UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

CRISTIANO SALLA LUNARDI

Otimizando o processo de *b*rainstorming com técnicas de Processamento de Linguagem Natural e Aprendizado de Máquina

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Leandro Krug Wives
Coorientadora: MSc. Brenda Salenave Santana

Porto Alegre
2021

“Criatividade é a arte de conectar ideias.“

— STEVE JOBS

AGRADECIMENTOS

Primeiramente gostaria de agradecer aos meus pais Maria Lorena e Mauro pelo apoio na minha trajetória ao longo do curso.

Sou grato ao meu orientador Prof. Dr. Leandro Krug Wives por aceitar conduzir minha pesquisa e por me direcionar corretamente.

Grato à minha coorientadora Msc. Brenda Salenave Santana por ajudar a conduzir minha pesquisa e sanar minhas dúvidas.

Grato à todos os meus professores do curso de Engenharia da Computação da Universidade Federal do Rio Grande do Sul pela excelência acadêmica.

Agradeço também aos meus instrutores e colegas do curso *Apple Developer Academy* onde tive a ideia do tema desse trabalho.

RESUMO

Brainstorming, ou tempestade de ideias é uma atividade desenvolvida para explorar a potencialidade criativa de um indivíduo ou de um grupo. Porém, esse processo pode ser muito demorado devido ao agrupamento das ideias, que pode levar horas para ser concluído. Esse agrupamento de ideias, também conhecido como clusterização, é necessário para descobrir quais os principais temas presentes e em um segundo momento, definir qual ou quais desses temas serão relevantes para o grupo, descartando os outros. Atualmente, é muito comum que pessoas em organizações empresariais e escolas utilizem notas adesivas para realizar *brainstorming*. O objetivo deste trabalho é, através da aplicação de métodos de processamento de linguagem natural e aprendizado de máquina, facilitar e reduzir o tempo para concluir esse processo. Através do desenvolvimento de uma aplicação iOS com agrupamento de sentenças por *KMeans*, foi possível realizar um *brainstorming* mais eficiente para os usuários de teste.

Palavras-chave: Brainstorming. Processamento de Linguagem Natural. Desenvolvimento iOS. Python. Django. Swift.

Optimizing the brainstorming process with Natural Language Processing techniques and Machine Learning

ABSTRACT

Brainstorming is an activity designed to explore the creative potential of an individual or group. However, this process can be very time-consuming, taking hours to complete a grouping of these ideas. This grouping of ideas, also known as clustering, is necessary to discover the main themes present and, in a second moment, to define which of these themes will be relevant to the group, discarding the others. It is now widespread for people in business organizations and schools to use sticky notes to perform brainstorming. Through the application of natural language processing and machine learning methods, the objective of this work is to facilitate and reduce the time to complete this process. By developing an iOS application with grouping sentences by KMeans, it was possible to perform more efficient brainstorming for test users.

Keywords: Brainstorming, Natural Language Processing, iOS Development, Python, Django, Swift.

LISTA DE ABREVIATURAS E SIGLAS

PLN	Processamento de Linguagem Natural
EM	*Expectation Maximization*
SUS	*System Usability Scale*
MVP	Produto Viável Mínimo
MTV	*Model-Template-View*
DRY	*Don't Repeat Yourself*
CBOW	*Continuous Bag Of Words*

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente, empresas de tecnologia e escolas realizam *brainstormings* de maneira mecânica, escrevendo suas ideias em notas adesivas e agrupando essas ideias manualmente pelos participantes, nota por nota, através da análise feita pelos mesmos. Em equipes muito grandes, esse processo de agrupamento pode levar horas, pois cada frase deve ser analisada e discutida para entender o seu contexto e então ser inserida em um dos grupos. No caso de empresas, esse tempo pode ter um custo alto visto que demanda atenção de um time inteiro durante horas, tempo que poderia ser melhor utilizado se essa etapa pudesse ser otimizada ou automatizada; já em escolas, pode consumir um longo período de tempo de uma aula.

Diante disso, este trabalho tem como objetivo desenvolver uma aplicação que permita a execução de processos de *brainstorming* e que agrupe automaticamente as frases dos participantes. Para tanto, serão aplicadas técnicas de Processamento de Linguagem Natural em conjunto com o algoritmo de agrupamento *KMeans* com a biblioteca de *word embedding Word2Vec*.

O trabalho apresenta-se dividido da seguinte forma: no Capítulo 2 são descritos os fundamentos necessários para compreensão do presente trabalho, e em sequência é descrito o estado da arte do tema abordado. Já no Capítulo 3 o foco está em apresentar trabalhos e ferramentas correlatas. O Capítulo 4 apresenta a proposta de desenvolvimento elaborada. O Capítulo 5 expõe os resultados alcançados diante dos cenários de testes utilizados para os experimentos de validação da proposição exposta. Por fim, o Capítulo 6 apresenta as considerações finais do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os conceitos e fundamentos teóricos necessários para a compreensão e o desenvolvimento deste trabalho.

2.1 *Brainstorming*

O *brainstorming*, é um modelo de dinâmica de grupo para explorar o potencial criativo dos envolvidos. A proposta é que o grupo se reúna e explore a diversidade de pensamentos e experiências para gerar soluções inovadoras, sugerindo pensamentos e ideias sobre um determinado tema. Utilizando essa técnica, espera-se coletar o maior número possível de ideias, propostas, visões e possibilidades que levem a uma solução eficaz para solucionar problemas (SMITH, 2020). O processo é dividido em três etapas principais: (1) encontrar os fatos, (2) gerar a ideia, e (3) encontrar a solução.

Na etapa 1 o problema é definido o problema e os participantes podem escolher qualquer tema. Após, segue a etapa 2, onde são coletadas todas as informações pertinentes ao problema. É a parte da geração de ideias. E na última etapa, busca-se a solução para o problema de foco a partir da seleção das melhores ideias propostas.

2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural, ou PLN, é a área que estuda problemas da geração e compreensão automática das línguas humanas naturais. Esta área de pesquisa aborda três aspectos da comunicação (PEREIRA, 2020):

1. Som: fonologia;
2. Estrutura: morfologia e sintaxe;
3. Significado: semântica e pragmática;

Para soluções que utilizam apenas textos, são geralmente usados os aspectos de morfologia (para reconhecer unidades primitivas e radicais de palavras), sintaxe (a fim de entender o sentido de cada frase e como as palavras se relacionam), semântica (associando significado às frases) e pragmática (verificação se o significado inferido está de acordo com o mais apropriado no contexto). Aspectos de fonologia não são utilizados

para conteúdos textuais.

2.2.1 Pré-processamento dos textos

Em soluções que utilizam PLN, é fundamental utilizar o *pipeline* de pré-processamento no conteúdo que será analisado para conseguir os melhores resultados na classificação ou no agrupamento dos textos.

O pré-processamento é a etapa em que os dados são tratados, nesse caso, onde será eliminado palavras que não agregam sentido a frase ou que não contribuem para a analise mais profunda do contexto.

O *pipeline* idealizado para esta aplicação consiste em:

1. Determinar Linguagem;
2. Separar *tokens*;
3. Lematizar;
4. *Semantic Role Labeling*;
5. Remover palavras vazias;

Esse *pipeline* é feito através da biblioteca *NaturalLanguage* [1] disponível para desenvolvimento em Swift.

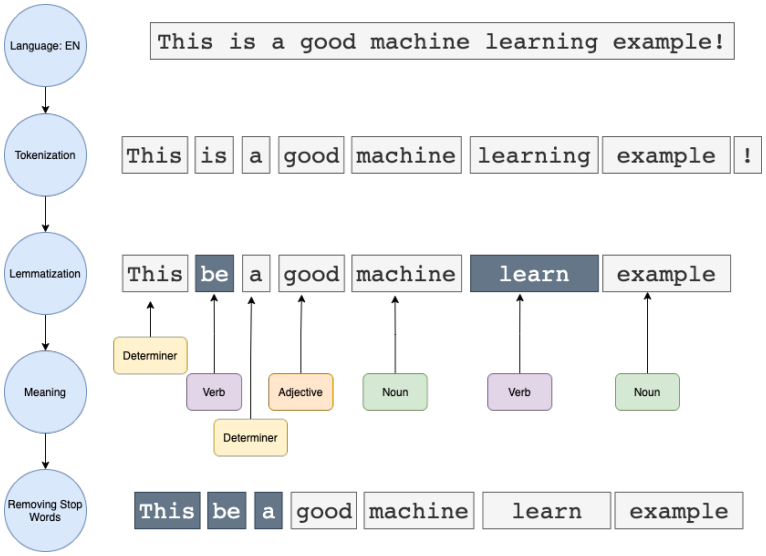
Na Figura 2.1 mostra as etapas do pré-processamento. A primeira etapa serve apenas para identificar qual o idioma da frase. Isso é muito importante para línguas que possuem a mesma palavra porém podem ter significados diferentes. A segunda etapa é onde separa-se a frase em vários *tokens* contendo palavras ou pontuações. *Tokens*, também conhecido como componente léxico é uma cadeia de caracteres que tem um significado coerente. No caso de Processamento de Linguagem Natural, *tokens* são formados fatiando o texto no menor fragmento possível, que são as palavras. Na separação dos *tokens* pode-se optar por não criar *tokens* a partir de pontuações. Então com isso, conseguimos lematizar as palavras, ou seja, capturar a forma base da palavra. Como exemplo, podemos remover a conjugação verbal. *Semantic Role Labeling* refere-se a parte de classificação do tipo gramatical do elemento em questão. Como exemplo, classificando pronomes, verbos, adjetivos, números, conjunções e outros.

Por fim, é feita uma seleção de palavras que serão removidas da frase. Essas palavras têm a característica em comum de não agregar nenhum valor à frase, e podem

[1] Documentação do framework NaturalLanguage(SAHIN, 2020)

ser artigos definidos, preposições, advérbios e outros.

Figura 2.1: Etapas do pré-processamento.



Fonte: Autor.

2.2.2 *Word Embeddings*

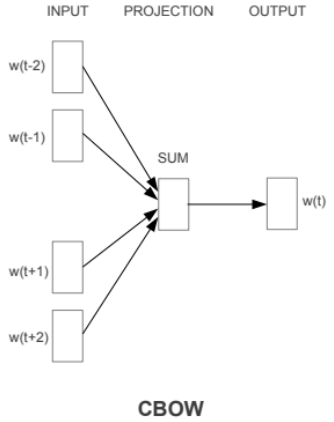
Em Processamento de Linguagem Natural, *Word Embedding* é um método de mineração de textos onde podemos representar palavras em forma de vetores. Diferenciando-se de outras abordagens, como *Bag of Words*, que representa vetores grandes e esparsos, o método de *Word Embeddings* utiliza vetores densos de tamanhos fixos que podem armazenar informações relacionadas ao contexto e significado das frases. As palavras são representadas por pontos em um espaço multidimensional chamado de *embedding space* (MOURA, 2018). A ideia é definir o significado da palavra por suas propriedades distribucionais, ou seja, em quais contextos eles ocorrem. Palavras que ocorrem em contextos semelhantes deve ter representações semelhantes. Em modelos de espaço vetorial, a similaridade pode ser medida por uma função de distância, por exemplo, a distância cosseno do ângulo entre os vetores (KOEHN, 2017).

