CENTRO UNIVERSITÁRIO DO PARÁ - CESUPA
ESCOLA DE NEGÓCIOS, TECNOLOGIA E INOVAÇÃO - ARGO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Elielson Fernando do Santos Barbosa
Lucas da Silva Freitas
Pedro Augusto Pinto de Lima

iTutor - Aplicação híbrida para medida de aleatoriedade em redes de discussão

BELÉM
2022

Elielson Fernando do Santo Barbosa
Lucas da Silva Freitas
Pedro Augusto Pinto de Lima

iTutor - Aplicação híbrida para medida de aleatoriedade em redes de discussão

Trabalho de conclusão de curso apresentado à Escola de Negócios, Tecnologia e Inovação do Centro Universitário do Estado do Pará como requisito para obtenção do título de Bacharel em Ciência da Computação na modalidade PRODUTO.

Orientador(a): Me. Fábio Rocha de Araújo
Coorientador(a):Dr.Cláudio Eduardo Corrêa Teixeira

BELÉM
2022

Elielson Fernando do Santos Barbosa
Lucas da Silva Freitas
Pedro Augusto Pinto de Lima

iTutor - Aplicação híbrida para medida de aleatoriedade em redes de discussão

Trabalho de conclusão de curso apresentado à Escola de Negócios, Tecnologia e Inovação do Centro Universitário do Estado do Pará como requisito para obtenção do título de Bacharel em Ciência da Computação na modalidade PRODUTO.

Data da aprovação: 05 / 12 / 2022
Nota final aluno(a) I: 9,5
Nota final aluno(a) II: 9,5
Nota final aluno(a) III: 9,5

Banca examinadora

Documento assinado digitalmente
gov.br FABIO ROCHA DE ARAUJO
Data: 21/12/2022 10:51:40-0300
Verifique em https://verificador.iti.br

Prof(a). Me. Fábio Rocha de Araújo
Orientador(a) e Presidente da banca

Documento assinado digitalmente
gov.br POLYANA SANTOS FONSECA NASCIMENTO
Data: 21/12/2022 16:02:38-0300
Verifique em https://verificador.iti.br

Prof(a). Ma. Polyana Fonseca
Examinador(a) interno(a)

Documento assinado digitalmente
gov.br ISAAC SOUZA ELGRABLY
Data: 21/12/2022 10:20:34-0300
Verifique em https://verificador.iti.br

Prof(a). Dr. Isaac Elgrably
Examinador(a) interno(a)

Dados Internacionais de Catalogação-na-publicação (CIP)
Biblioteca do CESUPA, Belém – PA

Barbosa, Elielson Fernando dos Santos.

iTutor: aplicação híbrida para medida de aleatoriedade em redes de discussão / Elielson Fernando dos Santos Barbosa, Lucas da Silva Freitas, Pedro Augusto Pinto de Lima; orientador Fábio Rocha de Araújo. – 2022.

Trabalho de Conclusão de Curso (Graduação) – Centro Universitário do Estado do Pará, Ciência da Computação, Belém, 2022.

1. Teoria dos grafos. 2. Linguagem de programação. 3. Métricas de desempenho. I. Freitas, Lucas da Silva. II. Lima, Pedro Augusto Pinto de. III. Araújo, Fábio Rocha de, orient. IV. Título.

CDD 23ª ed. 005.131

Dedicamos este trabalho a todos que amamos.
Em especial àqueles que acreditam que a tecnologia
pode mudar a nossa sociedade para melhor.

AGRADECIMENTOS

Gostaríamos primeiramente de agradecer a Deus, que nos possibilitou nos deu forças para concluir este trabalho com sucesso e orgulho do que escrevemos e entregamos. Segundamente gostaríamos também de agradecer às nossas famílias, que nos deram a oportunidade de fazer essa faculdade, nos possibilitando uma oportunidade que poucos podem ter no nosso país e também nos deram o suporte necessário para que pudéssemos chegar ao fim desta jornada. A esta universidade, aos docentes, coordenadores e administração que proporcionaram os melhores conhecimento para poder desenvolver este trabalho. Aos professores Me. Fábio Araújo e Dr. Cláudio Teixeira pela oportunidade e apoio durante todo o processo de orientação e construção deste TCC. Ao professor Dr. Alan Souza, por nos elucidar em certos tópicos durante a execução do trabalho.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API - Application Programming Interface

CE - Consulta externa

CPU - Central Processing Unit

CRUD - Create, Read, Update and Delete

EE - Entrada externa

ER - Erdos-Renyi

GB - Gigabyte

GiB - Gibibyte

IOS - iPhone operating system

JSON - JavaScript Object Notation

MVP - Minimum viable product

NoSql - not only SQL

PBL - Problem Based Learning

PF - Ponto de função

RAM - Random Access Memory

SE - Saída Externa

SW - Small World

TB - Terabyte

US - United States

UTIs - Unidade de terapia intensiva

WWW - World Wide Web

SUMÁRIO

RESUMO

As redes são estruturas no qual existe um conjunto de elementos conectados entre si, podendo ser observado em vários contextos, como visto em estruturas sociais ou materiais concebidos pela humanidade. Diante disso, a sociedade vem utilizando de redes de discussão com um foco em potencializar a obtenção de uma conclusão concisa acerca de um determinado tema, discussões essas que possuem uma métrica de aleatoriedade a qual avalia a eficiência das interações da rede, com foco em obter os resultados das redes de discussão para classificação da qualidade da discussão. O objetivo deste trabalho de curso é a definição de métricas para análise do desempenho de redes de discussão e a criação de uma aplicação que irá permitir o registro e visualização das interações dessas redes. Para isso foi desenvolvido o aplicativo que registra as interações durante uma discussão e também um servidor que processa esses dados e os retorna para o aplicativo. Os resultados obtidos foram uma aplicação que faz o gerenciamento do fluxo de grupos e discussões, além de também gerar o índice de aleatoriedade e o gráfico de interações, devidamente disponibilizado na play store. Este trabalho de curso cumpriu de forma satisfatória os objetivos propostos, criando métricas e formas de avaliação satisfatórias de redes de discussão.

Palavras-chave: Métricas de Desempenho; React Native; Teoria de Grafos; Linguagem Python; Linguagem R.

ABSTRACT

Networks are structures in which there is a set of elements connected to each other and can be observed in various contexts, as seen in social or material structures designed by humanity. In view of this, society has been using discussion networks with a focus on enhancing the achievement of a concise conclusion about a given topic, discussions that have a randomness metric which evaluates the efficiency of the network interactions, with a focus on obtaining the results of the discussion networks to classify the quality of the discussion. The objective of this course work is the definition of metrics for analyzing the performance of discussion networks and the creation of an application that will allow the recording and visualization of the interactions of these networks. For this, an application was developed that records interactions during a discussion and also a server that processes this data and returns it to the application. The results obtained were an application that manages the flow of groups and discussions, in addition to also generating the randomness index and the interaction graph, duly available in the play store. This course work satisfactorily fulfilled the proposed objectives, creating metrics and satisfactory forms of evaluation discussion networks.

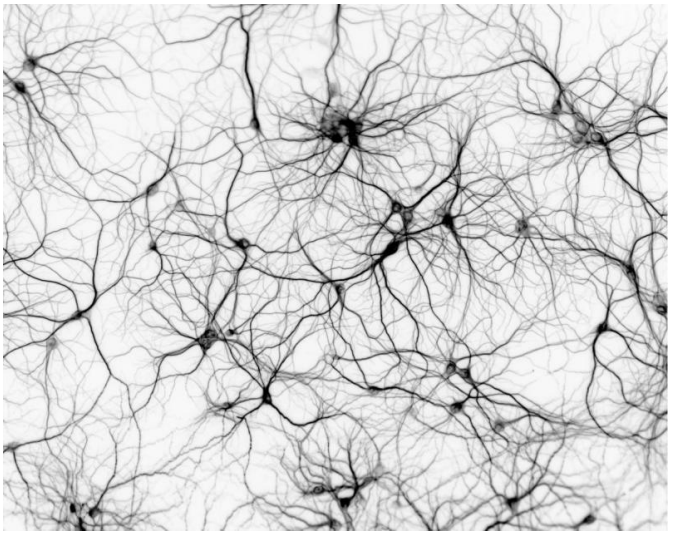
Palavras-chave: Performance Metrics; React Native; Graphs Theory; Python language; R language.

1 CONTEXTUALIZAÇÃO

1.1 Introdução

Redes são estruturas descritas como conjunto de elementos conectados entre si, sendo observada em inúmeros contextos, como visto em estruturas sociais ou materiais concebidas pela humanidade (Carvalho, 2012) (Figuras 1-3).

Figura 1 - As funções cognitivas e psíquicas emergem da atividade de redes de interações neuronais.

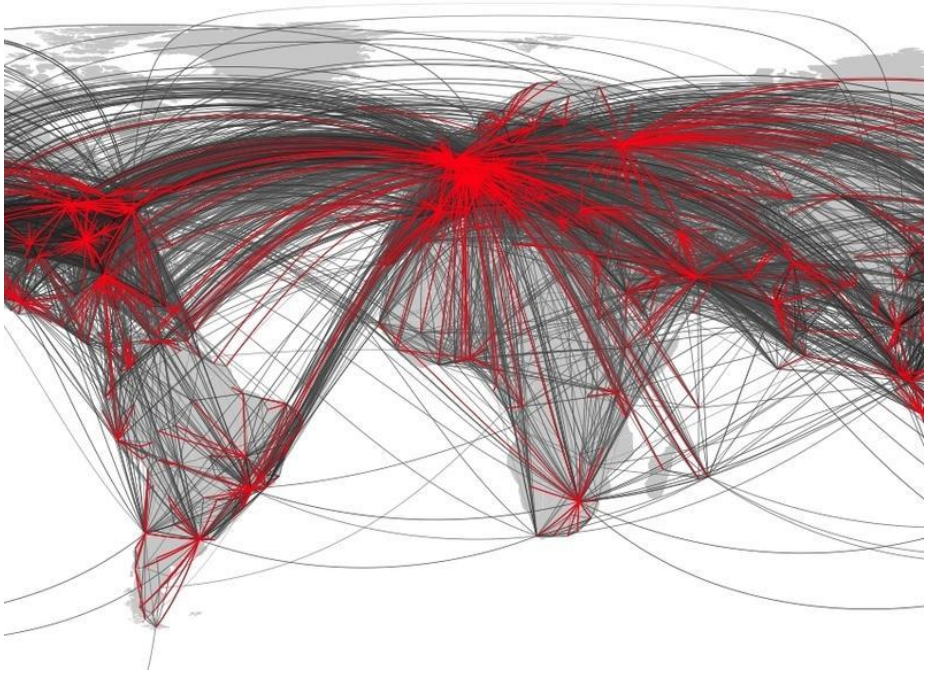


Fonte: disponível em https://www.nextplatform.com/2015/05/11/deep-learning-pioneer-pushing-gpu-neural-network-limits/

Figura 2 - O Facebook como exemplo de rede social virtual.

Fonte: disponível em https://www.theatlantic.com/technology/archive/2012/05/facebook-and-limits-network-effect/327919/

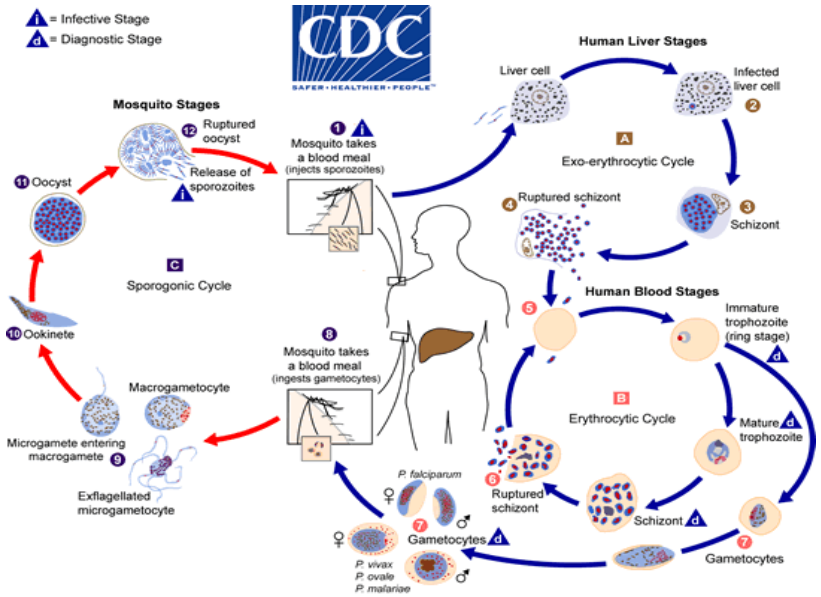
Figura 3 - A rede de transporte aéreo como exemplo de rede social real.



