavaScript and React*.** Disponível em: <https://facebook.github.io/react-native/>. Acesso em: 16 nov. 2018.

LOPER, Edward; KLEIN, Ewan; BIRD, Steven. **Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit**. Disponível em: <http://www.nltk.org/book/>. Acesso em 19 Nov. 2018.

SERAPIAO, Paulo Roberto Barbosa; SUZUKI, Kátia Mitiko Firmino; MARQUES, Paulo Mazzoncini de Azevedo. **Uso de mineração de texto como ferramenta de avaliação da qualidade informacional em laudos eletrônicos de mamografia.** Radiol Bras, São Paulo , v. 43, n. 2, p. 103-107, Abr. 2010. Disponível em: <http://www.scielo.br/scielo.php?script=sci\_arttext&pid=S0100-39842010000200010&lng=en&nrm=iso>. Acesso em 19 Nov. 2018.

ARANHA, Christian; PASSOS, Emmanuel. **A Tecnologia de Mineração de Textos.** RESI: Revista Eletrônica de Sistemas de Informação, Vol. 5, No 2 (2006). Disponível em: <http://www.periodicosibepes.org.br/index.php/reinfo/article/view/171>. Acesso em 19 Nov. 2018.

MACHADO, Aydano P; FERREIRA, Rafael; BITTENCOURT, Ig Ibert; ELIAS, Endhe; BRITO, Patrick; COSTA, Evandro. **Mineração de texto em Redes Sociais aplicada à Educação a Distância.** Colabor@ - A Revista Digital da CVA-RICESU, Vol. 6, No 23 (2010). Disponível em: <http://pead.ucpel.tche.br/revistas/index.php/colabora/article/view/132>. Acesso em 19 Nov. 2018.

OLIVEIRA, Daniel José Silva; BERMEJO, Paulo Henrique de Souza. **Mídias sociais e administração pública: análise do sentimento social perante a atuação do Governo Federal brasileiro**. **Organ. Soc.**, Salvador , v. 24, n. 82, p. 491-508, set. 2017. Disponível em <http://www.scielo.br/scielo.php?script=sci\_arttext&pid=S1984-92302017000300491&lng=pt&nrm=iso>. acessos em 19 nov. 2018.

G1. ***Metrô e CPTM têm um problema operacional a cada 23 horas em 2016.*** Disponível em: <https://g1.globo.com/sao-paulo/noticia/metro-e-cptm-tem-um-problema-operacional-a-cada-23-horas-em-2016.ghtml>. Acesso em: 20 nov. 2018

**Learning React Native: Building Native Mobile Apps with JavaScript**. 2 ed. O'Reilly Media, Inc., 2017.

LAKATOS, E. M.; MARCONI, M. de A**. Fundamentos de metodologia científica**. 6. ed. São Paulo: Atlas, 2009. 315p.

1. - Aluno do CST em Análise e Desenvolvimento de Sistemas – *e-mail*: [bruno\_miazaki@outlook.com](mailto:bruno_miazaki@outlook.com) [↑](#footnote-ref-2)
2. - Aluno do CST em Análise e Desenvolvimento de Sistemas – *e-mail*: [vbaptista7@gmail.com](mailto:vbaptista7@gmail.com) [↑](#footnote-ref-3)
3. - Doutor em Engenharia Naval e Oceânica – *e-mail*: [dewar.chavez@fatec.sp.gov.br](mailto:dewar.chavez@fatec.sp.gov.br) [↑](#footnote-ref-4)