2.2.2.1 Word2Vec

Word2Vec é um método estatístico para aprender de forma eficiente um *Word Embedding* a partir do corpus de um texto, e foi desenvolvido pela Google em 2013, tornando o treinamento de *embeddings* baseados em redes neurais mais eficientes (TOMALOK, 2019). O *Word2Vec* utiliza essencialmente dois modelos de redes neurais, os que pos-

suem camada de entrada, uma camada oculta e uma camada de saída. O primeiro modelo é o *Continuous Bag Of Words* (i.e., *CBOW*) que é utilizado para descobrir a palavra central de uma frase baseado no conjunto de palavras da frase. *CBOW* tem como objetivo prever a palavra central de acordo com um determinado texto. A camada de entrada recebe os dados pré-processados e na saída terá apenas uma palavra que será a com maior probabilidade de ser a palavra alvo dessa sentença. O *CBOW* está representado na Figura 2.2.

Figura 2.2: Representação das camadas do *CBOW*.



Fonte: Mikolov et al. (2013)

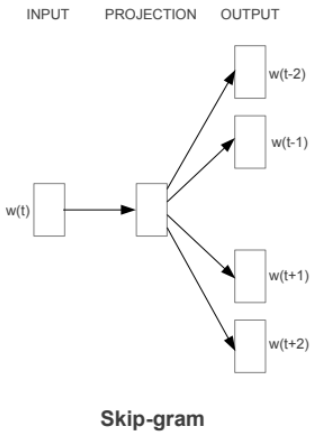
O segundo modelo é um *Skip-Gram* da Figura 2.3 que faz o caminho inverso do *CBOW*. *Skip-Gram* é uma técnica de aprendizado não supervisionado utilizada para achar a palavra mais importante em um dado contexto. No caso, através de uma palavra-alvo, tenta-se descobrir quais palavras estão no contexto. Ou seja, em sua camada de entrada, o modelo recebe apenas a palavra-alvo e, em sua saída, são retornadas palavras de possíveis contextos, de acordo com a palavra alvo.

O método *Word2Vec* é a etapa essencial para definir quais frases possuem semelhanças entre si. Esse é o primeiro passo para conseguirmos atingir o agrupamento desejável.

2.2.3 Aprendizado de Máquina

O aprendizado de máquina, é um método de análise de dados que constrói modelos analíticos de forma automatizada. Faz parte da grande área de inteligência artificial que baseia-se na ideia que sistemas podem aprender com dados, identificando padrões e tomando decisões sem interferência humana (STANGE, 2011).

Figura 2.3: Representação das camadas do *Skip-Gram*.



Fonte: Mikolov et al. (2013)

2.2.3.1 Aprendizado supervisionado

Dentre as técnicas de aprendizado de máquina, existem os algoritmos supervisionados que são usados quando deseja-se, a partir de um conjunto de dados rotulados, encontrar uma função capaz de predizer os rótulos desconhecidos. Com essa aprendizagem, pode-se tomar decisões precisas ao receber novos dados não rotulados a partir de um treinamento com dados que possuem rótulos conhecidos (BARROS, 2016).

2.2.3.2 Aprendizado não supervisionado

Há também os algoritmos de aprendizado não supervisionados. Nessa classe, os dados não possuem um rótulo, os algoritmos apenas agrupam os dados de acordo com alguma medida utilizada. Esses algoritmos são chamados de algoritmos de clusterização ou simplesmente *clustering*. A ideia aqui é descobrir uma similaridade e anomalias entre os dados (BARROS, 2016).

2.2.3.3 Clustering

Na literatura, pode-se obter diferentes definições do que é um *cluster*:

"Um cluster é um conjunto de entidades que são semelhantes e entidades de diferentes clusters não são semelhantes."

"Um cluster é um agregado de pontos no espaço de testes em que a distância entre dois pontos do mesmo cluster é menor que a distância de dois pontos em clusters diferentes."

"Clusters podem ser classificados como regiões contínuas no espaço com d-dimensões contendo alta densidade de pontos, separados dos outros região por região contendo uma relativa baixa densidade de pontos." (XU; WUNSCH, 2008).

Em mineração de dados, *clustering* é uma análise de agrupamento de dados que tem como objetivo agrupar de maneira automatizada dados baseando-se no seu grau de semelhança ou diferença. Esse critério de semelhança está relacionado ao algoritmo e à definição do problema.

Soluções com *clustering* são frequentemente utilizadas para análise e agrupamento de textos e podem ter como elemento de atribuição ao grupo a medida da distância da quantização desses textos (SEGARAN, 2007).

2.2.3.4 Algoritmo K-Means

Em mineração de dados, existem vários algoritmos não supervisionados de agrupamentos de dados. Alguns exemplos de algoritmos são o *Expectation Maximization*, *DBSCAN*, Agrupamento Hierárquico e o próprio *K-Means*.

Expectation Maximization ou EM executa uma parte de expectativa e outra de maximização. A expectativa cria uma função para atribuir a similaridade entre os dados no modelo atual. E a maximização tenta estender o conjunto aumentando a similaridade dos dados no conjunto. Já o método *DBSCAN*, iterativamente coleta objetos alcançáveis por densidade diretamente de pontos centrais, que pode envolver a união de alguns *cluster* alcançáveis por densidade. O processo termina quando nenhum novo ponto pode ser adicionado a qualquer *cluster*. O Agrupamento hierárquico é um algoritmo de agrupamento com uma abordagem hierárquica aglomerativa que constrói grupos aninhados de uma maneira sucessiva (AMIDI; AMIDI, 2020).

Por fim, existe o algoritmo *K-Means*. Esse algoritmo não trabalha com dados rotulados e tem como objetivo particionar os dados em K grupos. O conceito principal do *K-Means* é dividir os grupos por distância. Ele atribui os pontos de dados ao grupo que representa a menor distância. Esse processo se divide em quatro etapas (SANTANA, 2017):

1. Inicialização;
2. Atribuição ao *Cluster*;
3. Movimentação de Centroides;
4. Otimização do *K-Means*;

Na inicialização, é preciso definir o número k de centroides, que, como sugere o nome, serão os pontos centrais do grupo de dados. Centroide é um termo utilizado no campo da geometria para referir-se ao ponto de interseção das medianas que fazem parte de um triângulo. Pode-se dizer que o é o ponto onde qualquer linha traçada que passe por ele divide o segmento em dois. Geralmente esses centroides são espalhados de forma aleatória no espaço. Centroides assemelham-se ao centro de massa de um objeto (HIBBELER, 2011).

Na etapa de atribuição ao *cluster*, para cada ponto é calculada a distância desse para cada um dos centroides e então é feita a sua atribuição ao centroide com a menor distância para o ponto. Após atribuir todos os pontos a um grupo, recalcula-se as posições dos centroides de acordo com a média dos valores do seu grupo e essa é a fase de Movimentação dos Centroides.

Na otimização do *K-Means*, é feito novamente uma atribuição de pontos aos *cluster*, mas agora de acordo com a nova posição dos centroides, e, após, é feito novamente a movimentação dos centroides. Essa etapa permanece repetindo as etapas 2 e 3 até que os *clusters* tornem-se estáticos. Pode-se perceber na Figura 2.4 a representação de 3 grupos com 3 centroides.

2.2.3.5 Silhouette Score

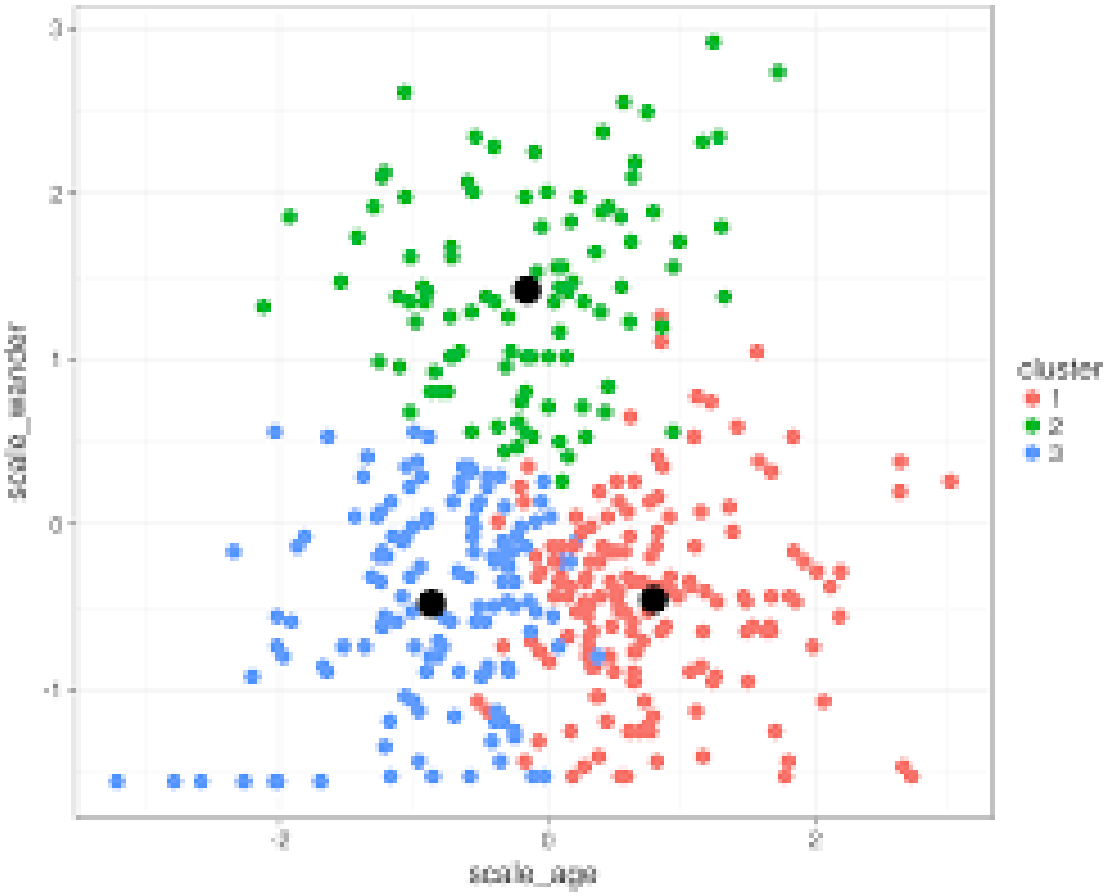
Silhouette refere-se a um método de interpretação e validação de consistência de dados clusterizados. É uma técnica de representação gráfica do quão bem cada objeto foi agrupado (ROUSSEEUW, 1986). Essa métrica será essencial para definir a quantidade de *clusters* de forma autônoma no *K-Means*.