Fonte: disponível em https://phys.org/news/2012-06-network-skeleton.html

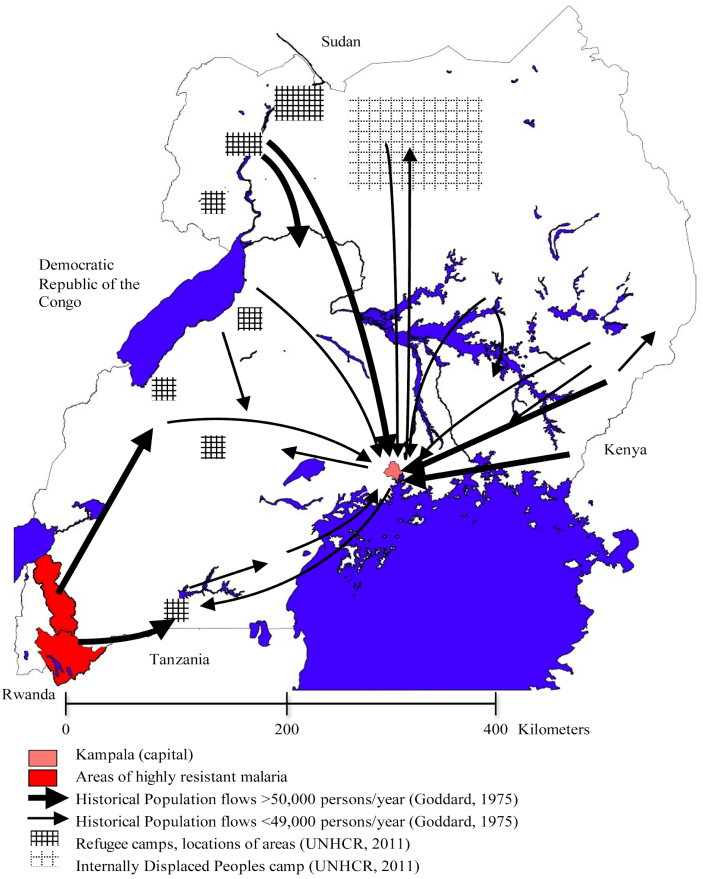
Dentre as variadas redes existentes, destacam-se as redes de ciclos de doenças infecciosas (Figura 4) e de sua dispersão (Figura 5), as quais são frequentemente assumidas como complexas por serem dinâmicas e modificáveis por várias variáveis também dinâmicas e modificáveis (Romero *et al*., 2011).

Figura 4 - Rede do ciclo infeccioso da malária.



Fonte: disponível em https://www.cdc.gov/malaria/about/biology/index.html

Figura 5 - Rede de dispersão da malária.



Fonte: disponível em http://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1001040

No decorrer deste capítulo iremos explicar e caracterizar tipos de redes complexas como redes de discussão, representar matematicamente essas redes e que tipos de padrões elas possuem.

1.1.1 Redes complexas e Redes de Discussão

Na sociedade, o uso de discussões para se alcançar metas é muito difundido em várias áreas com o intuito de se chegar a uma conclusão sobre determinado tema. Por exemplo, em UTIs de hospitais, médicos discutem sobre casos difíceis a fim de definir o melhor diagnóstico e/ou tratamento; na TV, comentaristas de esportes avaliam e discutem sobre o desempenho de atletas e times em jogos de um campeonato; em sala de aula, o professor pode formar grupos de discussão sobre temas diversos e posteriormente avaliar o resultado dessas interações entre os alunos, no qual se aplica na metodologia ativa PBL (QUEIROZ, 2012).

Nestes contextos, formam-se redes complexas de discussão, cujas interações fazem emergir uma nova propriedade, seja uma tomada de decisão terapêutica, seja a conclusão sobre o melhor jogador em campo, seja a emergência de um aprendizado coletivo (Figura 6).

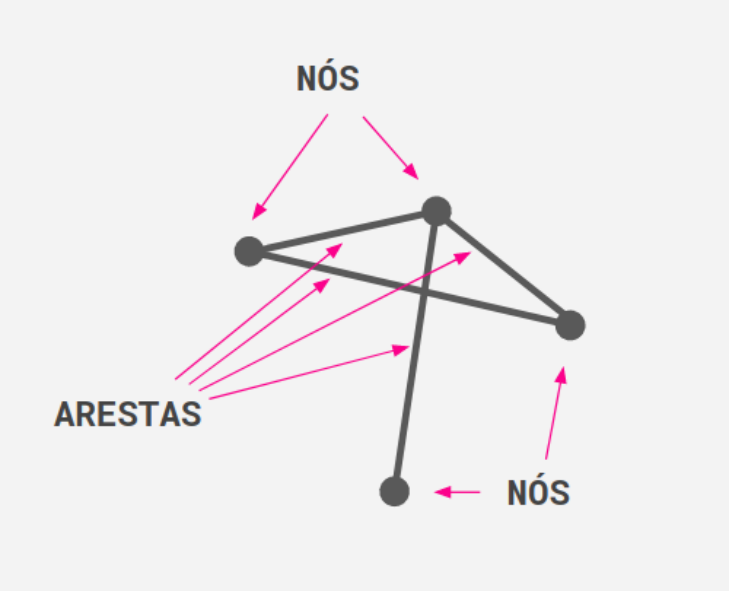
Figura 6 - Rede de discussão semântico-cognitiva em atividade escolar.

Fonte: disponível em https://www.colegiopentagono.com/ensino-fundamental/tutoria/

1.1.2 Redes complexas e Teoria dos Grafos

Matematicamente, o termo "redes complexas" refere-se a um grafo que apresenta uma estrutura topográfica não trivial, formada por um conjunto de vértices (nós) que interagem por meio de arestas (Albert & Barabási, 2002) (Figura 7). Os vértices podem representar indivíduos ou organizações, e as arestas representam as conexões ou as interações entre eles, podendo ser direcionadas e não direcionadas. Existem ainda arestas múltiplas, caracterizadas por mais de uma aresta entre o mesmo par de vértices, e laços, ou seja, quando uma aresta conecta um vértice a si mesmo (Ossada, 2011).

Figura 7 - Estrutura topográfica de uma rede, apresentando nós que interagem por meio de arestas.



Fonte: disponível em https://escoladedados.org/tutoriais/visualizando-redes-do-youtube-com-o-gephi/

As redes podem apresentar diferentes topologias, que dependendo dos mecanismos utilizados determinam sua formação e/ou evolução. Para se quantificar a estrutura topológica de uma rede, diversas medidas têm sido desenvolvidas e utilizadas como parâmetros nesta

quantificação: o Grau do Vértice (*ki*), o qual indica o número de ligações *k* entre o vértice *i* e os demais; o Grau Médio do Vértice, o qual identifica os *hub*s (vértices com maior número de conexões) e também quantificar a densidade dessas conexões; a Distribuição de Grau – *P(k)* –, a qual descreve a probabilidade *P* de um vértice escolhido aleatoriamente ter grau *k*, e assim determina se as conexões entre os vértices são uniformes, o que pode indicar se uma rede pertence a uma categoria de redes complexas com características conhecidas ou não.

O Coeficiente de Agrupamento, o qual indica a probabilidade de dois vértices vizinhos terem um terceiro vértice vizinho a ambos. A média de agrupamento entre todos os vértices da rede, a qual caracteriza globalmente uma rede; o Menor Caminho Médio, o qual mostra o comprimento do caminho que conecta dois vértices *i* e *j* dado pelo número de arestas ao longo deste caminho, caracterizando assim a estrutura interna de uma rede, etc.

Pode-se definir uma rede complexa como estática, i.e. quando as arestas entre os vértices são permanentes ao longo do período estudado, ou dinâmica, i.e. quando as arestas entres os vértices podem se alterar ao longo do período estudado (Metz *et al*., 2007; Ossada, 2011; Carvalho, 2012).

Nesse contexto fica claro que, para se modelar a dinâmica de interações nas redes semântico-cognitivas tutoriais, é preciso utilizar algoritmos que possam reconstruir no tempo a variação da estrutura subjacente dessas redes em sua dinâmica espacial e temporal, e assim permitir analisar as vias de interação no mundo real. Considerando que, em sessões tutoriais, o processo semântico-cognitivas propaga-se ao longo dos elos de uma rede oculta ou não observável, uma maneira de captar a dinâmica dessa rede subjacente é inferir a dinâmica das ligações (arestas) entre os contatos (vértices) dessa rede, ou seja, inferir as arestas entre os vértices da rede semântico-cognitiva subjacente, a qual funciona como um meio físico para o processo de transmissão se espalhar.

Neste sentido, vários algoritmos de inferência da estrutura de redes foram desenvolvidos recentemente (Gomez-Rodriguez *et al*., 2014; Myers & Leskovec, 2010). Algumas dessas abordagens inferem não apenas a estrutura estática das redes (Gomez-Rodriguez *et al*, 2014), como também inferem a probabilidade *a priori* da ativação de arestas nessas redes (Myers & Leskovec, 2010), ou até mesmo as taxas de transmissão através destas arestas.

1.1.3 Padrões em Redes Complexas

Diferentes padrões em redes complexas se relacionam em termos de heterogeneidade (*heterogeneity*), modularidade (*modularity*) e aleatoriedade (*randomness*), o que pode ser observado graficamente (SOLE; VALVERDE, 2004). Na Figura 8, em um extremo do gráfico

encontram-se as redes de padrões regulares, as malhas (*mesh*) e árvores (*tree*), as quais são geralmente redes artificiais que possuem a menor heterogeneidade (por exemplo, o número de conexões que cada nó possui é aproximadamente o mesmo) e menor aleatoriedade (a probabilidade de quaisquer dois nós escolhidos serem aleatoriamente ligados mutuamente é muito baixa ou zero).

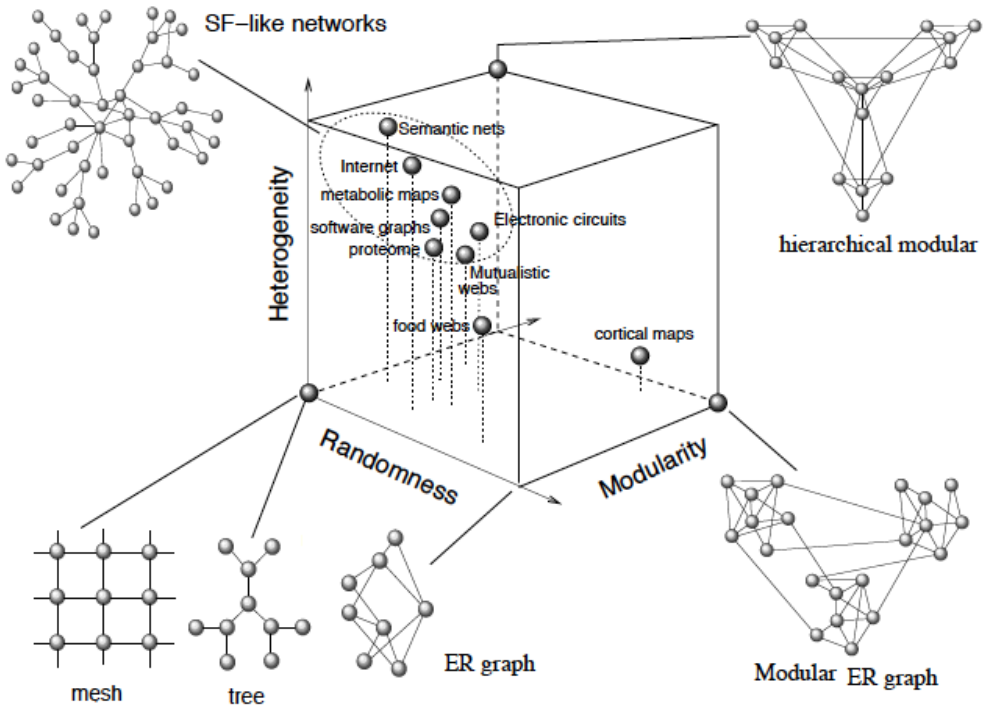
As redes de padrões regulares tendem a ter, na média, caminhos longos, além de um alto agrupamento, pois os nós tendem a ser densamente conectados em grupos. Ainda na Figura 8, em outro extremo encontram-se as redes ER (Erdos-Renyi) aleatórias (*ER graph*), que são geradas iniciando com um conjunto desconectado de nós que são então pareados com uma probabilidade uniforme. Tais redes aleatórias terão baixa heterogeneidade, pois a maioria dos nós tem o mesmo número de conexões, com a Distribuição de Graus sendo uma curva Gaussiana. Gráficos aleatórios construídos após o algoritmo ER têm, em média, caminhos curtos, além de um baixo agrupamento.

A maioria das redes de dispersão do mundo real, no entanto, não tem Distribuição de Graus homogênea, como as redes regulares ou aleatórias têm. O número de conexões que cada nó tem varia muito na maioria das redes, e elas são posicionadas em algum lugar entre redes regulares e aleatórias. Neste sentido, Watts e Strogatz (1998) propuseram um modelo em que as conexões entre os nós em um grafo regular eram religadas com uma certa probabilidade. Os gráficos resultantes estavam entre o regular e o aleatório em sua estrutura, sendo então chamados de redes *small-world* (SW). As redes SW estão muito próximas estruturalmente de muitas redes do mundo real, pois possuem um agrupamento mais alto e quase a mesma média de caminho das redes aleatórias com o mesmo número de nós e arestas. As redes SW geralmente têm alta modularidade, i.e. grupos de nós que são mais densamente conectados juntos do que ao resto da rede. Além disso, a Figura 8 mostra ainda um tipo diferente de uma rede SW (*modular ER graph*), a que é gerada pela reconfiguração das conexões a partir dos módulos interconectados entre esses módulos - uma arquitetura frequentemente encontrada em mapas corticais (redes neurais no cérebro).

Finalmente, a Figura 8 também mostra que há uma classe das chamadas redes sem escala (*SF-like networks*) caracterizadas por uma Distribuição de Graus altamente heterogênea, e que "obedecem a uma função potencial" (Barabasi & Albert, 1999). Elas são chamadas sem escala, porque o zoom em qualquer parte da Distribuição não muda sua forma: há um número pequeno, mas significativo de nós com muitas conexões e há uma cauda de nós com poucas conexões em cada nível de ampliação. Essas redes são típicas da estrutura da WWW (*World Wide Web)*, de mapas semânticos, e de circuitos eletrônicos. Essas estruturas

podem combinar heterogeneidade e aleatoriedade, podem ter baixa ou alta modularidade, sendo que muitas redes SW também são redes sem escala.

Figura 8 - Diferentes tipos de redes em função de seu grau de aleatoriedade, heterogeneidade e modularidade.



Fonte: Sole & Valverde, 2004.

Independentemente de as redes de dispersão do mundo real terem uma ou mais das características acima mencionadas, uma coisa é comum em todas elas: a existência de alguma quantidade de aleatoriedade ou desordem na estrutura de suas conexões. A aleatoriedade nas conexões é um dos ingredientes mais importantes e desejáveis para a funcionalidade adequada ou o desempenho eficiente de sistemas com estrutura de rede subjacente (Strogatz, 2001). Estudos recentes sugerem a importância da aleatoriedade em vários campos, tais como, por exemplo, o processamento de informação no cérebro (Cohen & Tong, 2001), a evolução das capacidades de linguagem de diferentes espécies (Treves, 2005), a evolução da cooperação na teoria dos jogos (Ohtsuki *et al*., 2006), etc.

Entretanto, se por um lado estes estudos enfatizam a importância do grau de aleatoriedade nas redes de dispersão para o funcionamento de diversos sistemas, por outro lado, outros estudos indicam que não se pode desprezar a importância do grau de estruturação (modularidade) ou regularidade (heterogeneidade) nessas mesmas redes, além da força de interação entre seus nós, para um desempenho funcional desses mesmos sistemas (Ravsaz *et al*., 2002; Guimera & Amaral, 2005). Além disso, já foi demonstrado que uma quantidade muito pequena de aleatoriedade tem um impacto drástico no diâmetro de uma rede de dispersão, levando ao fenômeno do *small world* (SW) (Watts & Strogatz, 1998).

Neste contexto, o presente trabalho propõe implementar uma aplicação móvel para registro de interações em uma rede de discussão, geração do grafo-espelho desta rede e medida quantitativa da aleatoriedade presente nesta discussão. Assim, considerando que 0% de aleatoriedade representa uma discussão determinística, e considerando que 100% de aleatoriedade representa uma discussão caótica, podemos através desta ferramenta medir quantitativamente um importante parâmetro da qualidade de uma discussão através do entendimento de sua rede resultante.

1.2 Problema

O uso de técnicas manuais para o acompanhamento de grupos de discussão pode ocasionar em resultados com baixa acurácia e precisão, ocasionando assim uma falha na avaliação do mediador em relação ao desempenho geral do grupo e individual dos membros. Além disso, o fato de não serem utilizadas métricas pré-definidas pode resultar em resultados diferentes a cada aferição, comprometendo assim a confiabilidade dos resultados.

Assim, há a necessidade de um estudo para identificação de métricas de análise eficazes, bem como a criação de uma ferramenta que auxilia no processo de avaliação do mediador.

Desse modo, a definição de parâmetros de verificação para que os participantes sejam avaliados de forma precisa e justa devem ser aplicadas assim como a criação de uma aplicação que automatiza a geração de dados sobre tais interações.

1.3 Justificativa

Considerando a robustez que têm as diversas ferramentas matemáticas utilizadas para a inferência e caracterização de redes complexas, as quais possibilitam a previsão de eventos de variada natureza; considerando que a dinâmica de redes de discussão as caracterizam como redes semântico-cognitivas complexas; e considerando a crescente necessidade de se desenvolver e implementar critérios independentes, acurados e precisos de avaliação dos meios geradores de tomadas de decisão e aprendizado, assim, se fazendo necessárias a definição de métricas de análise de desempenho dos participantes, junto com a criação de uma aplicação que automatiza o processo de registro das interações, o que irá auxiliar na aferição da estrutura da forma como foi aplicada a metodologia.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo deste trabalho é a definição de métricas de análise de desempenho de redes de discussão e a criação de uma aplicação que irá permitir o registro e visualização das interações dessas redes.

1.4.2 Objetivos específicos

Para desenvolvimento desta proposta, será necessário cumprir como objetivos específicos:

- Gerar um grafo das interações do grupo.
- Gerar índice de aleatoriedade da rede a partir da discussão.
- Implementar servidor que fará a disponibilização de rotas para requisições feitas pela aplicação *mobile*.
- Implementar conexão com banco de dados não relacional para armazenamento de informações geradas na aplicação *mobile.*
- Desenvolver aplicação *mobile* que fará o registro das interações.
- Disponibilizar aplicativo *mobile* na loja de aplicativos Play Store

1.5 Estrutura do trabalho

O trabalho consiste em um primeiro capítulo que abrange a introdução, onde foi apresentada a proposta para o problema, além da justificativa e a complexidade do assunto. Em seguida no capítulo dois é apresentado o desenvolvimento do produto, mostrando a prototipação, o processo de construção da aplicação, e a forma como é realizado o cálculo de aleatoriedade e a geração dos grafos. Ainda no mesmo capítulo é apresentado o modelo de negócio, as tecnologias utilizadas e a forma de homologação do *minimum viable product* (MVP). Os resultados e as discussões são apresentados no capítulo três. O capítulo quatro traz as considerações finais acerca do trabalho apresentado.

2 DESENVOLVIMENTO DO PRODUTO

Neste capítulo são mostradas as análises feitas para averiguar a viabilidade do produto, além de apresentarmos o processo de prototipação e desenvolvimento do mesmo.

2.1 Análise de viabilidade

2.1.1 Mercado e Público-alvo

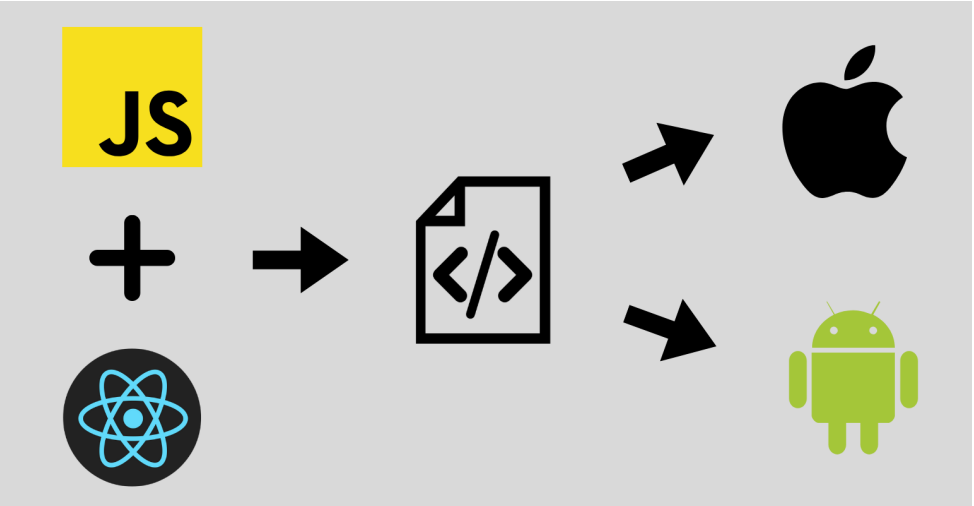
Gerenciar a interação de grupos é um trabalho recorrente de pessoas que lidam com equipes, devido a isto, o nosso público-alvo se trata de qualquer pessoa ou equipe que precise analisar seu grupo baseado nas suas interações durante uma discussão. Além disso, o produto foi direcionado para o mercado *mobile*, devido à facilidade de registro das informações necessárias para avaliação e também em função da grande utilização de dispositivos móveis nas diversas atividades de nossa sociedade (“Retrospectiva 2021: Brasil tem dois dispositivos digitais por habitante, revela pesquisa da FGV”, 2021; “Strategy Analytics: Half the World Owns a Smartphone”, 2021).

2.1.2 Competências técnicas

Para desenvolver o aplicativo é necessário que a equipe tenha conhecimento em 3 áreas diferentes do desenvolvimento de software. Primeiramente, como requisito mínimo, esses programadores deverão ter conhecimentos sólidos de lógica de programação para obter a capacidade de entender cada tecnologia utilizada, além de se fazer necessário o conhecimento da teoria dos grafos.

O conhecimento das linguagens de programação TypeScript e Javascript, sendo utilizadas no *framework* React Native para criação de uma única base de código que nativamente renderiza um aplicativo para as plataformas Android e IOS (“React Native · A framework for building native apps using React”, 2022), são fundamentais para o desenvolvimento da aplicação *mobile*. Junto a esses conhecimentos, a equipe precisará ter uma noção prévia sobre banco de dados *NoSql (not only SQL)*, que é parte dos serviços do Firebase, que foi utilizado pelo aplicativo móvel (“Firebase”, 2022).

Figura 9 - Fluxo criação do aplicativo.



Fonte: Autores, 2022.

Para o desenvolvimento do *backend* da aplicação, a equipe responsável deverá ter conhecimento da linguagem de programação Python, que em conjunto com do *microframework* Flask foram utilizados para a criação de uma *Application Program Interface* (API) (RONACHER, 2014), disponibilizando as rotas necessárias para que o *client* tenha acesso às funcionalidades necessárias para geração dos resultados necessários. Será necessário também o conhecimento prévio em análise estatística para tratamento de dados de entradas vindos do *client,* no qual será feita uma análise. A análise será realizada na linguagem R, devido ao fato dela ser especializada em: manipulação de dados; análise estatística e visualização de dados, sendo bastante utilizada na área do *Data Science* (YEE, 2017). Foi fundamental uma biblioteca que permitisse a execução de códigos R no ambiente do Python, onde o servidor utiliza a linguagem Python para fazer o tratamento dos dados e gerenciar as requisições da API e a linguagem R faz a geração do grafo e os cálculos necessários.

Para a publicação da API, se fazem necessários conhecimentos em plataformas de hospedagem, mais especificamente no Cloud Run, que é oferecido pelo Google, e permite a publicação de API's em vários tipos de linguagens de programação. (“Cloud Run: do contêiner à produção em segundos”, 2022).

Além disso, para a publicação do aplicativo *mobile*, conhecimentos sobre como fazer a publicação na loja digital Google Store se faz necessário, além de também serem essenciais conhecimentos sobre o processo de *build* (construção) da aplicação híbrida para código o nativo que é necessário para fazer a publicação.

2.1.3 Possíveis Dificuldades

Dentre algumas possíveis dificuldades que podemos enfrentar na criação e publicação da aplicação, a primeira, seria a confiabilidade nos resultados que nós iremos produzir, que precisam ser consistentes e confiáveis para podermos atender o problema definido no trabalho. Ademais, como possível problema, temos também o escalonamento da aplicação para diversos usuários, o que se não for feito de maneira correta, pode ocasionar em dificuldade no uso do aplicativo e até mesmo na queda dos servidores responsáveis pelo armazenamento e troca de informações.

2.1.4 Análise financeira

O custo para a autenticação do usuário que é realizada pelo Firebase é de graça caso não ultrapasse 10 mil autenticações por telefone ao mês. Ultrapassando esse valor, a cada 10 mil autenticações será cobrada uma taxa de US $0,06/verificação. (“Firebase Pricing”, 2022).

Segundo a documentação do Firebase, utilizando o Cloud Firestore para armazenar as informações geradas pelo aplicativo, podemos utilizar os seus serviços de forma gratuita baseado nas seguintes condições:

- Armazenar 1 GiB de dados.
- 10 GiB/mês de saída de rede.
- Realizar 600 mil gravações.
- Realizar 1.500 milhões de leituras.
- Realizar 600 mil exclusões.