Uma vez que o *K-Means* não consegue definir quantos *clusters* terá no espaço, o *Silhouette* score é um valor de -1 a +1 que consegue medir a pontuação da clusterização com um dado número de *clusters*. Então, uma boa abordagem é usar no *K-Means* a quantidade de *clusters* com a maior pontuação na medida *Silhouette*.

2.3 ***System Usability Scale***

O *System Usability Scale* (i.e., SUS) é um método de averiguação do nível de usabilidade de um sistema dos mais conhecidos. Ele se tornou popular devido, entre outros motivos, ao fato de apresentar um balanço entre ser cientificamente apurado e

Figura 2.4: Representação dos pontos no espaço em um algoritmo *K-Means* .



Fonte: Yobero (2018).

também não ser extremamente longo para o usuário que está sob teste e o pesquisador.

O método foi criado por John Brooke em 1986, e é usado para avaliar produtos, serviços, softwares, hardwares, aplicações *mobile*, aplicações Web e outros tipos de interface. Basicamente, o SUS ajuda a avaliar a efetividade eficiência e satisfação. O índice de efetividade avalia se os usuários conseguem completar o objetivo proposto pela interface. Já o índice de eficiência mede quanto esforço e recursos são necessários para que esse objetivo seja alcançado. E por fim, a satisfação tenta indicar o quão satisfatória foi a experiência do usuário ao utilizar a interface para atingir seu objetivo.

Esse teste consiste em um questionário de 10 perguntas e para cada pergunta, o usuário responde em uma escala de 1 a 5 onde 1 significa que ele está em total desacordo com a pergunta e 5 significa que ele está completamente de acordo com a pergunta. O teste foi idealizado para ser aplicado após o usuário tentar realizar um determinado grupo de tarefas através da interface disponibilizada para ele.

Após colher os resultados, é necessário calcular a pontuação final, tal como será descrito a seguir. Existem vários métodos para calcular essas respostas, um deles é para perguntas ímpares (1, 3, 5, 7, 9) deve-se subtrair 1 da pontuação que o usuário respondeu, para perguntas pares (2, 4, 6, 8 e 10) deve-se subtrair a resposta de 5. Ao final, deve ser feita a soma de todos os valores das perguntas e multiplicar por 2,5. E essa será a pontuação final, podendo ir de 0 a 100. A média do *SUS* é 68 pontos. Caso a interface realize menos pontos, provavelmente ela possui problemas de usabilidade bem relevantes.

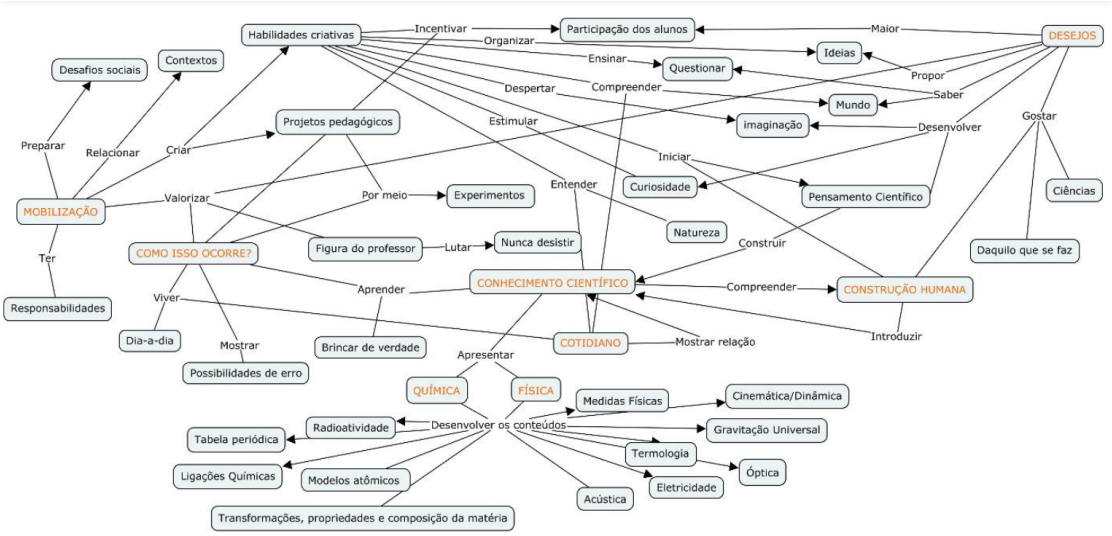
3 TRABALHOS RELACIONADOS

Este capítulo descreve algumas ferramentas que foram identificadas como relevantes para o trabalho remoto em equipe e que podem servir de base para a identificação de funcionalidades para o presente estudo.

3.1 *Brainstorming* no ensino de Química e Física

Foi feito um estudo em uma escola de educação básica a respeito da prática de *brainstorming* e *clustering* visando explorar melhor a concepção dos alunos à respeito das disciplinas de Física e Química. Esses estudantes envolvidos foram divididos em grupos para construírem um mapa conceitual (Figura 3.1) para chegar às conclusões sobre a importância das disciplinas. O estudo conseguiu extrair conceitos-chaves como construção humana, cotidiano, desejos e conhecimento científico. Ao fim do experimento, pode-se constatar que essa atividade possibilitou realizar uma análise do que se espera dos profissionais e abordagens que podem ser utilizadas nas aulas (MELO; KIPPER, 2020).

Figura 3.1: Mapa conceitual construído pelos estudantes.



Fonte: Melo e Kipper (2020).

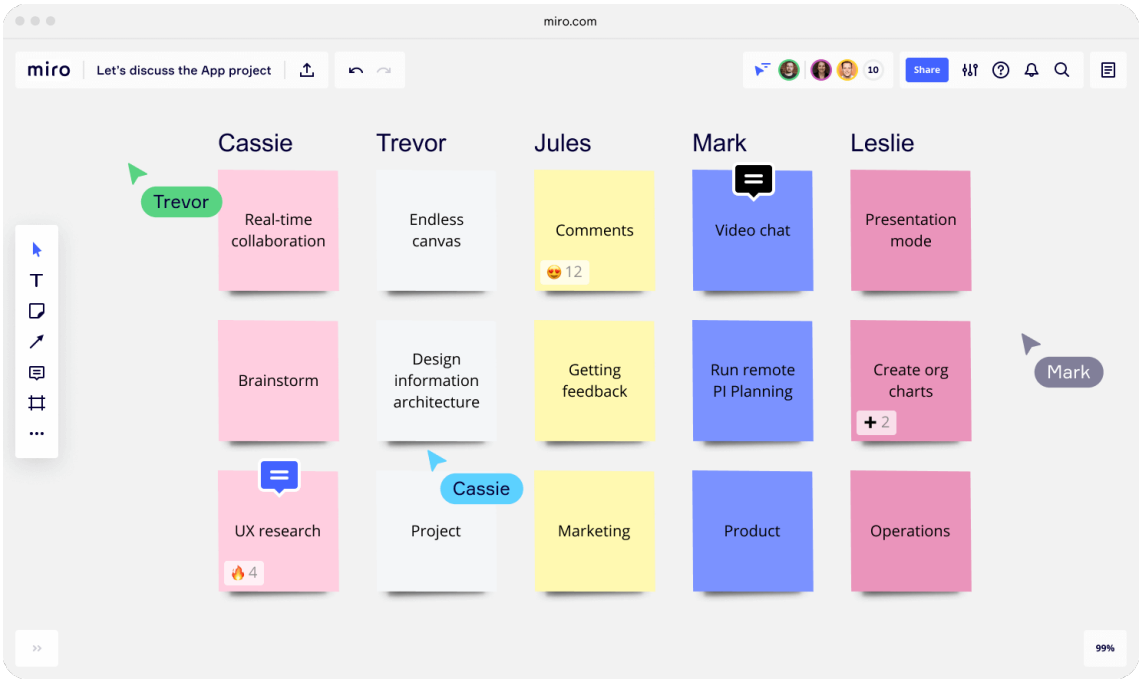
3.1.1 Ferramentas colaborativas para *Brainstormings*

Existem muitas ferramentas que hoje auxiliam em processos de brainstorming. Com a migração em massa para o trabalho remoto no ano de 2020, nota-se que houve

uma grande expansão na busca por esse tipo de ferramenta de auxílio. Também é possível perceber que surgiram muitas melhorias nesses serviços e muita pesquisa nessa área. Alguns exemplos de ferramentas utilizadas são o Miro e o LucidChart.

O Miro[1] (Figura 3.2) é uma ferramenta online que auxilia na criação de mapas mentais, diagramas e quadros com notas em tempo real de forma colaborativa. Sem duvidas uma das ferramentas mais versáteis, conseguindo oferecer uma diversidade de *templates* para inúmeros tipos de projetos, bem como pode-se incluir comentários, fotos, e outros conteúdos relevantes à reunião. Uma de suas principais características é permitir a edição em tempo real. Os usuários conectados podem interagir e visualizar o que cada um está editando no mesmo momento, o que o torna uma ferramenta colaborativa. Essa ferramenta possui suporte para plataformas mobile e web.

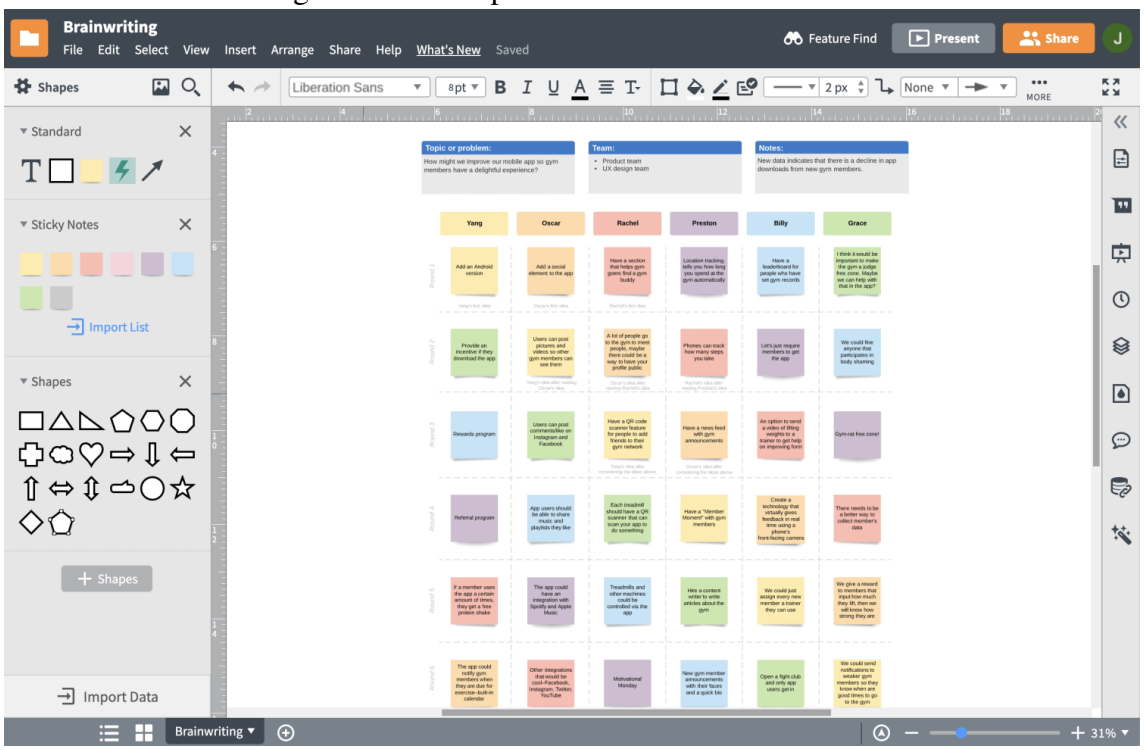
Figura 3.2: Exemplo da ferramenta Miro.



Outra ferramenta bastante conhecida é o LucidChart[2], ilustrado na Figura 3.3. Nessa ferramenta, a maioria das *features* é muito semelhante às do Miro, porém possui um foco maior no ciclo de vida completo de projetos. Ele possui recursos avançados de lousa digital, além de ter a colaboração em tempo real, possui tela de desenho infinita. Possui avançados recursos de diagramação, formatação e geração automática de fluxos. E por fim, possui uma grande integração com bibliotecas de formas para UML, diagramas, organogramas e outros.

[1]Disponível em: <https://miro.com>
[2]Disponível em: <https://www.lucidchart.com/pages/pt>

Figura 3.3: Exemplo da ferramenta LucidChart.



3.1.2 Análise das ferramentas

Essas ferramentas são amplamente utilizadas por equipes remotas e conseguem atender aos mais diversos tipos de reuniões com suas inúmeras funcionalidades. No entanto, cada usuário pode ser identificado por seu login na plataforma fazendo com que a geração de conteúdo não seja anônima. Também, pelo excesso de opções e funcionalidades disponíveis nelas, pode levar um tempo para os usuários conseguirem configurar um *brainstorming*. Por fim, nenhuma delas consegue agrupar os *cards* dos usuários, e esse processo acaba sendo muito demorado quando muitas ideias são levantadas em uma reunião. A motivação da criação deste projeto foi principalmente por ainda não existir uma ferramenta que realize *brainstormings* remotos e que ofereça o agrupamento das ideias automaticamente.

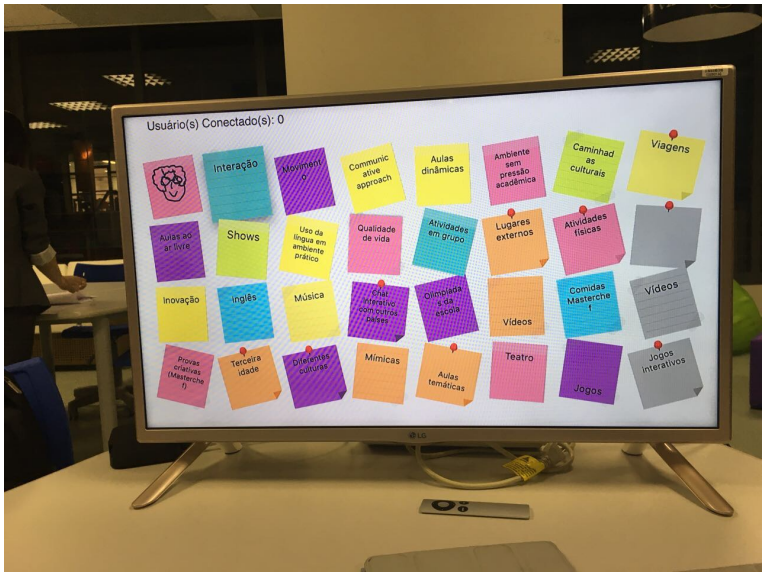
4 METODOLOGIA PROPOSTA

No ano de 2017, foi realizado um *MVP*. O Produto Viável Mínimo (i.e., *MVP*) é um termo utilizado para referir-se à versão mais simples de um produto que pode ser lançada com a quantidade mínima de esforço de desenvolvimento. É muito utilizado para fazer testes com usuários poupando recurso e tempo de uma aplicação que poderia substituir o método utilizado atualmente para realizar esse tipo de reunião. A aplicação, bastante simples, utilizava uma Apple TV, como pode ser visto na Figura 4.1, para ser o quadro de ideias (i.e., *Board*) da turma. O *Board* é o local onde serão fixadas as notas adesivas dos participantes e fica visível para o time inteiro.

A Figura 4.2 mostra cada usuário com seu dispositivo (iPhone ou iPad), o qual poderia escrever ou desenhar nas notas adesivas e enviá-las para a Apple TV, onde seriam mostradas todas as notas de todos os usuários. Com esse teste, foi possível perceber que o time que estava utilizando essa ferramenta foi o primeiro a finalizar todo o *brainstorming*.

Pode-se perceber então a praticidade de algo virtual para acelerar a criação de ideias. Porém no momento os usuários ainda tinham que fazer o agrupamento das ideias propostas manualmente.

Figura 4.1: Aplicação executando na Apple TV com as ideias do time.



Fonte: Autor.

Esse *MVP* foi o precursor do trabalho atual. A proposta consiste, portanto, em desenvolver uma aplicação iOS que permita que os usuários consigam realizar um *brainstorming* dentro da aplicação e que esta aplicação realize a etapa de clusterização de forma automatizada. A seguir serão descritas as etapas realizadas para o desenvolvimento da

Figura 4.2: Interação do time com a aplicação.

Fonte: Autor.

aplicação, a inclusão do algoritmo de agrupamento e sua avaliação e validação.

4.1 Desenvolvimento iOS

Ao planejar o início do projeto, foi necessário definir a plataforma que seria utilizada pelos usuários. Como em sua essência, *brainstormings* são processos bastante manuais, existe a dificuldade de colocar esses textos em uma máquina para analisar e gerar o resultado de grupos para os usuários.

Uma abordagem para resolver esse problema é começar um processo de *brainstorming* virtual. Nesse processo de *brainstorming* virtual, os participantes geram o conteúdo direto na ferramenta de processamento.

Com isso, a abordagem será criar uma aplicação interativa e fácil de utilizar, que consiga replicar as ações dos participantes de forma virtual. Foi decidido utilizar tecnologias Apple, tal como Swift e SwiftUI, descritas nas subseções seguintes.

4.1.1 Swift

Swift é uma linguagem de programação criada em 2014 para desenvolvimento em sistemas iOS, macOS, watchOS, tvOS e possui compatibilidade com Linux (METZARCHIVE, 2014; WEBER, 2014). Atualmente, Swift é uma alternativa à linguagem

Objective-C, empregando conceitos modernos da teoria de linguagem de programação e oferecendo uma sintaxe mais simples, moderna, flexível e rápida.

A escolha por desenvolvimento em Swift é justamente pela familiaridade com a linguagem no desenvolvimento em sistemas iOS e também para poder utilizar do novo *framework* de desenvolvimento de telas: o SwiftUI.

4.1.2 SwiftUI

Partindo da escolha da linguagem de programação Swift, precisa-se definir como será feita a arquitetura da aplicação e por consequência suas telas. Nesse caso, existe a opção de fazer por *Storyboards*, *Xibs*, *View Code* legada ou SwiftUI.

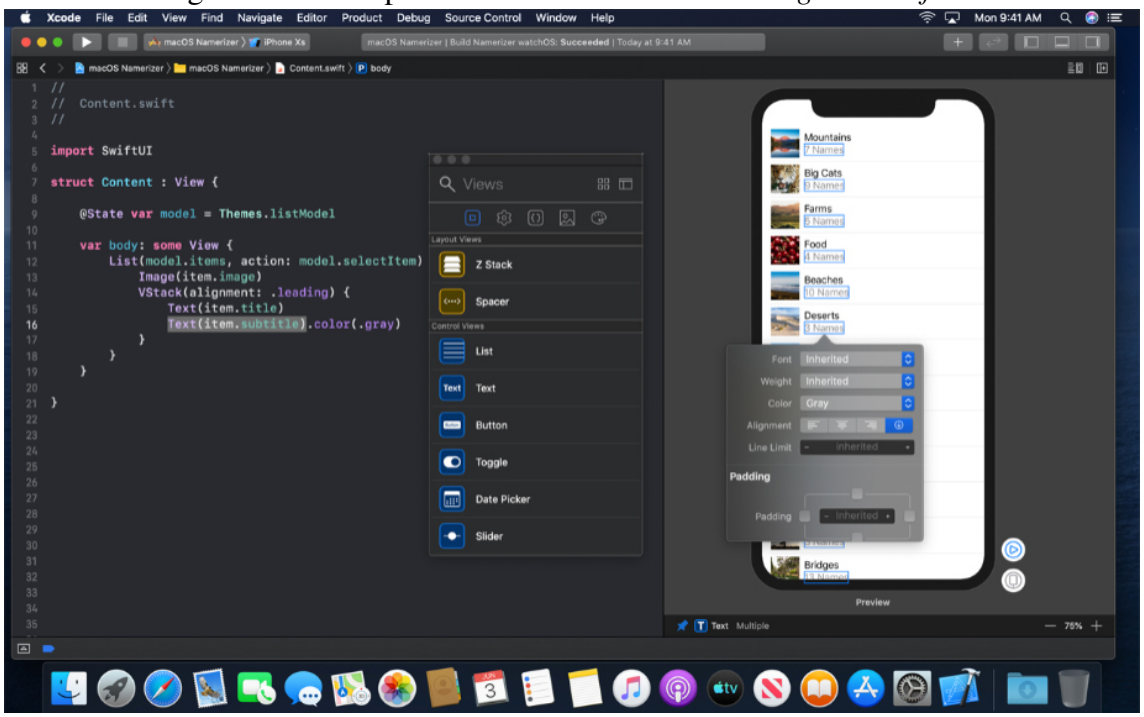
A escolha de SwiftUI foi pela praticidade de conseguir desenvolver em um único arquivo, telas para vários tamanhos de dispositivos tanto *mobile* quanto *desktops*. SwiftUI utiliza uma sintaxe declarativa simples de definir o status da interface de usuário. Desenvolver listas de itens que costumam ser partes demoradas em uma implementação de tela, podem ser descritas com SwiftUI em poucas linhas, como visto na Figura 4.3. Além disso, é fácil de ler e utiliza uma forma de escrita natural, o que ajuda na manutenção do código (MOREFIELD SARAH REICHELT; BELLO, 2020). Outro ponto que facilita na implementação das telas é a ferramenta de design que mostra em tempo de escrita como está o leiaute da tela desenvolvida.

4.1.3 Persistência dos dados

Para persistir os dados em nuvem, foi usado o Firebase. O Firebase é uma ferramenta do Google e possui extensões que auxiliam no desenvolvimento, manutenção e crescimento da aplicação. Ele é denominado um *Backend-as-a-Service*, que possui uma variedade de ferramentas para desenvolvedores para auxiliar em diversas funções da aplicação. É também categorizado como um programa de banco de dados NoSQL que armazena dados no estilo JSON de documentos. Suas principais funções são: autenticação, banco de dados em tempo real, hospedagem, testes de laboratório e notificações. Nessa aplicação, está sendo usado o Cloud Firestore do Firebase, que armazena os dados enviados em JSON de maneira simples e organizada (SINGH, 2018).