Ultrapassando esses limites serão cobrados:

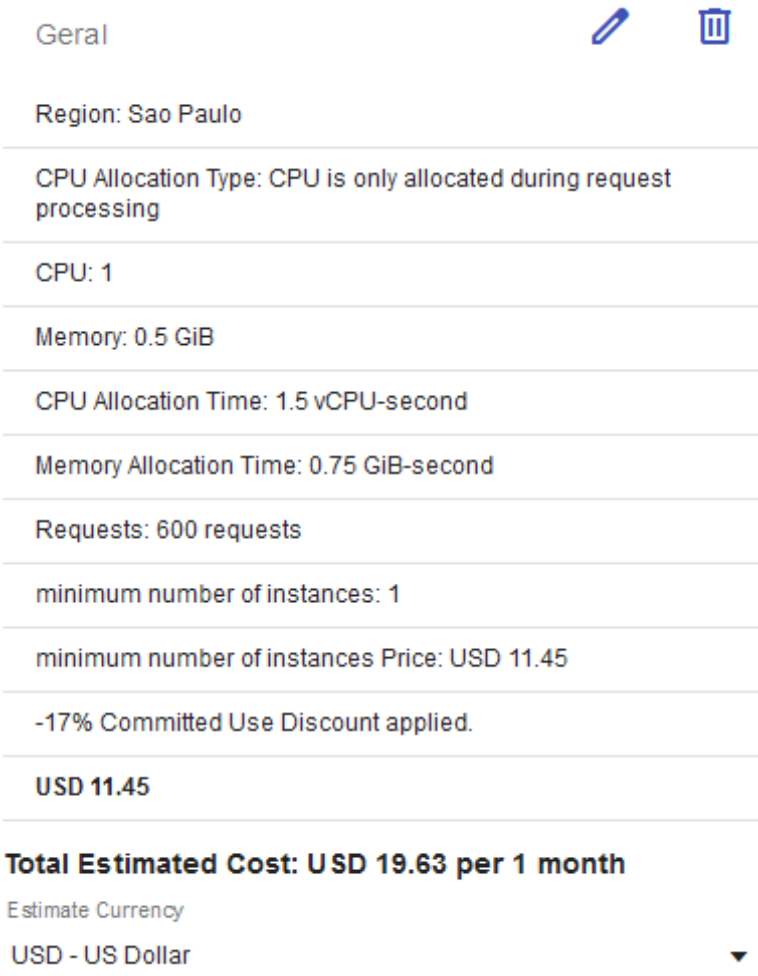
- US $0,108 a cada GiB usado.
- 10 GB a 1 TB US $0,12, 1 a 10 TB US $0,11 e mais de 10 TB US $0,08.
- US $0,135 por 100 mil documentos gravados.
- US $0,045 por 100.000 documentos lidos.
- US $0,015 por 100.000 documentos excluidos.

Não existe custo para hospedar a API no Cloud Run do Google durante os 3 primeiros meses de uso, ou até gastar os US $300 dados como crédito aos novos usuários. Ao finalizar o período grátis, será necessário pagar US $19,63 no plano mais básico, onde obtemos uma máquina contendo:

- 1 unidade de processamento
- a CPU que só irá realizar alterações caso receba requests

- com memória RAM de 0.5 GiB
- com tempo de alocação da CPU de 1.5 vCPU-second
- com tempo de alocação de memória de 0.75 GiB-second
- com 600 requests mensais, pois, supondo que dentro de um mês sejam feitas 300 discussões, como é necessário a realização de 2 requests para criação de visualização da imagem do grafo, então o total de requests seria 600
- com o número mínimo de instâncias sendo 1, que têm um preço mínimo de instância sendo US $11.45. (“Google Cloud Pricing Calculator”, 2022)

Figura 10 - Precificação da cloud.



Fonte: Autores, 2022.

A partir da técnica de análise de ponto de função (HÜLLER et al., 2021) e utilizando a tabela para averiguação das pontuações de cada arquivo de EE (Entrada externa), SE (Saída externa) e CE (Consulta externa), mostrada abaixo.

Tabela 1 - Grau de complexidade e contribuição de pontos de função pelos EE, SE e CE.

Complexidade EE				Complexidade SE e CE				Quantidade em ponto de função			
AR	< 5	5 a 15	> 15	AR	< 6	6 a 19	> 19		EE	SE	CE
< 2	Baixa	Baixa	Média	< 2	Baixa	Baixa	Média	Baixa	3	4	3
2	Baixa	Média	Alta	2 a 3	Baixa	Média	Alta	Média	4	5	4
> 2	Média	Alta	Alta	> 3	Média	Alta	Alta	Alta	6	7	6

Fonte: Adaptado de Hüller (2021).

Foi calculado a quantidade de pontos de função nas duas frentes do desenvolvimento do aplicativo, no qual foram obtidos os seguintes dados:

- 112 pontos de função para o Mobile
- 12 pontos de função para o Backend

Baseado nisto, considerando que 1 PF (Ponto de função) é desenvolvido a cada 8 horas diárias e de acordo com uma pesquisa feita pelo tribunal de contas da união (2019) entre os anos de 2014 e 2017 tem um valor aproximado de R$488,00. Para o desenvolvimento do aplicativo seriam necessários 2 programadores por cerca de 3 meses, com o salário de aproximadamente R$9.100,00, já no *backend* seria necessário 1 programador com o salário de aproximadamente R$1.400,00, assim totalizando R$56.400,00 para o desenvolvimento da aplicação inteira.

Dito isso, baseados nas competências técnicas que a equipe precisaria atender, o custo total para criação da aplicação, incluindo custos dos serviços necessários, publicação na Play Store que é de US $25, e da equipe de desenvolvimento durante 3 meses, seria de R$ 56.844,06 ou US $10.739,07.

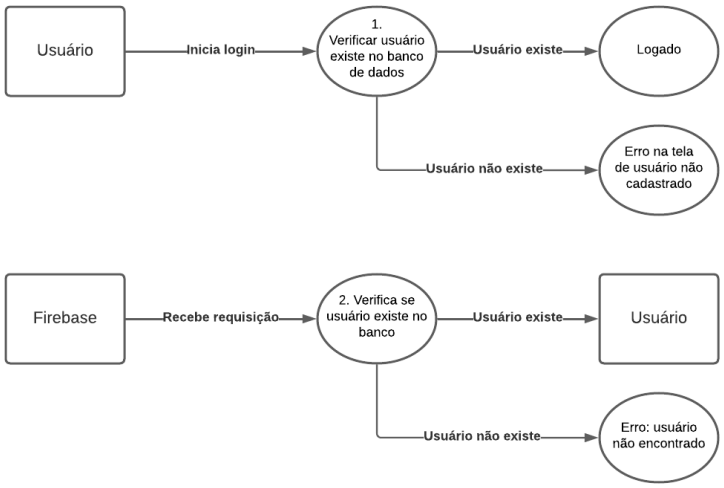
2.2 Prototipação

2.2.1 Requisitos

Nesta seção serão apresentados diagramas e fluxos do aplicativo.

2.2.1.1 Diagrama de Fluxo de Autenticação

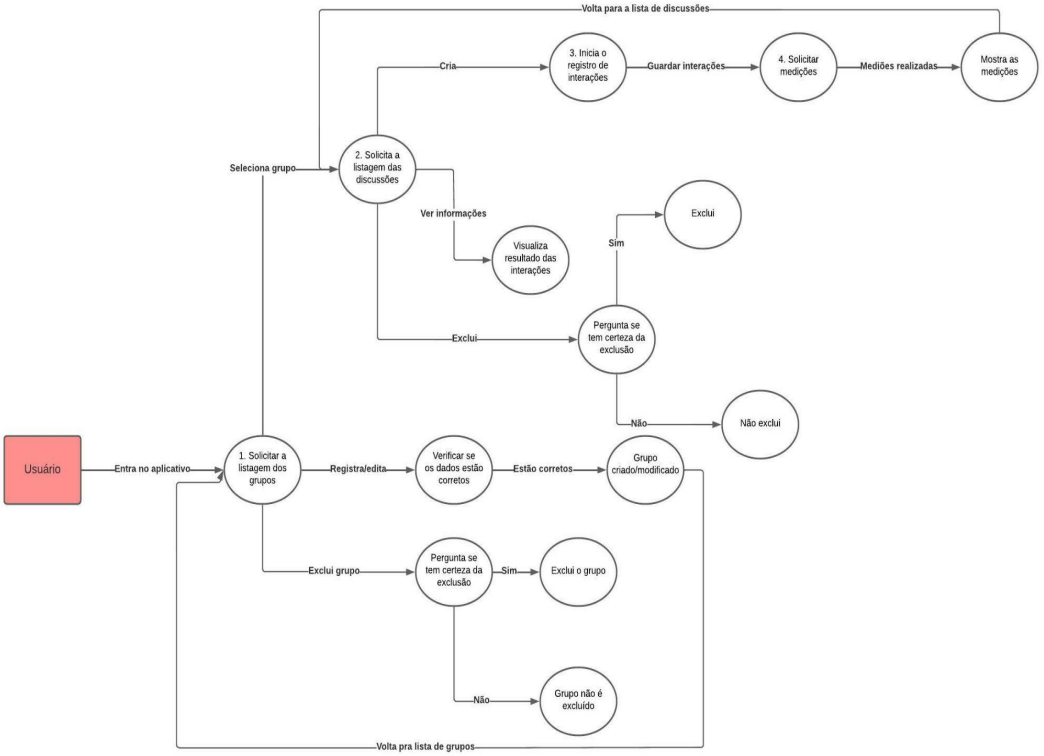
Figura 11 - Fluxo requisição de log in.



Fonte: Autores, 2022.

2.2.1.2 Diagrama de Caso de Uso Estendido

Figura 12 - Fluxo de criação de grupo, discussões e interações.



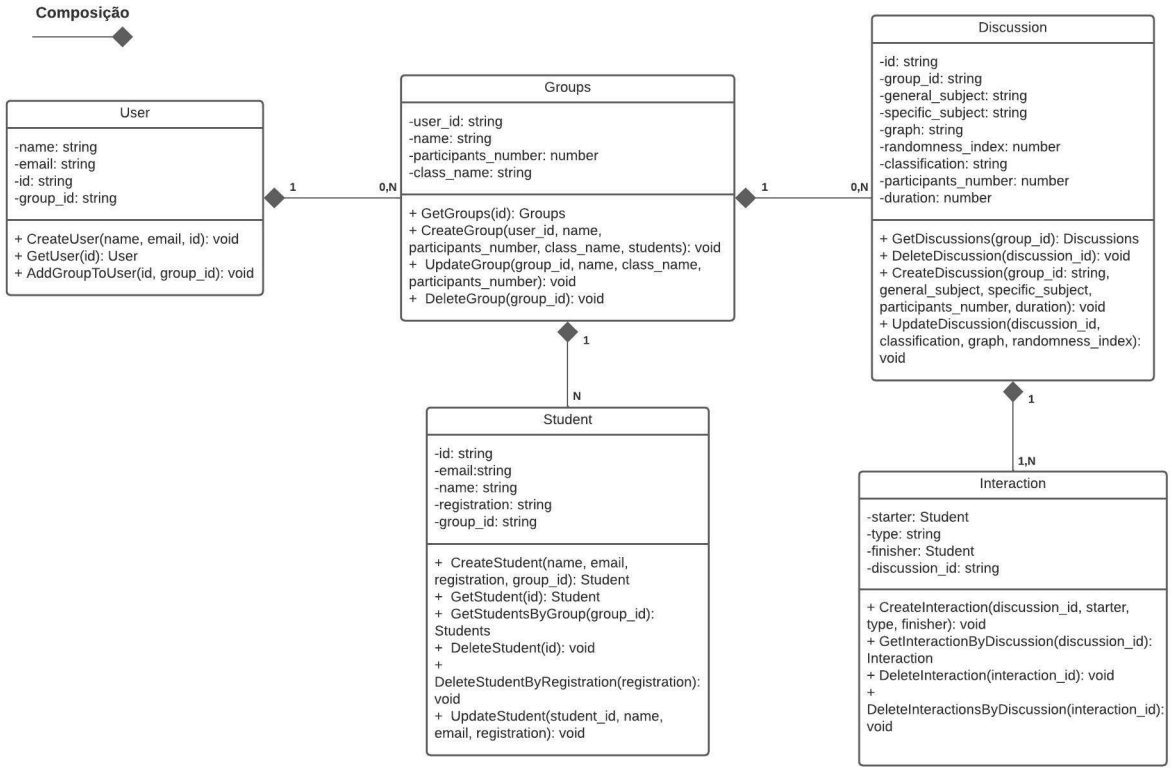
Fonte: Autores, 2022.

Figura 13 - Fluxo requisição de medições.

Fonte: Autores, 2022.

2.2.1.3 Diagrama de Classes

Figura 14 - Entidades.



Fonte: Autoral

2.2.1.4 Arquitetura e Processo de Software

O modelo arquitetural escolhido para o desenvolvimento do produto foi o Cliente-Servidor, no qual o processamento da informação se divide em módulos e processos heterogêneos (KUROSE; ROSS, 2012). O *client* reúne informações enquanto o servidor modifica as informações, onde o *client* é o aplicativo *mobile* e o servidor é a API em Flask feita com Python. O padrão de projeto utilizado no Python foi o *Factory*, onde os módulos que não fazem parte da aplicação principal foram injetados e assim se tornaram parte da mesma (“Factory Method”, 2022).

No aplicativo, o usuário faz o log in, após esse log in ser armazenado localmente no dispositivo, o usuário irá criar um grupo, criar uma discussão e começar a registrar as interações geradas durante aquela discussão.

Quando o *backend* recebe as informações ele inicia o processo de análise, que consiste em primeiro formatar os dados brutos que chegam no servidor, depois cria um grafo com os

dados formatados e inicia seu processo de análise e por fim retorna para o *client* os resultados que são mostrados na tela de resultados após a finalização de uma discussão.

2.2.2 Tecnologias Utilizadas

2.2.2.1 Backend

O desenvolvimento do *backend* foram utilizadas as seguintes ferramentas:

- Python
- Flask
- R
- rpy2
- igraph
- KSgeneral
- Firebase

O Flask é um *microframework* web, *open-source* para a criação de aplicações minimalistas sem a necessidade de compor todas as ferramentas que um *framework* costuma ter (RONACHER, 2014). A linguagem R se trata de uma linguagem bastante utilizada para manipulação de dados, análise estatística e visualização de dados, além de possuir pacotes que permitem abranger ainda mais análises estatísticas (SINGH; GARG, 2019). Na linguagem R serão utilizados dois pacotes, o KSgeneral e o igraph.

O igraph é um pacote *open-source* que viabiliza a criação de grafos dadas uma lista de *strings*, o pacote permite realizar várias manipulações nos grafos gerados, facilitando a obtenção de dados como por exemplo: uma matriz de adjacência e a imagem do grafo. (“igraph – Network analysis software”, 2022). O KSgeneral serve para testar se uma amostra é distribuída como uma Kolmogorov-normal ou não, sendo retornado dois valores em seus testes: o *pvalue* que caso seu valor seja maior que 5% então aceitamos a hipótese nula que significa que a amostra se trata de uma distribuição Kolmogorov-normal e caso seja inferior a 5% então ela não se trata de uma Kolmogorov-normal. E o segundo valor é a estatística, que quanto menor é a porcentagem desse valor, mais próximo de uma distribuição Kolmogorov-normal a amostra que testamos está e quanto maior esse valor mais distante ela se encontra. (DIMITROVA *et al*., 2022).

A biblioteca rpy2 permite que seja executado código R dentro de código Python, utilizamos essa biblioteca para poder manter o Python como linguagem principal no *backend*,

e para que fosse possível comunicar mais facilmente os resultados obtidos no R e utilizá-los no Python (“Documentation for rpy2 — rpy2 3.3.1 documentation”, 2022).

O Firebase se trata de um serviço do Google, que fornece funcionalidades relacionadas a autenticação de usuários, banco de dados não relacionais além de trazer uma facilidade de escalabilidade devido a sua robustez (“Firebase”, 2022).

Com o Firebase, fizemos todo fluxo de autenticação do usuário e também a gravação e armazenamento de dados relacionados aos usuários, grupos, discussões e estudantes.

2.2.2.2 Mobile

Para desenvolvimento do aplicativo *mobile*, foram utilizadas as seguintes tecnologias:

- Javascript
- Typescript
- React Native
- Expo

O Expo é uma ferramenta *open-source* para criação de aplicações multiplataforma, instrumento esse sendo criado com base no React Native visando melhorar a experiência de desenvolvimento, além de trazer diversas ferramentas que ajudam na criação das aplicações (Expo, 2022). O expo foi utilizado na criação da base da aplicação, junto com o pacote para armazenamento dos dados do usuário na memória do seu celular, que foi por sua vez utilizado para validação de log in do usuário. Utilizamos o React Native por sua característica de podermos criar uma única base de código feita em Javascript para às duas principais plataformas *mobile* do mercado, que são o Android e o IOS. (“React Native”, 2022). O Typescript foi escolhido pelo fato de trazer a tipagem estática que o Javascript não possui, o que ajudou no fato de sempre sabermos quais dados são enviados e quais dados vamos receber de volta na aplicação (MICROSOFT, 2015).

2.2.2.3 Deploy

Para *deploy*, que é o processo de disponibilização do servidor e do aplicativo foram utilizadas as seguintes tecnologias:

- Cloud Run
- Play Store

O Cloud Run é uma plataforma de computação gerenciada para a execução de contêineres diretamente na infraestrutura escalonável do Google, no qual pode se utilizar da base de código para se criar uma imagem de container executar no ambiente (“What is Cloud

Run | Cloud Run Documentation", 2022). A Play Store é a loja de aplicativos no qual os usuários de Android têm acesso para poder instalar aplicativos e acessá-los em seus dispositivos ("How Google Play Works", 2022).

2.2.3 Funcionalidades do Produto

2.2.3.1 Mobile

Neste item, iremos descrever todas as funcionalidades que compõem o aplicativo que será usado para a geração dos dados, este possuindo dois principais fluxos. O primeiro sendo relacionado à autenticação dos usuários, composto pelas seguintes funcionalidades:

- log in do usuário na aplicação com validação de email e senha, onde o usuário irá inserir seus dados, passar pelas validações, que caso dê errado, esses erros serão mostrados aos usuários abaixo dos campos de entrada dos dados, e caso dê certo, as informações de log in serão armazenadas na memória do dispositivo e o usuário terá seu log in feito com sucesso.
- Criação de usuário com campos de nome, email, senha, com checagem de email válido, se a senha atende aos requisitos mínimos de segurança, como a inclusão de 8 ou mais caracteres, letras maiúsculas e minúsculas, e a presença de caracteres especiais como: *!@#$%^&*(),.?":{}|<>*.
- Recuperação de senha com envio de link de redefinição para o email cadastrado do usuário.
- *Logout* do usuário com remoção de seus dados armazenados no dispositivo.

Além disso, teremos o fluxo de gerenciamento de grupos e discussões que contemplará as seguintes funcionalidades

- Listagem de grupos associados ao usuário que está logado no sistema
- Criação de grupos novos
- Edição de um grupo já existente
- Remoção de um grupo da lista de grupos do usuário
- Listagem de discussões relacionados a um grupo
- Criação de nova discussão dentro do grupo
- Remoção de discussão da lista de discussões de um grupo

Para registro das interações de alunos nas discussões, é necessário primeiro criar uma nova discussão, em seguida teremos a tela que fará o registro das interações dos alunos, na

qual o usuário irá escolher o inicializador da interação, que ficará com a borda destacada na cor laranja. Os tipos de interações que podem ser escolhidas são:

- C para concordar
- D para discordar
- I para iniciar
- R para responder

A interação escolhida ficará com a borda branca em destaque, e ao final, o usuário selecionará a pessoa com a qual o inicializador interagiu, que aqui chamaremos de finalizador, que ficará destacado com a borda roxa. Ao final da interação, o usuário clicará no botão para guardar a interação, que será gravada no banco de dados. Esse processo se repetirá até a finalização da discussão, onde guardaremos as interações em uma lista que será posteriormente usada para a geração do grafo e do índice de aleatoriedade. A exibição do grafo ocorrerá em uma tela separada, onde será exibida uma imagem contendo o grafo de interações, e o índice de aleatoriedade gerados por tais interações.

2.2.3.2 API

A API será dividida em duas partes, uma responsável por disponibilizar as rotas para o *client* gerar os grafos e outra responsável pela manipulação dos dados e geração dos grafos. As rotas serão divididas em: criação, visualização e exclusão do grafo.

Quando o *backend* receber as informações via requisição de método *POST* ele inicia o processo de análise, que consiste em primeiro formatar os dados brutos que chegam no servidor, e utilizando a biblioteca rpy2 do Python, passamos os dados tratados para o *script* em Python que executa um *script* em R, onde o R irá utilizar o pacote igraph para gerar a imagem do grafo, e o pacote KSgeneral para gerar o valor da estatística com base nos dados recebidos do Python, e ao finalizar a análise retornará para o *script* Python os resultados obtidos, então o Python converte os dados para o formato JSON contendo um *link* para a imagem do grafo e o valor da estatística de aleatoriedade e devolve para o *client*.

2.2.4 Análise de Dados/Uso de Arquivos de Entrada e Saída

O fluxo começa pelo envio dos dados que o *client* manda para a API possui o seguinte formato:

Figura 15 - Dados enviados para o servidor.

```
[
    {
        "discussion_id": "123456",
        "starter":
        {
            "name": "Pessoa1",
            "registration": "1",
            "group_id": "1"
        },
        "type": "Concordou",
        "finisher":
        {
            "name": "Pessoa2",
            "registration": "2",
            "group_id": "1"
        }
    }
]
```

fonte: Autoral

Realizando uma iteração sobre os dados da imagem acima, no qual é verificado se o número de *registration* (o id de cada pessoa) de quem iniciou a discussão não está em um dicionário (estrutura de dados mutável para mapear objetos através de chaves) de interações, caso não esteja presente, é então criado um dicionário utilizando o valor de *registration* como chave, isso faz com que cada pessoa possua apenas um registro no dicionário. Os objetos que cada pessoa adicionada no dicionário irá armazenar serão: o nome da pessoa que possui aquele *registration* e uma lista (estrutura de dados mutável para armazenar um conjunto de objetos) de interações que essa pessoa teve.

Em seguida é verificado se o valor de quem finalizou existe no dicionário criado anteriormente, caso o finalizador não seja nulo, é criado um objeto utilizando como chave o *registration* do finalizador que armazenará as seguintes informações: o nome da pessoa que finalizou, e uma lista com suas interações. Logo em seguida é inserido no objeto de quem iniciou a interação que ela interagiu com o finalizador, é inserido então no campo de interações do inicializador uma tupla (estrutura de dados imutável para armazenar uma pequena quantidade de objetos) (“Built-in Types — Python 3.8.1rc1 documentation”, 2019), onde no primeiro índice contém o *registration* do finalizador e no segundo índice contém o nome do finalizador, como podemos ver na imagem abaixo:

Figura 16 - Dados tratados.

fonte: Autoral

Caso o valor correspondente ao finalizador não exista no dado (e caso exista, porém ele seja nulo) então é interpretado que quem iniciou a discussão apenas iniciou um assunto e não interagiu com ninguém, essa pessoa ficará salva como se ela tivesse interagido com ela mesma. Por fim, após realizar a iteração sob todos os dados, é iniciada uma nova iteração sob a lista de chaves do dicionário criado anteriormente. E sob essa lista de chaves é iniciada uma nova iteração sob a lista de iterações de cada chave armazenada. A partir daí será armazenado em uma lista uma tupla contendo as interações entre todas as pessoas, onde no primeiro índice está o da pessoa que iniciou a interação, e no segundo está o da pessoa com quem ela interage, como pode ser visto abaixo:

Figura 17 - Lista de interações.

fonte: Autoral

A partir da lista gerada, é criado o grafo utilizando a biblioteca igraph, onde a partir do grafo criado é obtido a matriz adjacente das interações e a matriz é transformada em uma lista de apenas uma dimensão que utilizaremos para iniciarmos o teste de Kolmogorov-Smirnov utilizando o pacote KSgeneral (DIMITROVA *et al.*, 2022). Passamos essa lista para a função *disc_ks_test* que irá realizar o teste, onde é verificado se nossa amostra se trata de uma distribuição normal. O primeiro valor retornado pela função é referente a estatística do teste, quanto menor o valor da estatística mais próximo o nosso conjunto de dados se assemelha de uma distribuição normal, e quanto maior, mais distante de uma distribuição normal ele está. Já

o *pvalue* responde se a nossa distribuição se trata ou não de uma distribuição Kolmogorov-normal, caso seu valor seja acima de 5% significa que os dados são de uma distribuição Kolmogorov-normal, caso esteja abaixo de 5% significa que os dados não são de uma distribuição Kolmogorov-normal.

O índice de aleatoriedade será retornado para a API utilizando a biblioteca rpy2, sendo executada como sub-processo do servidor, para o qual passamos a lista de interações que é utilizada no pacote KSgeneral o qual retornará o resultado da estatística que é armazenada em uma variável, onde o valor será subtraído por 1 devido ao fato de que quanto menor a porcentagem, mais aleatória a amostra é, logo se fazemos essa subtração, obtemos o índice de aleatoriedade, fazendo assim com que possamos mandar para o aplicativo a porcentagem do nível de aleatoriedade do grafo gerado.

Quanto a rota que retorna esses dados analisados para o *client*, é enviado um JSON (a notação de objetos estilo Javascript que permite armazenar objetos através de chaves) (“JSON”, 2022) contendo as informações que serão mostradas, sendo elas:

- O parâmetro *graph* é um link para a imagem do grafo gerado pela lista de interações
- O *id* é o nome do grafo que foi armazenado no servidor
- O campo *random_percent* que é referente ao resultado da análise do teste de Kolmogorov Smirnov onde indica o nível de aleatoriedade da interação.

2.2.5 Etapa de teste

Para o teste do aplicativo, foi utilizado uma planilha contendo as interações realizadas em sala de aula por um professor do curso de Medicina do Cesupa, as interações foram registradas no aplicativo assim como ocorre em campo.

Ao total foram 9 participantes, onde os que estão destacados em vermelho representam os alunos que não direcionaram suas falas a ninguém, porém interagiram com a turma durante a conversa, iniciando um novo assunto, como mostra a Tabela abaixo.

Tabela 2 - Tabela adjacência das interações.

Registration	names/ names	P1	P2	P3	P4	P5	P6	P7	P8	P9
1	P1	0	0	0	0	0	0	0	0	0
2	P2	0	0	0	0	1	0	0	0	2
3	P3	1	1	0	0	1	0	0	0	0
4	P4	0	0	0	0	1	0	0	0	0
5	P5	0	0	0	2	0	0	0	0	1
6	P6	0	0	0	0	0	0	0	0	0
7	P7	0	0	0	1	1	0	0	0	1
8	P8	1	0	0	0	0	0	0	0	0
9	P9	0	0	0	0	0	0	0	0	0

Fonte: Autores, 2022.