Para persistir os dados locais, optou-se por utilizar uma ferramenta nativa do iOS

Figura 4.3: Exemplo de uso da ferramenta de *design* do *SwiftUI*.



Fonte: Morefield Sarah Reichelt e Bello (2020)

chamada *User Defaults* (VRIES, 2020). O *User Defaults* é um arquivo do tipo *.plist* no pacote da aplicação onde pode-se atribuir ou resgatar partes de dados. É uma estrutura muito semelhante a um dicionario de dados e conhecido também por ser um armazenamento do tipo chave-valor. O que acaba sendo muito parecido com dados de *JSON* (Figura 4.4).

Figura 4.4: Exemplo do armazenamento em *User Defaults*.

Key	Type	Value
▼ Root	Dictionary	(3 items)
name	String	John Doe
language	String	English
occupation	String	Stuntman

Fonte: Vries (2020)

4.1.4 Experiência de Usuário, Fluxos e Navegação

O leiaute das telas foi feito pelo autor, seguindo padrões da Apple: *Human Interface Guidelines*[1]. Usando somente componentes já disponibilizados pelos *frameworks* nativos da linguagem, justamente para não ter uma sobrecarga desnecessária com a parte

[1]Disponível: <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>

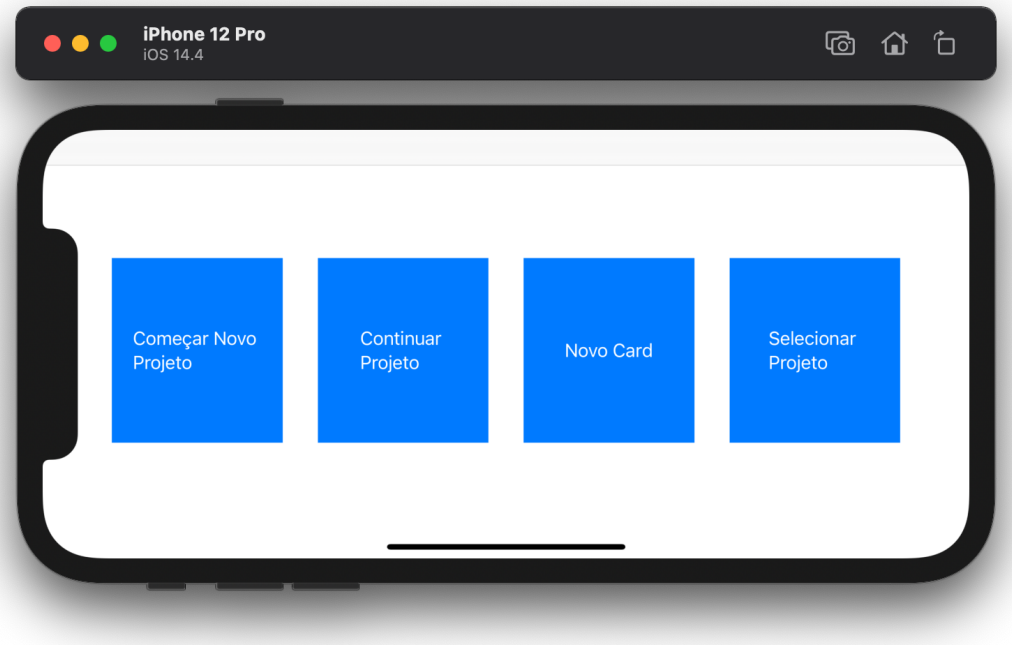
visual.

Existem 5 telas:

1. Menu inicial;
2. Continuar Projeto;
3. Edição de *Card*;
4. Lista de *Boards*;
5. *Board*;
6. Sessão de *clustering*.

A primeira tela, ilustrada na Figura 4.5 é o menu de escolhas do usuário. A Figura 4.6 mostra as telas de Edição, lista de *Boards* e *Board*. Na segunda tela, o usuário edita a nota que deseja enviar para o *Board* podendo escolher um título, um texto para a nota e também a cor. Na terceira tela, mostra a lista de *Boards*. A quarta tela, mostra o *board*, que é onde ficará visível todas as notas enviadas. Finalmente, na Figura 4.7 é a tela onde mostra o agrupamento realizado pela ferramenta, chamada de Sessão de *clustering*, e pode ser acessada a partir da tela de *Board*.

Figura 4.5: Menu Inicial.



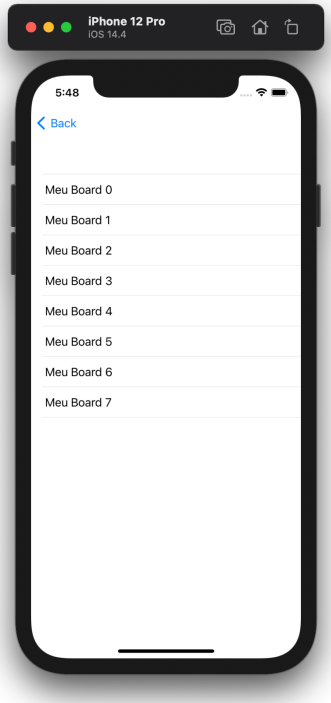
Fonte: Autor.

Com um design visual simples, o maior foco foi pensar em uma experiência de usuário intuitiva que fosse prático de iniciar um processo de *brainstorming* ou recuperar

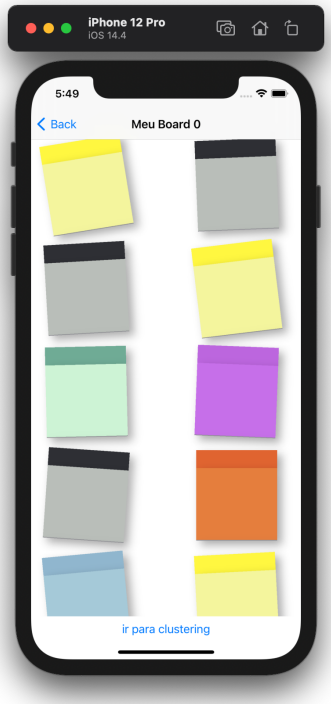
Figura 4.6: Telas de edição, lista de *boards* e *board*.

(a) Edição de Card

(b) Lista de Boards

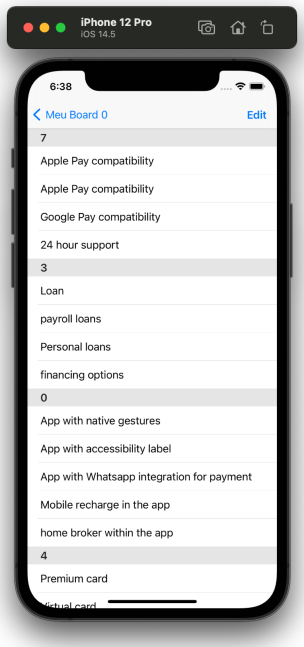


(c) Board



Fonte: Autor.

Figura 4.7: Sessão de *clustering*.

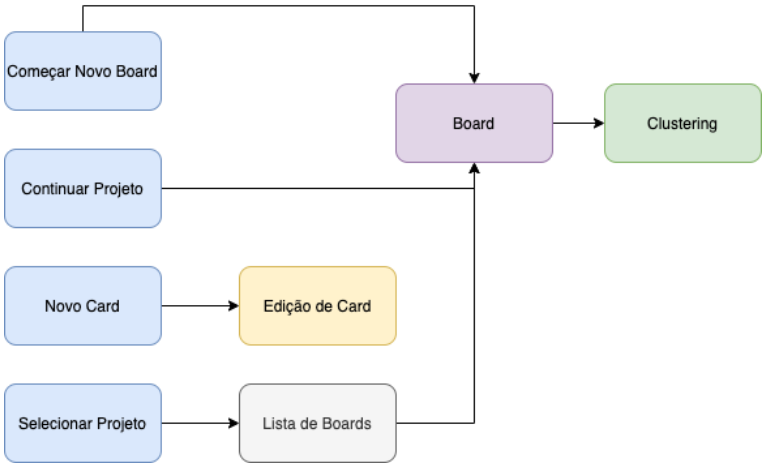


Fonte: Autor.

um processo já iniciado. Tendo esses princípios acima mencionados, o menu principal consiste em 4 principais opções:

1. Começar Novo Projeto;
2. Continuar Projeto;
3. Novo Card;
4. Selecionar Projeto;

Figura 4.8: Diagrama de Fluxo.



Fonte: Autor.

Na Figura 4.8 mostra os fluxos da aplicação. A primeira opção é para iniciar

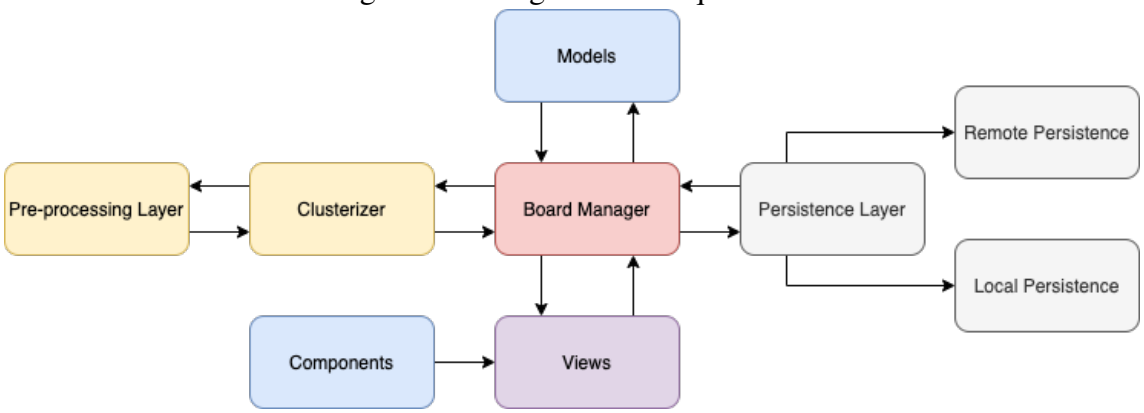
um novo projeto, o que provavelmente será a primeira escolha do usuário para realizar o processo. A segunda opção é para aqueles que já acessaram uma vez o projeto e desejam continuar o último processo corrente de forma a agilizar a busca dos usuários. A terceira opção é para o usuário começar a enviar suas frases no *brainstorming* corrente. A quarta opção serve para acessar a lista de projetos que foram iniciados e se quiser, abrir algum deles.

A partir da nota, o usuário pode escolher a cor que deseja na sua nota e enviar para o último *board* aberto.

A partir da escolha do *board* no menu inicial, o usuário pode visitar o *board* de interesse. E visitando o *board* de interesse, o usuário pode visitar os *clusters* criados a partir das notas correntes.

4.1.5 Organização do projeto Mobile

Figura 4.9: Diagrama da Arquitetura.



Fonte: Autor.

Conforme a Figura 4.9 O projeto está dividido em sete módulos principais:

1. Gerenciador dos *Boards*;
2. Componentes;
3. Telas
4. Modelos;
5. Camada de persistência
6. Clusterizador;
7. Camada de pré-processamento

O gerenciador dos *boards* é o ponto central do projeto. Nesse módulo passa toda a

informação que será destinada as telas, assim como os comandos para iniciar o processo de classificação, persistir os modelos ou mostrá-los na tela. Nesse módulo acontece a instanciação do *board* corrente, da lista de *boards* e de todas as notas do usuário.

O grupo de componentes é o conjunto de estruturas da interface de usuário que são utilizadas em todas as telas. Separá-los das telas facilita para reaproveitar o mesmo componente em mais de uma tela. Por exemplo, as notas que são usadas na tela de criar uma nova nota são os mesmos componentes que aparecem no *board*. As telas representam a etapa que o usuário está navegando pelos fluxos.

Os modelos são as estruturas de dados que serão salvas em JSON na camada de persistência dos dados. Na camada de persistência, pode ser acoplado qualquer serviço de persistência local ou remoto para o projeto. Essa camada simplesmente tem a função de buscar os dados ou fazer uma requisição de salvar esses dados. Atualmente, esses dados estão sendo persistidos no *User Defaults* do dispositivo (persistência local) e no Firebase (persistência remota).

Por fim, existe o Clusterizador onde os dados serão enviados para serem agrupados. O Clusterizador envia os dados para a camada de pré-processamento para conseguir uma classificação mais limpa.

4.2 Python

Python é uma linguagem de programação de alto nível que possui um interpretador e é escrita em forma de *scripts*, imperativa, orientada a objetos, funciona, de tipagem dinâmica e forte. Python possui um modelo de desenvolvimento comunitário. A linguagem possui como filosofia enfatizar a importância do esforço do programador sobre o esforço computacional. Ela prioriza a legibilidade do código sobre a velocidade ou expressividade. Python possui um interpretador interativo, que acaba sendo um dos diferenciais da linguagem. Com esse interpretador, há a possibilidade de testar o código de um programa e receber o resultado em tempo real, antes de iniciar a compulação ou incluí-las nos programas. Possui também um analisador sintático. Os programas são divididos em linhas lógicas que são separadas pelo *token* de nova linha. Os blocos de código são caracterizados pela identação e pela ausência da pontuação ponto e vírgula.

4.2.1 Django

O framework mais comum para desenvolvimento Web com Python é o Django que utiliza o padrão de arquitetura Model-View-Template (MTV). Originalmente ele foi criado para gerenciar sites jornalísticos e tornou-se um projeto de código aberto a partir de 2005. Esse framework utiliza o princípio DRY (*Don't Repeat Yourself*) que faz com que o desenvolvedor aproveite ao máximo o código já feito, evitando repetições. Por ser popular, Django possui uma grande comunidade de desenvolvedores, o que facilita encontrar materiais de estudo e programadores para auxiliarem e resolver duvidas.

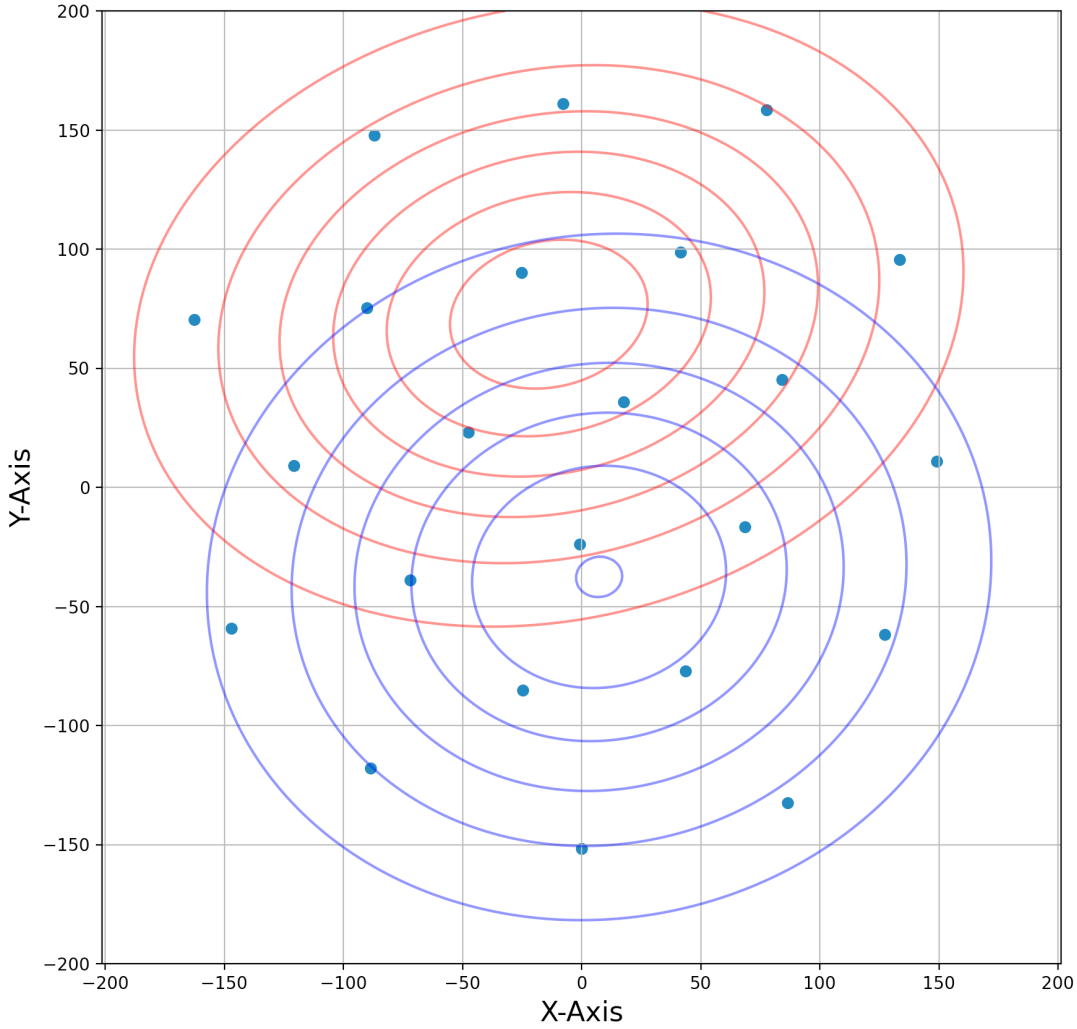
4.3 Implementação da clusterização

Para o agrupamento, foram feitos testes com algumas soluções como *EM* (i.e., *Expectation Maximization*) e *DBSCAN*. Porém *EM* (Figura 4.10) acaba sendo necessário definir o número de *clusters*, e pela dispersão dos dados recebidos do Word2Vec, suas classificações ficaram muito abaixo do esperado. Já *DBSCAN* (Figura 4.11), ficou muito complexo de fazer o ajuste dos dados que também é conhecido como *fit* classificação não funcionou como o esperado, o modelo sofria muito com *overfitting*, ou seja, o modelo se ajustou muito bem aos dados classificados, e se torna ineficaz para cenários com novos dados e *underfitting*, quando o modelo não consegue ver distinção no conjunto de dados, sendo muito difícil de achar um meio termo.

Na Figura 4.10 pode-se perceber que o modelo agrupou todos os dados em um único conjunto. Ocorreu um *underfitting*, que é quando o modelo não consegue encontrar uma relação entre os dados analisados. Na Figura 4.11 o modelo encontra 2 *clusters* porém está longe do esperado, visto que os dados de entrada foram colocados para formarem no mínimo 5 *clusters*. A melhor solução encontrada foi o *K-Means*. Onde a classificação é feita pelo estado mais próximo da média do grupo. Com isso, foram obtidos os melhores resultados dentre as opções escolhidas. Porém esse algoritmo não sabe qual o numero ideal de *clusters*.

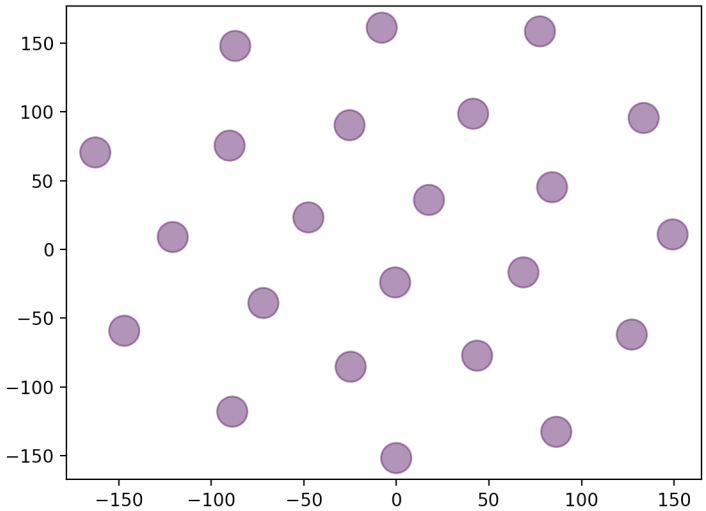
Para descobrir o melhor número de *clusters*, foi usado *silhouette score*. Testando os dados com grupos de 2 até 8 *clusters*, e analisando qual tinha a melhor pontuação. Com isso, foi possível obter grupos de palavras que tinham mais associação entre si.

Figura 4.10: Resultado de *EM* com *underfitting*.



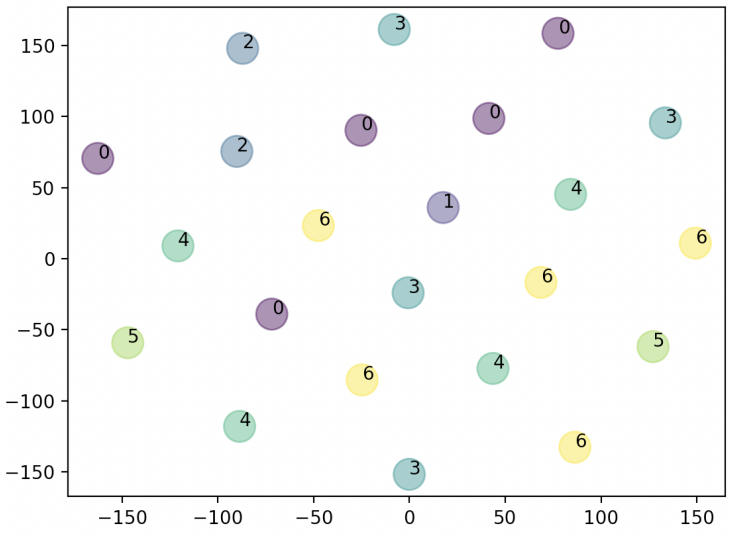
Fonte: Autor.