Foi criado no aplicativo um grupo contendo todos os 9 alunos da Tabela 1. Com o grupo criado foi iniciada uma nova discussão, onde foi registrado que P1 iniciou a discussão, porém não direcionou sua fala a ninguém. Já a P2 interagiu com P5, portanto foi registrado que ele “Concordou” com P5. Ao finalizar de registrar todas as interações da planilha, clicamos no botão de “Finalizar Discussão” e confirmamos. Então o aplicativo envia as informações das interações para o servidor, e o mesmo retorna com a imagem gerada e a porcentagem do índice de aleatoriedade. A discussão realizada fica salva na lista de discussões, e quando for necessário visualizar a informação da discussão novamente, basta clicar em “Ver Informações” e os dados serão mostrados novamente.

2.2.6 Comercialização do Produto

O total gasto para a criação do produto seria de R$ 56.844,06 ou US $10.739,07 como citado no item 2.1.4. Dito isso, a Play Store, loja de aplicativos do Android, cobra uma taxa de 15% a para serviços de renovação de assinatura automáticos (“Taxas de serviço - Ajuda do Play Console”, 2022), além de também precisamos incluir os custos de supervisão do produto por parte dos programadores, onde dois programadores teriam foco no mobile recebendo R$ 9.100,00 por mês e um programador com foco no *backend,* recebendo R$ 1.400,00. Além disso, podemos também considerar que a manutenção pode ser feita por grupos de pesquisa de dentro da própria instituição o que diminuiria consideravelmente o custo de manutenção da aplicação.

Caso a manutenção não seja feita por grupos de pesquisa, a aplicação, ao utilizar o modelo de assinatura mensal, ofertando o aplicativo a US $5 (R$ 27,05), contando com os descontos Play Store, o custo inicial do aplicativo juntamente com a sua manutenção no

primeiro ano, seria pago ao conseguir 2550 usuários assinantes, tornando assim o aplicativo viável.

2.2.7 Produtos Correlatos

Existem alguns produtos que se assemelham na parte de organização e condução de debates, como:

- *Disqus*
    - Este aplicativo, apresenta como funcionalidade principal a facilitação e acompanhamento de um debate, apresentando um funcionamento similar ao de uma rede social.

Figura 18 - Logo do Disqus.

Fonte: disponível em https://disqus.com/

- *Scorp*
    - O aplicativo tem como objetivo ter conversas sobre os diferentes assuntos, mostrando no formato de rede social, trazendo a funcionalidade de poder comentar sobre os diversos temas postados por usuários da rede.

Figura 19 - Logo do Scorp.

Fonte: disponível em https://www.scorpapp.com/

- *Quinto*
    - É um aplicativo que vem com a proposta de tornar o debate organizado, com foco em dar voz ao público, tendo como um de seus diferenciais o conteúdo

postado, sendo escrito por jornalistas internos com foco em trazer o máximo de entendimento sobre o assunto.

Figura 20 - Logo do Quinto.

Fonte: disponível em
https://www.projetodraft.com/o-quinto-e-um-aplicativo-que-da-voz-ao-debate-de-maneira-organizada/

Entretanto, todos os aplicativos acima não entregam uma análise de grupo, não mostram as interações por meio de grafos e não possuem uma forma de comprovação da efetividade do debate.

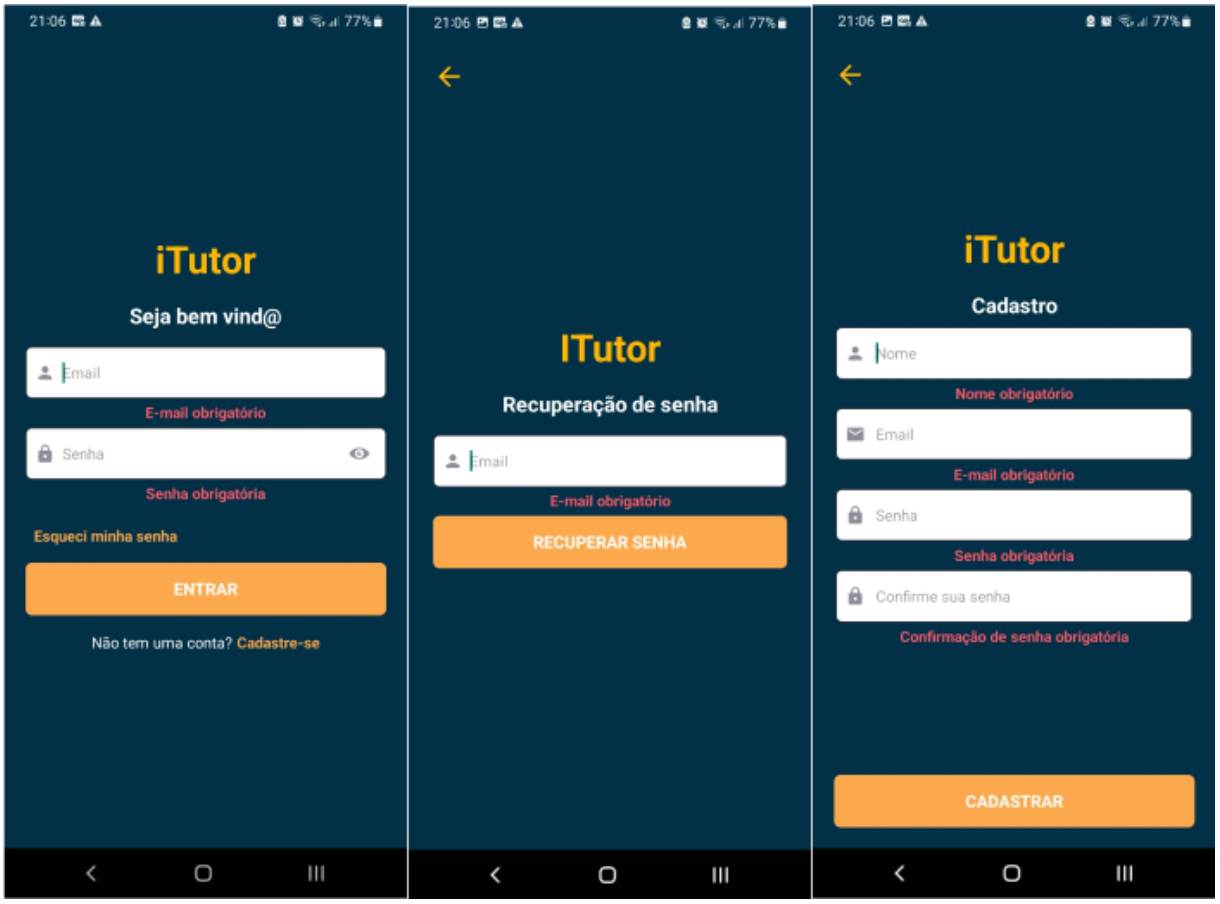
3 RESULTADOS E DISCUSSÃO

3.1 Resultados

O protótipo de alta fidelidade foi desenvolvido com o Figma, site gratuito para desenvolvimento de protótipos no navegador (FIGMA, 2022), onde foram criadas 11 telas que representam o fluxo completo do aplicativo. Como já dito no item 2.2.3.1, o aplicativo possui 2 fluxos principais, o primeiro que é relacionado à autenticação dos usuários, e o outro, sendo relacionado ao gerenciamento de grupos e discussões de um usuário logado.

Como podemos ver na imagem abaixo, o fluxo de autenticação tem log in, recuperação de senha e cadastro do usuário, funcionais e com diversas validações para verificar se os campos estão preenchidos com os dados corretos, como por exemplo, na tela de cadastro, onde verificamos se o email é válido, e também onde verificaremos se às duas senhas são iguais e atendem aos requisitos mínimos de segurança citados anteriormente.

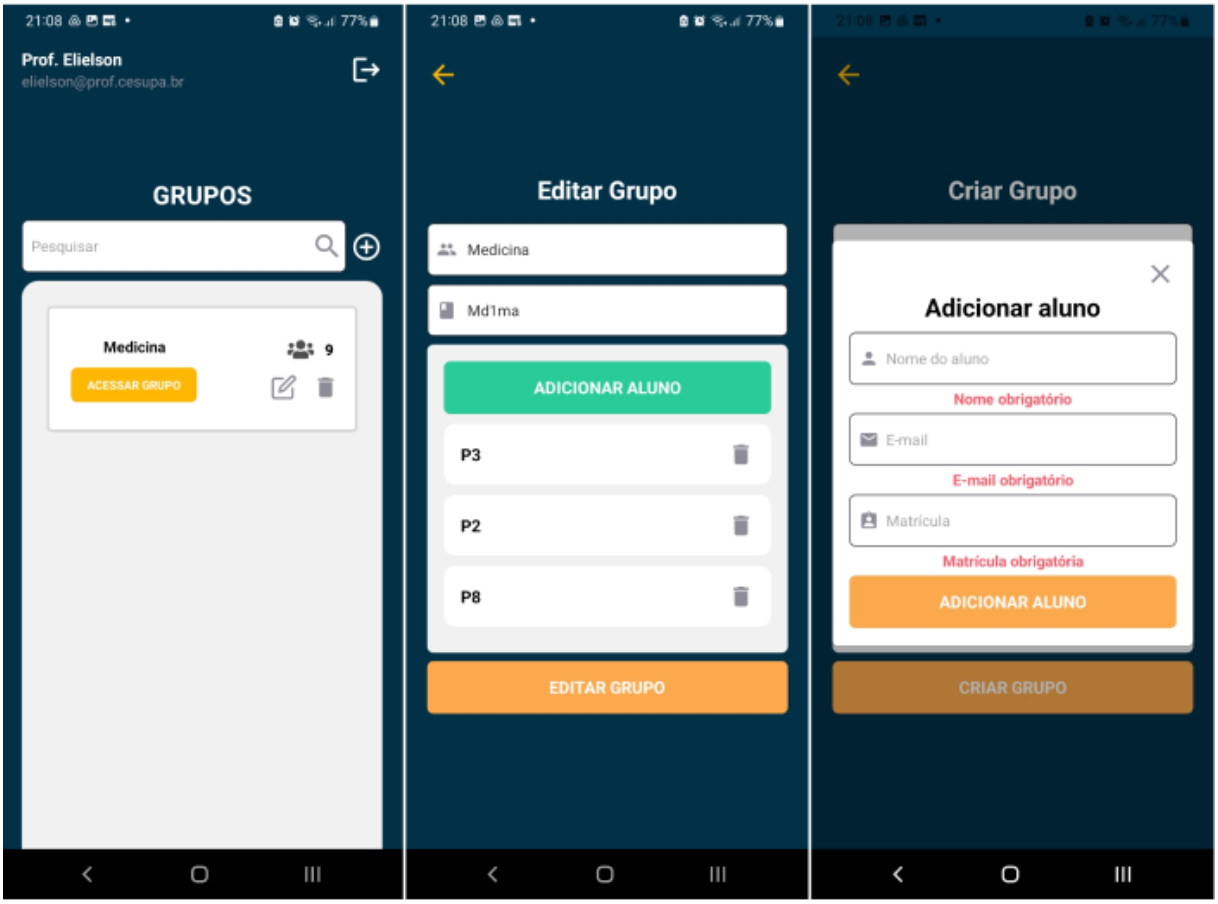
Figura 21 - Fluxo de login, cadastro e recuperação de senha.



Fonte: Autores, 2022.

No fluxo de gerenciamento dos grupos, nós fazemos a listagem, com pesquisa por nome, a criação e edição dos grupos, onde podemos adicionar e remover alunos.