Figura 4.11: Resultado de *DBSCAN*



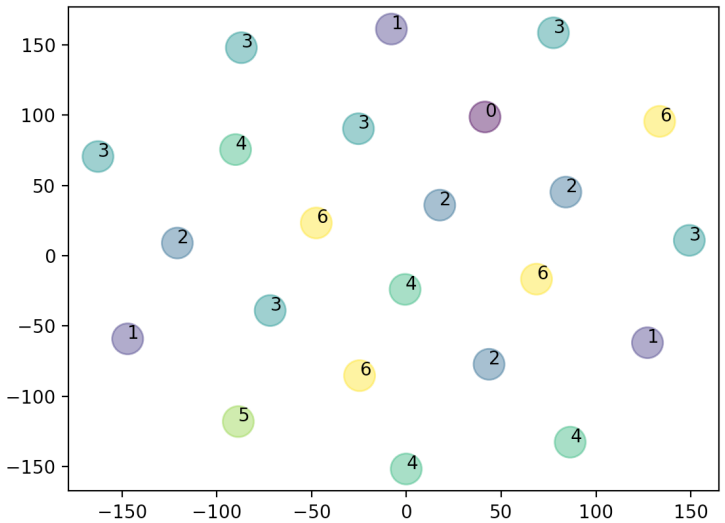
Fonte: Autor.

Figura 4.12: Resultado de possível agrupamento conforme os dados de entrada.



Fonte: Autor.

Figura 4.13: Resultado do *K-Means*

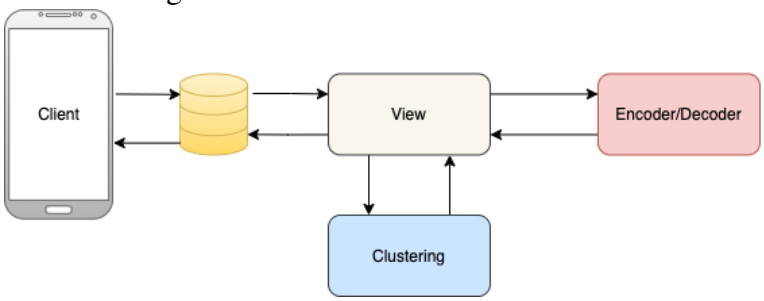


Fonte: Autor.

4.3.1 Organização do projeto de clusterização

O projeto desenvolvido em Python utilizando Django para foi organizado dividido em 3 partes conforme mostra a Figura 4.14: *View*, *Encoder/Decoder* e *Clustering*. A parte da *View* tem a responsabilidade de se comunicar com as aplicações do cliente, e enviar e receber as requisições feitas por esses clientes. Uma vez que recebe uma requisição, a *View* envia para o módulo de *Encode/Decode* para ser feita a decodificação dos dados da requisição, ou seja, a decodificação das sentenças enviadas pelo cliente no formato *JSON*. Após a decodificação, esses dados retornam para a *View* que os envia para o módulo *Clustering* e esse módulo devolve para a *View* as sentenças separadas em seus respectivos grupos. Ao receber essas sentenças classificadas, a *View* envia novamente para o *Encoder/Decoder* para ser feita a codificação dos dados em *JSON* e enviados ao cliente.

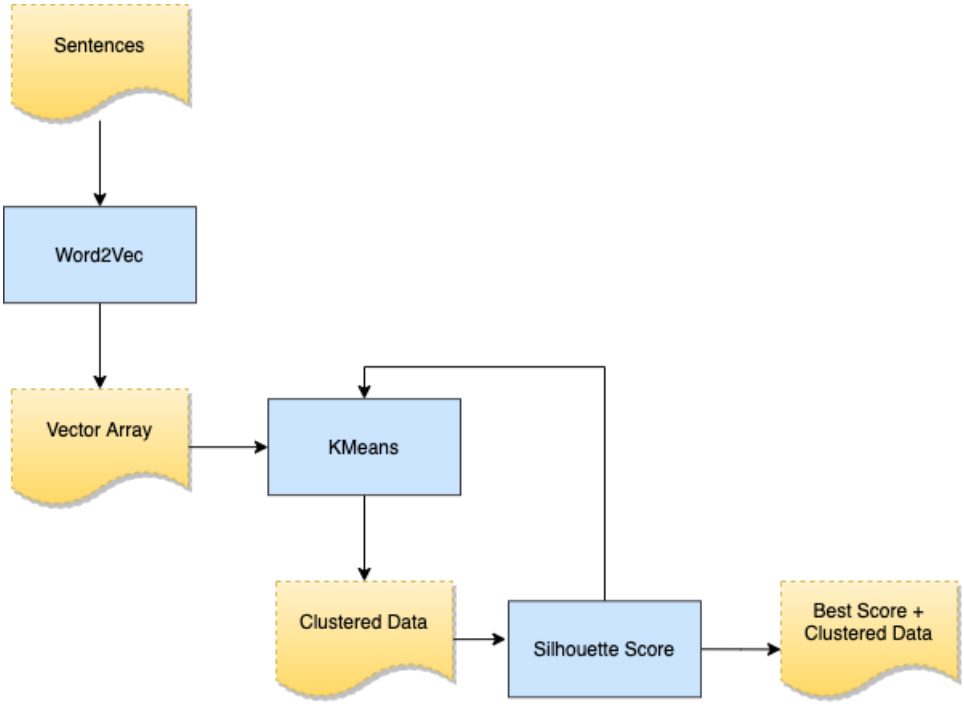
Figura 4.14: Estrutura do *Web Service*



Fonte: Autor.

O módulo de *Clustering* é onde está localizada toda a lógica de agrupamento utilizada para classificar as sentenças, o processo passa por 3 etapas segundo a Figura 4.15: *Word2Vec*, *KMeans* e *Silhouette Score*. Na primeira etapa as sentenças são transformadas num *array* de vetores que definirão a similaridade das sentenças utilizadas. Esses vetores contém as *features* que serão utilizadas no agrupamento feito pelo *KMeans*. O *KMeans* então gera uma classificação a partir desse *array* de vetores e envia para o *Silhouette Score*, que atribuirá um peso àquela classificação. Esse passo de classificação no *KMeans* e pontuação no *Silhouette Score* é realizado por N vezes onde N é o número máximo de clusters definido para o agrupamento. Após, é feita uma comparação das pontuações feitas no *Silhouette Score* e retornará a classificação realizada pelo número de *clusters* que obtiver a maior pontuação.

Figura 4.15: Processo de agrupamento realizado pelo *Web Service*



Fonte: Autor.

5 EXPERIMENTOS, RESULTADOS E DISCUSSÕES

Ao finalizar o *MVP*, foram escolhidos usuários para testar a ferramenta. Foram escolhidos usuários com o perfil de trabalhadores remotos, que participam de empresas com processos criativos ligados a tecnologia e também que trabalhem em alguma instituição financeira. Os requisitos para perfil de teste foram planejados pensando no público-alvo da ferramenta, exceto pelo requisito de trabalhar em uma instituição financeira. Esse último requisito foi escolhido para gerar o assunto do *brainstorming*, que seria: "Melhorias para atrair novos clientes à uma *fintech*". Foi realizado o teste *SUS* a fim de descobrir se existiam problemas graves de usabilidade e como o sistema iria se comportar em uma situação simulada de uso. Os usuários então se reuniram em uma sala de reunião remota para testar o aplicativo para realizar um *brainstorming*. No início, foi apresentado o tema para os usuários "Melhorias para atrair novos clientes à uma *fintech*". Dentro disso, os usuários deveriam sugerir funcionalidades e novas ideias para atrair mais clientes para esta *fintech*. Assim, cada usuário recebeu uma versão do aplicativo e deveria descobrir como utilizá-lo e então enviar suas ideias para o *Board* da reunião. Por aproximadamente 20 minutos os usuários começaram a lançar as ideias no *Board*. Essas ideias foram tanto faladas quanto escritas nas notas adesivas para a visualização do time. O único pedido feito pelo administrador foi que os usuários colocassem essas frases nas notas adesivas na língua inglesa, justamente para conseguir utilizar a biblioteca *Word2Vec* para realizar o agrupamento das ideias no final da sessão.

Após esse período de 20 minutos várias ideias foram levantadas desde ideias relacionadas a software, ideias de marketing, novas funcionalidades e promoções para atrair clientes. Então no final da sessão um dos usuários foi responsável por fazer o agrupamento, simplesmente apertando no botão *"Ir para clustering"* dentro do *Board*. Feito isso eles puderam visualizar o agrupamento realizado automaticamente pela aplicação e discutir se aquele agrupamento feito fazia sentido ou não. Como a aplicação ainda não permite fazer alteração no grupamento feito, o trabalho do grupo encerrou nessa etapa. Porém eles conseguiram perceber que algumas notas adesivas não faziam sentido do jeito que estavam agrupados. Ao final, os usuários receberam um formulário para colocar suas conclusões sobre aplicação, seu uso e para dar *feedbacks* das suas experiências.

O teste *SUS* foi realizado com as seguintes questões:

1. Eu gostaria de usar esse sistema com frequência;
2. Considero o sistema desnecessariamente complexo;

3. O sistema é fácil de usar;
4. Eu precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema;
5. As funções do sistema estão muito bem integradas;
6. O sistema apresenta muita inconsistência;
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente;
8. O sistema é atrapalhado de usar;
9. Eu me senti confiante ao usar o sistema;
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema;

Cada uma dessas questões o usuário tinha a opção de selecionar um valor de 1 a 5 onde 1 ele discorda totalmente do que está sendo dito e 5 ele concorda totalmente com o que está sendo dito. E além das perguntas clássicas do teste de usabilidade, foram incluídas 3 perguntas mais relacionadas com a aplicação em questão e 2 perguntas abertas para os usuários expressarem suas opiniões.

As 3 questões adicionadas foram as seguintes:

1. O sistema facilitou uma tarefa minha;
2. O sistema poderia substituir a maneira que eu realizo essa tarefa hoje em dia;
3. Eu recomendaria o sistema;

E as 2 perguntas abertas foram as seguintes:

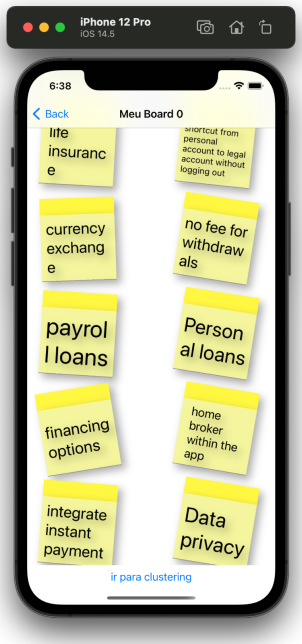
1. O que poderia ser melhor?;
2. Qual foi o diferencial?;

As 3 questões adicionais foram planejadas para ajudar a descobrir se a ferramenta conseguiria competir com ferramentas já existentes no mercado, no ponto de vista dos usuários que realizaram o teste. Já as perguntas abertas tiveram como objetivo colher *feedbacks* sobre falhas em pontos específicos da aplicação e também o que poderia fazer os usuários migrarem de outras aplicações. Essas perguntas ajudam no desenvolvimento e aprimoramento de novas funcionalidades para lançar versões futuras. As figuras 5.1 e 5.2 mostram as telas do *Board* enquanto os usuários estavam colocando suas ideias. E as figuras 5.3 e 5.4 mostram essas ideias na Sessão de *clustering*, já agrupadas.