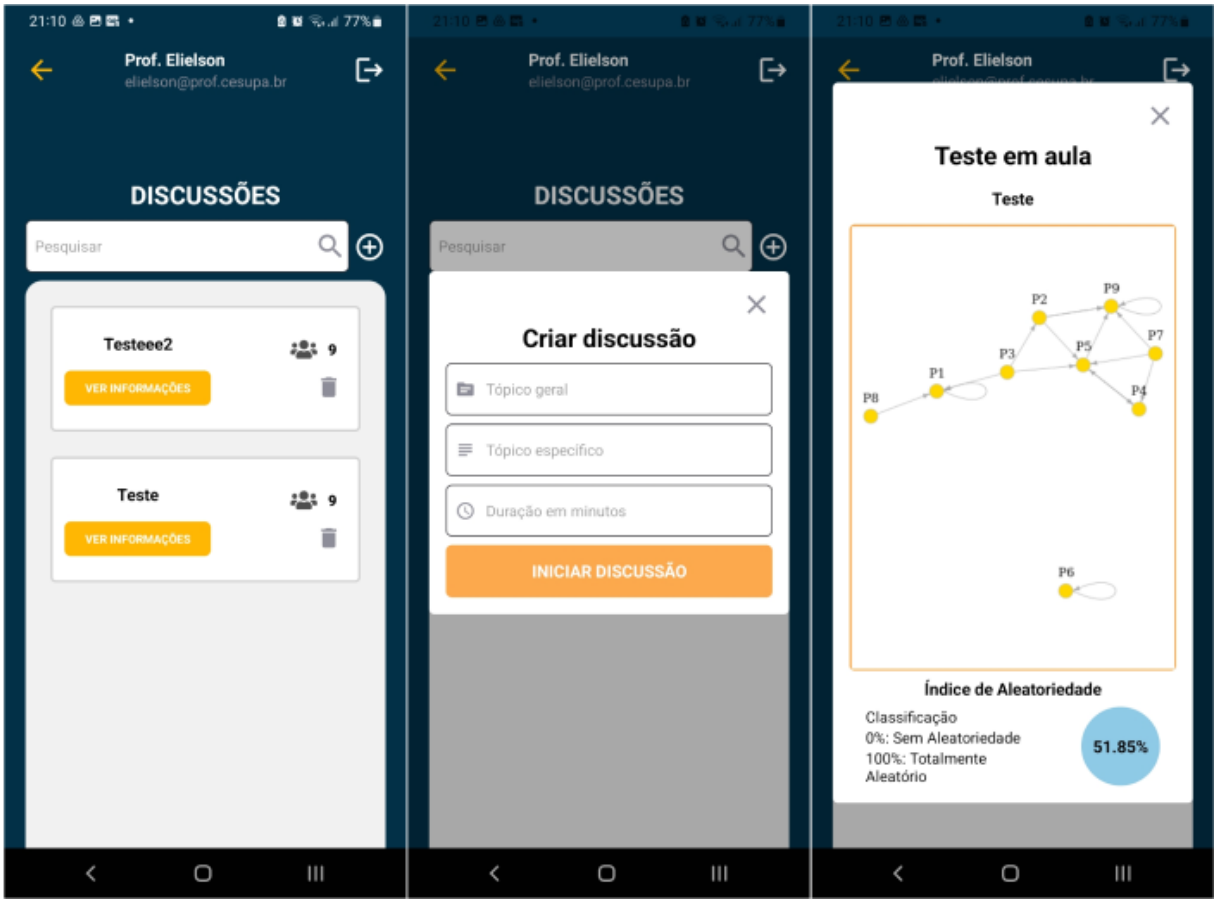
Figura 22 - Listagem, criação e edição de grupos.



Fonte: Autores, 2022.

Como é visto na imagem acima, na criação e na modificação do grupo é reaproveitado os campos, pois os dados são os mesmos. Abaixo, é mostrado o fluxo de discussão, no qual nós fazemos a listagem, também com a possibilidade de pesquisar um grupo, além de mostrarmos os resultados de discussões anteriores dentro de um modal. Para criação de uma nova discussão, mostramos um modal com alguns campos que também têm uma validação em tempo real.

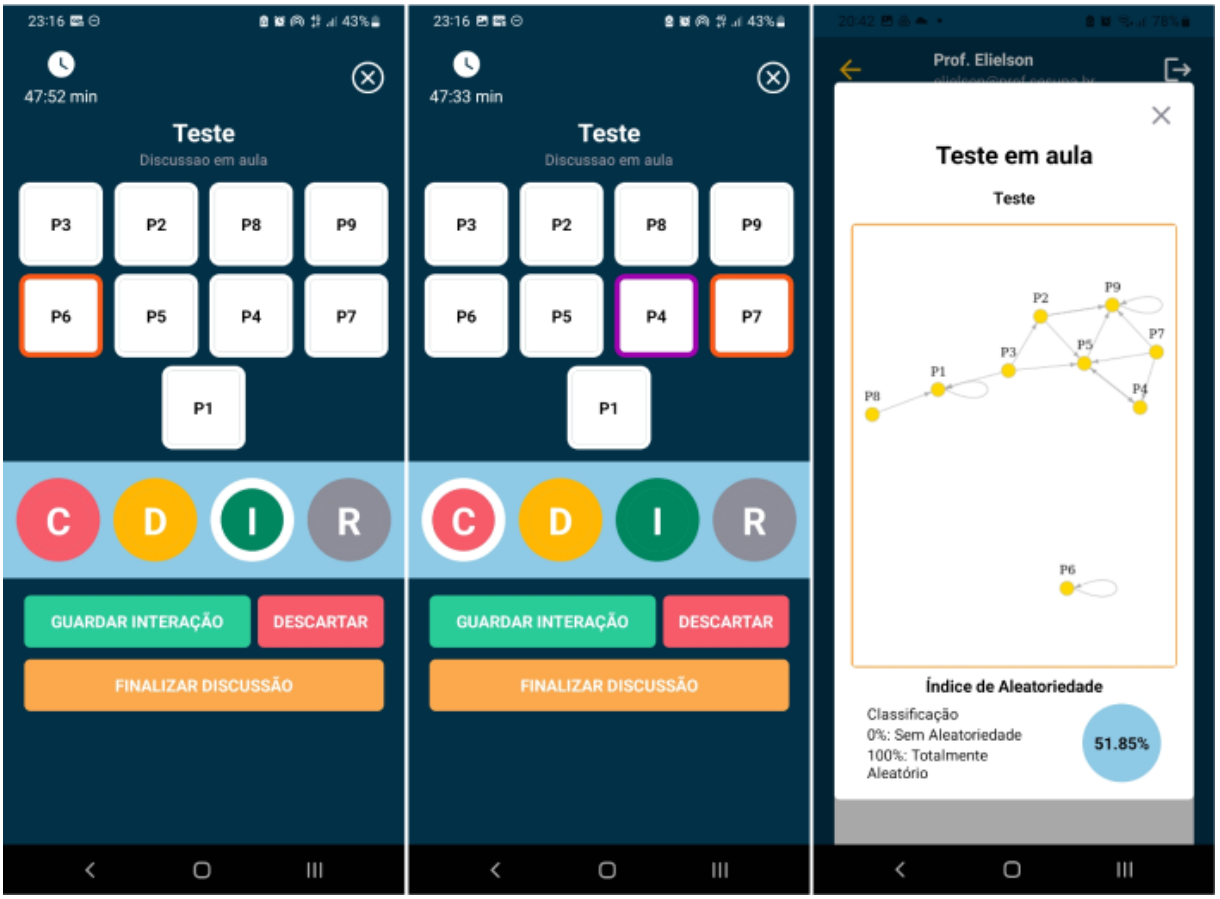
Figura 23 - Listagem, criação e visualização de resultados da discussão.



Fonte: Autores, 2022.

Por último temos a tela de criação das interações dos alunos, onde clicando nos botões referentes aos nomes de cada aluno e escolhendo o tipo de interação que foi realizada nós fazemos o registro das suas interações no banco de dados. Ao finalizar uma discussão, geramos o grafo das interações e mostramos o índice de aleatoriedade na tela de resultados.

Figura 24 - Criando interação.



Fonte: Autores, 2022.

Em paralelo às funcionalidades desenvolvidas no aplicativo, foram implementadas no servidor as rotas baseadas no modelo CRUD (porém sem o conceito *Update*) para a criação, leitura e exclusão dos grafos. Abaixo podemos ver a rota para criar o grafo utilizando o método *POST*:

Figura 25 - Rota POST para iniciar a análise.

```
@bp.route("/graph", methods=["POST"])
def graph_generate():
    data = request.json
    current_app.itutor.Reset()
    if data:
        current_app.itutor.FormatData(data)
    current_app.itutor.GenerateRandomName()
    current_app.itutor.GenerateGraph()
    current_app.itutor.CreatePlotComparison()
    porcento = f"{current_app.itutor.random_percent*100:.2f}%"
    return jsonify({"graph": f"{request.base_url}/{current_app.itutor.random_name}",
                    "random_percent": porcento,
                    "id": current_app.itutor.random_name}),\
                    200
```

Fonte: Autores, 2022.

A imagem acima retorna no campo *graph* um link para a rota da imagem abaixo, que é responsável pela visualização da imagem do grafo:

Figura 26 - Rota GET para retorno do grafo.

```python
@bp.route("/graph/<id>", methods=["GET"])
def graph_image(id):
    return send_file(f"{current_app.itutor.PATH_GRAPH_IMAGE.format(name=id)}", mimetype='image/png')
```

Fonte: Autores, 2022.

E para deletar a imagem gerada basta enviar uma requisição do tipo *DELETE* contendo o *id* do grafo, para a rota da imagem abaixo:

Figura 27 - Rota DELETE para exclusão de imagem.

```python
@bp.route("/graph/<id>", methods=["DELETE"])
def graph_delete(id):
    try:
        os.remove(current_app.itutor.PATH_GRAPH_IMAGE.format(name=id))
        os.remove(current_app.itutor.PATH_CURVE_IMAGE.format(name=id)+".png")
        return '', 204
    except Exception as e:
        print(e)
        return '', 404
```

Fonte: Autores, 2022.

Após uma discussão ser finalizada pelo *client*, é enviado um *array* de interações, que ao ser tratado e enviado para a biblioteca igraph gera a matriz adjacência como na imagem abaixo:

Figura 28 - Matriz de adjacência gerada pelo igraph.

	Renway	Ana	Josi	Amanda	Leo	Ellys	Mayra	Rafa	Blenda
Renway	1	0	0	0	0	0	0	0	0
Ana	0	0	1	2	0	0	0	0	0
Josi	0	0	0	1	0	2	0	0	0
Amanda	0	0	0	1	0	0	0	0	0
Leo	1	1	1	0	0	0	0	0	0
Ellys	0	0	1	0	0	0	0	0	0
Mayra	0	0	0	0	0	0	1	0	0
Rafa	0	0	1	1	0	1	0	0	0
Blenda	1	0	0	0	0	0	0	0	0

Fonte: Autores, 2022.

Ao realizar a conversão de matriz para um vetor de uma dimensão, temos o vetor da imagem abaixo:

Figura 29 - Vetor gerado pela conversão da matriz.

```
VETOR 1D: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, '...']
```

Fonte: Autores, 2022.

Inserindo o vetor acima para ser analisado pelo teste de Kolmogorov-Smirnov do pacote KSgeneral temos o resultado abaixo:

Figura 30 - Resultado do teste de Kolmogorov-Smirnov.

```
Statistics Recebida: 0.48148148148148145
```

Fonte: Autores, 2022.

E finalizando essa etapa, a rota de geração do grafo irá retornar o JSON abaixo.

Figura 31 - Resultado da requisição.

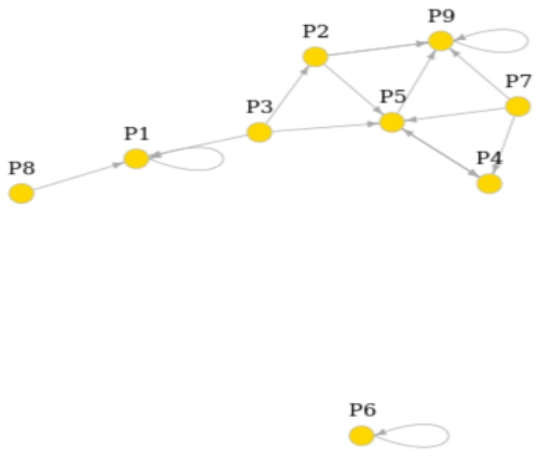

```
{
  "graph": "http://itutor-service-zeifba777q-uc.a.run.app/graph/123456",
  "id": "123456",
  "random_percent": "51.85%"
}
```


Fonte: Autores, 2022.

O parâmetro *graph* é um link para a imagem do grafo gerado pela lista de interações, o *id* é o nome do grafo que foi armazenado no servidor e o campo *random_percent* é referente ao resultado da análise do teste de Kolmogorov-Smirnov onde é indicado o nível de aleatoriedade da interação.

Com os dados acima analisados o resultado do grafo gerado será o da imagem abaixo:

Figura 32 - Grafo da rede de interações gerado pela requisição.



Fonte: Autores, 2022.

Além do aplicativo funcional, nós obtivemos como resultado o servidor que é capaz de receber as informações necessárias, processá-las e devolver os resultados que nós necessitamos para mostrar no aplicativo.

A publicação do aplicativo foi efetuada na Google Play Store, no qual foi gerada a versão de *build* do Android. Além disso, foram coletados os emails dos desenvolvedores e orientadores do projeto para que pudéssemos fazer a instalação das versões de teste antes da publicação oficial.

3.2 Discussão

Ao testar uma aplicação feita nativamente para a plataforma Android, em comparação com outra que foi feita com o *framework* React Native, os usuários não notaram grandes diferenças além das animações de transição no uso dos dois *apps*, fazendo assim com que não se importassem em usar o aplicativo feito com o *framework* (DANIELSSON; FRÖBERG; BERGLUND, 2016). De acordo com uma pesquisa realizada no 3 trimestre do ano pela Developers Nation (2022), existem cerca de 33,6 milhões de desenvolvedores no mundo e aqueles que utilizam Javascript como ferramenta principal somam 19,6 milhões. Isso significa que 58% dos desenvolvedores do mundo já tiveram contato com a tecnologia, e segundo o Stack Overflow (2022) um dos maiores sites de perguntas e respostas sobre programação do mundo, o Javascript é a tecnologia mais popular em 2022, fazendo com que o uso da linguagem na criação do aplicativo seja de grande valia quando se fala de custo e disponibilidade de mão de obra para manutenção.

Ao longo do desenvolvimento da estrutura do *backend*, nos deparamos com algumas dificuldades encontradas ao gerar a análise estatística e visualização do grafo utilizando o Python. Inicialmente começamos a analisar os dados das interações utilizando a biblioteca scipy do Python, que dentro dela possui um método para realizar o cálculo de Kolmogorov-Smirnov, porém, ao realizar diversos testes de interações, sempre obtemos o mesmo valor do índice de aleatoriedade: 50%. Investigando mais a fundo, foi percebido que dependendo da quantidade de valores 0 dentro do conjunto de dados que passamos para o teste, o teste puxava o resultado para próximo dos 50%.

Então foram realizados diversos tratamentos nos dados, como: eliminar os 0 das interações; manter apenas um 0 por pessoa. Porém, essas manipulações afetam a precisão do cálculo e fizeram com que os resultados não fossem confiáveis. A partir desse ponto foi investigado se havia alguma forma de utilizar os dados que tínhamos (valores discretos) e

realizar o teste de Kolmogorov-Smirnov em Python com resultados positivos, porém não foi encontrado nada em Python, mas, encontramos uma solução em R, um pacote chamado KSgeneral, que realizava o cálculo de Kolmogorov-Smirnov a partir de uma amostra de valores discretos e retornava resultados bastante satisfatórios. Em relação a imagem do grafo gerado, ao criar a imagem utilizando o igraph no Python, a imagem gerada não ficava legível o suficiente, dependendo da quantidade de interações que o grafo tinha, ficava impossível de entender quem interagiu com quem. Então, após conseguir utilizar o R para gerar a estatística, passamos a geração da imagem do grafo também para o R, que cria uma imagem mais legível mesmo com mais interações.

4 CONSIDERAÇÕES FINAIS

O mercado de desenvolvimento de aplicações móveis cresce cada vez mais, devido a grande quantidade de smartphones que estão sendo utilizados no mundo (“Strategy Analytics: Half the World Owns a Smartphone”, 2021). Assim, se faz necessário o uso de ferramentas que utilizam conceitos que mantém o produto estável e escalável sem necessitar de tantas atualizações recorrentes. Dessa forma, o Cientista da Computação com sua formação que engloba cada aspecto do desenvolvimento de software e o estudo da teoria dos grafos, é capaz de criar um produto contendo o que é necessário para um aplicativo publicável e funcional. Em teoria dos grafos temos estudos que em redes complexas é possível medir o benefício da aleatoriedade das conexões dos nós dos grafos. E redes complexas estão presentes no nosso cotidiano, inclusive em conversas que temos ao longo do dia, e em grupos de discussão, que geram redes de interações.

Diante disso, com a necessidade de avaliar o quanto de aleatoriedade houve nas interações de uma rede de discussão, para que ela seja classificada como uma rede gerada a partir de um bom grau de aleatoriedade, se fez necessário o desenvolvimento de um aplicativo móvel que pudesse realizar os registros dessas interações, como também a organização dos grupos em que essas discussões são feitas.

Este trabalho de curso apresentou o desenvolvimento de um aplicativo *mobile* que foi desenvolvido utilizando o *framework* React Native, com a linguagem Typescript, uma vez que o framework React Native apresenta facilidade para o desenvolvimento de aplicações *mobile* modernas e com boa performance em celulares Android que foi nosso foco inicial. O aplicativo foi disponibilizado para utilização na Play Store[1].

Como também apresenta o desenvolvimento de uma API, feita em Python utilizando a biblioteca Flask que permite a criação de rotas para serem acessadas pelo aplicativo, o Flask permite a criação de um servidor com poucas linhas de códigos, e por ser feito em Python é possível realizar tratamentos de dados de forma fácil e otimizada. O Python permite inclusive a execução de códigos da linguagem R ao utilizar a biblioteca rpy2, que escolhemos utilizar para poder gerar o grafo e a análise estatística na linguagem R que é popularmente conhecida por ser a linguagem mais utilizada por *Data Scientist*, em conjunto com Python. No R nós geramos o grafo que será visualizado pela tela de resultado do aplicativo, e também geramos o cálculo de análise de aleatoriedade gerada pelo pacote KSgeneral do R.

O Aplicativo e a API foram construídos utilizando vários conceitos do desenvolvimento *Mobile* e Web. Iniciamos com a prototipação, passamos para a escolha das

[1] Link para download: https://play.google.com/store/apps/details?id=com.itutor

tecnologias e começamos a construção do MVP e depois do aplicativo e da API. Nos deparamos com algumas dificuldades ao longo do desenvolvimento, implementamos novas tecnologias e conseguimos obter um bom resultado com um aplicativo *mobile* sendo o *client* e uma API sendo o servidor, onde nela é possível tratar os dados, realizar as análises e então retornar para o aplicativo num fluxo que pode ser visto pela arquitetura Cliente-Servidor.

Além de termos alcançado os nossos objetivos, propomos como trabalhos futuros e funcionalidades novas os seguintes itens:

- Fazer a publicação do aplicativo na Apple Store
- Criação de métricas e funcionalidades para avaliação de desempenho individual dos alunos dentro das discussões
- Interfaces para melhor gerenciamento dos grupos e discussões

A publicação na Apple Store é um cenário possível em trabalhos futuros, devido ao fato de termos criado o aplicativo com um *framework* híbrido, que acaba com a necessidade da criação de uma nova base de código para publicação na plataforma. Seria necessário ter acesso a um computador da Apple, para que os processos de *build*, geração de certificados e disponibilização na loja virtual fossem feitos.

Baseado nas análises acerca dos resultados gerados pelo servidor, uma possível nova funcionalidade, seria a criação de métricas para avaliação individual de cada participante, com foco em saber o desempenho de cada indivíduo da rede de discussão, o que pode trazer a possibilidade de melhor avaliar os participantes e a discussão como um todo. Aliado a isto, outra proposta de trabalhos futuros, seria a criação de novas interfaces para o gerenciamento dos participantes dentro dos grupos, consistindo na edição do participante que está presente dentro de um grupo, remoção de um participante que não está presente naquela discussão e a criação de uma interface para verificação do desempenho daquele participante visando avaliá-lo individualmente.

5 REFERÊNCIAS BIBLIOGRÁFICAS

ALBERT, R; BARABÁSI, A. L. **Statistical Mechanics of Complex Networks.** Rev. Mod. Phys. 74, p. 47-97, 2002.

BARABA´SI AL & ALBERT R. (1999). **Emergence of Scaling in Random Networks**. SCIENCE, 286: 509-512.

Built-in Types — Python 3.8.1rc1 documentation. Disponível em: <https://docs.python.org/3/library/stdtypes.html>. Acesso em: 27 Nov. 2022.

CARVALHO, A. M. **Dinâmica de doenças infecciosas em redes complexas**, 2012. 83f. Tese (Doutorado em Física) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012.

Cloud Run: do contêiner à produção em segundos. Disponível em: <https://cloud.google.com/run?hl=pt-br>. Acesso em: 3 nov. 2022.

COHEN, J. D., & TONG, F. (2001). **The face of controversy.** ***Science***, *293*(5539), 2405-2407.

DANIELSSON, W.; FRÖBERG, A.; BERGLUND, E. **React Native application development -A comparison between native Android and React Native**. [s.l: s.n.]. Disponível em: <https://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02.pdf>. Acesso em: 27 nov. 2022.

Developer Nation Community. Disponível em: <https://www.developernation.net/resources/reports/state-of-the-developer-nation-q3-2022>. Acesso em: 27 nov. 2022.

DIMITROVA, S. *et al*. **Package "KSgeneral" Title Computing P-Values of the K-S Test for (Dis)Continuous Null Distribution.** 2022.

Documentation for rpy2 — rpy2 3.3.1 documentation. Disponível em: <https://rpy2.github.io/doc/latest/html/index.html>. Acesso em: 27 nov. 2022.

Expo. Disponível em: <https://expo.dev/>. Acesso em: 26 set. 2022.

Factory Method. Disponível em: https://refactoring.guru/pt-br/design-patterns/factory-method. Acesso em: 26 set. 2022.

FIGMA. **Figma: the collaborative interface design tool**. Disponível em: <https://www.figma.com/>. Acesso em: 27 nov. 2022.

Firebase Pricing. Disponível em: https://firebase.google.com/pricing?hl=pt-br. Acesso em: 26 de set. 2022.

Google Cloud Pricing Calculator. Disponível em: https://cloud.google.com/products/calculator/. Acesso em: 26 out. 2022.

GOMEZ RODRIGUEZ M, LESKOVEC J, BALDUZZI D, SCHÖLKOPF B. **Uncovering the structure and temporal dynamics of information propagation**. Network Science 2 (1): 26–65, 2014.

GUIMERA, R., & AMARAL, L. A. N. (2005). **Cartography of complex networks: modules and universal roles**. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(02), P02001.

How Google Play Works. Disponível em: <https://play.google.com/about/howplayworks/>. Acesso em: 21 Nov. 2022

HÜLLER, K. S. et al. **Análise de Ponto de Função: estudo de caso para valoração de custos no desenvolvimento de um sistema computacional em NITs**. Navus: Revista de Gestão e Tecnologia, n. 11, p. 1–18, 2021.

igraph – Network analysis software. Disponível em: <https://igraph.org/>. Acesso em: 26 de set. 2022.

KUROSE, J. F.; ROSS, K. W.. **Computer networking: a top-down approach**. 6. ed. AddisonWesley, 2012. 864p.

METZ, J; CALVO, R; SENO, E. R. M; ROMERO, R. A. F; LIANG, Z. **Redes Complexas: conceitos e aplicações. Instituto de Ciências Matemáticas e de Computação. 2007**.

MICROSOFT. TypeScript - JavaScript that scales. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 26 de set. 2022

MYERS S & LESKOVEC J. **On the Convexity of Latent Social Network Inference**. Advances in Neural Information Processing Systems 23 (NIPS 2010).

OHTSUKI H., HAUERT C., LIEBERMAN E. & NOWAK M.A. **A simple rule for the evolution of cooperation on graphs and social networks**. Nature, 441: 502–505, 2006.

OSSADA, R. **Modelagem da dinâmica de doenças infecciosas em redes de movimentação de animais**. 2011, 43f. Dissertação (Mestrado em Ciências) – Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2011.

Pesquisa textual | Tribunal de Contas da União. Disponível em: <https://pesquisa.apps.tcu.gov.br/#/documento/acordao-completo/1346320179.PROC/%2520/DTRELEVANCIA%2520desc%252C%2520NUMACORDAOINT%2520desc/1/sinonimos%253Dfalse>. Acesso em: 20 dez. 2022.

QUEIROZ, A. **PBL, PROBLEMAS QUE TRAZEM SOLUÇÕES**. Revista Psicologia, Diversidade e Saúde, [S. l.], v. 1, n. 1, 2012. DOI: 10.17267/2317-3394rpds.v1i1.36. Disponível em: https://www5.bahiana.edu.br/index.php/psicologia/article/view/36. Acesso em: 27 nov. 2022.

React Native · A framework for building native apps using React. Disponível em: <https://reactnative.dev/>. Acesso em: 26 set. 2022.

Retrospectiva 2021: Brasil tem dois dispositivos digitais por habitante, revela pesquisa da FGV. Disponível em: <https://portal.fgv.br/noticias/retrospectiva-2021-brasil-tem-dois-dispositivos-digitais-habitante-revela-pesquisa-fgv>. Acesso em: 27 Nov. 2022.

ROMERO B, RODRÍGUEZ S, BEZOS J, DÍAZ R, COPANO MF, MEREDIZI, *et al*.. **Humans as Source of Mycobacterium tuberculosis Infection in Cattle**. Spain. Emerging Infectious Diseases, 17(12): 2393-95, 2011

RONACHER, Armin. **Documentação Flask**. Disponível em: <https://Flask-ptbr.readthedocs.io/en/latest/foreword.html>. Acesso em: 26 set. 2022.

SINGH, D.; GARG, R. **R, Language for Data Analytics**. Disponível em: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3355120>. Acesso em: 27 nov. 2022.

SOLÉ RV, VALVERDE V. **Information Theory of Complex Networks: On Evolution and Architectural Constraints**. In: Complex Networks, 2004, Volume 650, ISBN : 978-3-540-22354-2.

Stack Overflow Developer Survey 2022. Disponível em: <https://survey.stackoverflow.co/2022/>. Acesso em: 27 nov. 2022.

Strategy Analytics: Half the World Owns a Smartphone. Disponível em: <https://news.strategyanalytics.com/press-releases/press-release-details/2021/Strategy-Analytics-Half-the-World-Owns-a-Smartphone/default.aspx>. Acesso em: 27 Nov. 2022.

STROGATZ S.H. **Exploring complex networks**. Nature, 410(6825):268-76, 2001.

Taxas de serviço - Ajuda do Play Console. Disponível em: <https://support.google.com/googleplay/android-developer/answer/112622?hl=pt-BR>. Acesso em: 27 nov. 2022.

TREVES A. **Frontal latching networks: a possible neural basis for infinite recursion**. Cogn Neuropsychol, 22(3):276-91, 2005.

WATTS D.J. & STROGATZ S.H. **Collective dynamics of 'small-world' networks**. Nature, 393(6684):440-2, 1998.

What is Cloud Run | Cloud Run Documentation. Disponível em: <https://cloud.google.com/run/docs/overview/what-is-cloud-run>. Acesso em: 21 nov. 2022.

YEE, S. J. W. AND D. **Why You Should Become a UseR: A Brief Introduction to R**. APS Observer, v. 30, n. 3, 28 fev. 2017.