Figura 5.1: Captura de tela retirada dos testes realizados mostrando o *Board*.

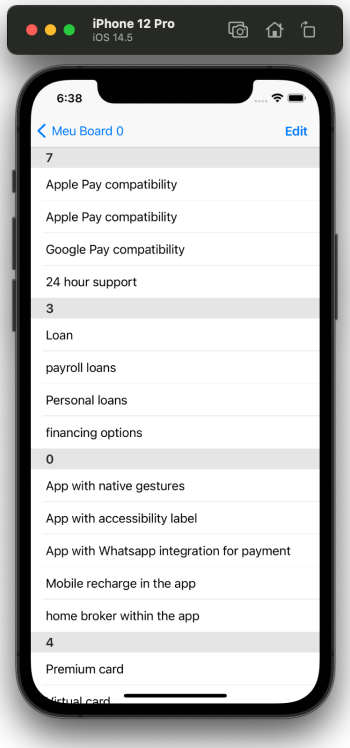
Fonte: Autor.

Figura 5.2: Captura de tela retirada dos testes realizados mostrando o *Board*.



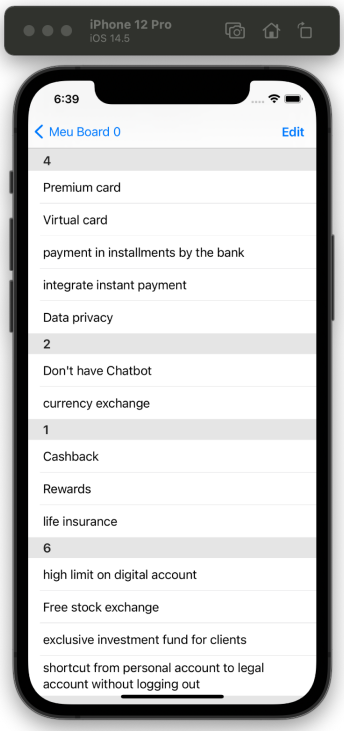
Fonte: Autor.

Figura 5.3: Captura de tela retirada dos testes realizados mostrando a sessão de *clustering*.



Fonte: Autor.

Figura 5.4: Captura de tela retirada dos testes realizados mostrando a sessão de *clustering*.



Fonte: Autor.

5.1 Resultados

O teste *SUS* realizado obteve a pontuação de 82,50. Já as 3 questões adicionais obtiveram a média de 89,3% Para a pergunta "O que poderia ser melhor?", as seguintes respostas foram recebidas:

- "Contagem de card repetido (ex: 2 pessoas colocam ApplePay, mostrar um 2 ao agrupar os cards)"
- "Indicativos visuais na hora de submeter a ideia"
- "Poderia ser listado o nome da categoria que foi dividido e possivelmente as razões do porquê
  das escolhas por inserir os itens naquela categoria"
- "Poderia melhorar na clusterização, alguns cards que acredito que deveriam estar junto estavam em clusters diferentes."
- "Quando rodar para agrupar as ideias, o nome da categoria pelo qual elas foram agrupadas, poderia aparecer ao invés de colocar 0, 1, 2, 3 (sections)."

Os usuários acabaram percebendo que o aplicativo não foi capaz de remover algumas ideias repetidas, além da falta de indicação visual ao submeter uma nova ideia. Outro ponto importante levantado foi que os agrupamentos não possuíam título, algo que poderia ser de grande ajuda no processo de *brainstorming*. E por fim, também levantaram o ponto que algumas das notas foram classificadas de forma errada pelo algoritmo de *clustering*.

Para a pergunta "Qual foi o diferencial?"as seguintes respostas foram recebidas:

- "Velocidade para realizar as cerimônias e participação em conjunto"
- "Portabilidade, facilidade de uso e economia de papel"
- "Facilidade de realizar uma tarefa que poderia demorar para chegar naquele ponto. No caso, ainda acho que valeria uma revisão na divisão dos itens em cada categoria, mas acredito que muito do processo de separação por categoria já estaria feito pelo sistema."
- "A possibilidade de unir a equipe para pensar em soluções mas de maneira anônima, permitindo o usuário inserir algumas ideias sem que seja julgado pela equipe."
- "O agrupamento de ideias semelhantes"

Ao serem questionados sobre o diferencial na usabilidade do aplicativo, surgiram

diferenciais que não foram previamente pensados na concepção da aplicação. Como a questão de conseguir-se realizar um *brainstorming* de maneira completamente anonima, o que permite que os participantes expressem mais suas opiniões sem receio de julgamentos. Também foi mencionada a questão de economia de papel, uma vez que tendo o processo totalmente virtual, empresas não precisem gastar recursos comprando materiais de papel. Além disso, foi possível perceber que a aplicação atingiu um dos seus principais objetivos, que é realizar o processo de *brainstorming* com mais agilidade realizando o agrupamento de ideias semelhantes.

6 CONCLUSÕES

Em ambientes corporativos e educacionais que vivenciam diariamente exercícios de criação de ideias, seja para desenvolvimento de produtos ou para aprimorar conhecimentos, as ferramentas disponíveis são indispensáveis para obtenção dos objetivos. Porém, por mais modernas e desenvolvidas, essas ferramentas ainda não conseguem automatizar partes mais mecânicas desses processos. No caso de *brainstormings*, o agrupamento de ideias consome um tempo significativo dos participantes envolvidos. Esse trabalho tem o objetivo de tornar o processo de *brainstorming* mais eficiente agrupando automaticamente, com métodos de Processamento de Linguagem Natural e Aprendizado de Máquina, todas as ideias colocadas na reunião.

Através das pesquisas feitas com usuários de teste, pode-se perceber que a ferramenta consegue chegar muito próximo de um agrupamento idealizado pelos usuários e assim otimizando o *brainstorming* realizado. Segundo relatos dos usuários, mesmo não conseguindo agrupar totalmente, como esperado pelos usuários, a ferramenta conseguiu otimizar o tempo da reunião uma vez que seria necessário os usuários reordenarem apenas parte das ideias que estavam em grupos errados, e não todas as ideias.

Além disso, foi possível perceber que usuários relataram a economia de papel e facilidade em usar a ferramenta como diferenciais para outras plataformas. Outro ponto foi a anonimidade que a ferramenta trouxe para os usuários, fato considerado importante para um dos usuários em teste.

Acredito que a ferramenta conseguiu cumprir com seu objetivo principal de otimizar o *brainstorming* porém, ainda precisa-se de ajustes antes de ser lançada no mercado. A interface ainda não é amigável para os usuários, o que pode ser um problema para competir com as ferramentas existentes hoje em dia. Além disso, pode ser necessário incluir uma rede neural no fim do processo para conseguir aprimorar o agrupamento, a partir dos *feedbacks* dos usuários. Com essas duas principais *features* a ferramenta estaria em condições de competir com as existentes atualmente.

REFERÊNCIAS

AMIDI, A.; AMIDI, S. **Introdução ao aprendizado não supervisionado**. 2020. <https://stanford.edu/~shervine/l/pt/teaching/cs-229/dicas-aprendizado-nao-supervisionado>.

BARROS, P. **Aprendizagem de Maquina: Supervisionada ou Não Supervisionada?** 2016. <https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-n~ao-supervisionada-7d01f78cd80a>.

HIBBELER, R. C. **Estática - Mecânica Para Engenharia 12ª Ed. 2011**. [S.l.: s.n.], 2011.

KOEHN, P. **Statistical Machine Translation**. 2017. <https://arxiv.org/pdf/1709.07809.pdf>.

MELO, C. B. da S.; KIPPER, L. M. **MAPA CONCEITUAL POR MEIO DO BRAINSTORMING E CLUSTERING: EXPERIÊNCIA NA DISCIPLINA PRÁTICA DE ENSINO EM FÍSICA**. 2020. Available from Internet: <https://periodicoscientificos.ufmt.br/ojs/index.php/reamec/article/view/9546/pdf>.

METZARCHIVE, R. **Apple Seeks a Swift Way to Lure More Developers**. 2014. <https://www.technologyreview.com/2014/06/03/250717/apple-seeks-a-swift-way-to-lure-more-developers/>.

MIKOLOV, T. et al. **Efficient Estimation of Word Representations in Vector Space**. 2013.

MOREFIELD SARAH REICHELT, A. T. B.; BELLO, A. **SwiftUI by Tutorials**. 2020.

MOURA, W. **Word Embedding**. 2018. <https://hackinganalytics.com/2018/01/31/word-embedding/>.

PEREIRA, S. do L. **Processamento de Linguagem Natural**. 2020. <https://www.ime.usp.br/~slago/IA-pln.pdf>.

ROUSSEEUW, P. J. **Silhouettes: a graphical aid to the interpretation and validation of cluster analysis**. 1986. <https://www.sciencedirect.com/science/article/pii/0377042787901257>.

SAHIN Özgür. **Introduction to Apple ML Tools**. 2020.

SANTANA, F. **Entenda o Algoritmo K-means e Saiba como Aplicar essa Técnica.** 2017. <https://minerandodados.com.br/entenda-o-algoritmo-k-means/>.

SEGARAN, T. **Programming Collective Intelligence: Building Smart Web 2.0 Applications**. [S.l.]: O'REILLY, 2007.

SINGH, V. **Introduction to Firebase**. 2018. <https://medium.com/codingurukul/introduction-to-firebase-f9f6ccc8a785>.

SMITH, R. **Red, Green, Blue: A Speedy Process for Sorting Brainstorm Ideas**. 2020. <https://digitalfacilitation.net/?p=596>.

STANGE, R. L. **Adaptatividade em Aprendizagem de Máquina: Conceitos e Estudo de Caso**. 2011. <https://www.teses.usp.br/teses/disponiveis/3/3141/tde-02072012-175054/publico/Dissertacao_RLStange_2011_Revisada.pdf>.

TOMALOK, E. **Word2Vec e sua importância na etapa de pré-processamento**. 2019. <https://medium.com/@everton.tomalok/word2vec-e-sua-importancia-na-etapa-de-pre-processamento-d0813acfc8ab>.

VRIES, R. de. **Working with UserDefaults in Swift**. 2020. <https://learnappmaking.com/userdefaults-swift-setting-getting-data-how-to/>.

WEBER, H. **Apple announces 'Swift,' a new programming language for OS X & iOS**. 2014. <https://venturebeat.com/2014/06/02/apple-introduces-a-new-programming-language-swift-objective-c-without-the-c/>.

XU, R.; WUNSCH, D. **Clustering**. Wiley, 2008. (IEEE Press Series on Computational Intelligence). ISBN 9780470382783. Available from Internet: <https://books.google.com.br/books?id=kYC3YCyl_tkC>.

YOBERO, C. **K-Means Clustering Tutorial**. 2018. <https://rpubs.com/cyobero/k-means>.