UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Carlos César Morais

Sistema Web para Mapeamento de Perfis Suscetíveis a Serem Ludibriados Por Fake News - Quiz Fato ou Fake?

Uberlândia, Brasil

2022

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Carlos César Morais

Sistema Web para Mapeamento de Perfis Suscetíveis a Serem Ludibriados Por Fake News - Quiz Fato ou Fake?

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof.ª Dra. Maria Adriana Vidigal de Lima

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2022

Resumo

A todo momento, a humanidade é bombardeada com inúmeras informações sobre diversos assuntos através do compartilhamento de informações. Estes compartilhamentos ganharam força com o surgimento das redes sociais, dos *podcasts*, portais de notícias, entre outros, uma vez que, ao receber uma informação, uma pessoa pode encaminhá-la a diversas outras pessoas. Contudo, nem sempre existe veracidade nessas informações, gerando assim um fluxo de *fake news*. Destarte, este trabalho propõe um sistema capaz de integrar-se ao site do G1 (*g1.globo.com*) e coletar notícias que foram verificadas pela plataforma e já definidas como fato ou *fake*, disponibilizando-as em formato de *quiz*, para que as pessoas possam responder se consideram aquela notícia verdadeira ou falsa, ou seja, se é fato ou *fake*. Com o objetivo de testar o sistema desenvolvido, aplicou-se o *quiz* a um grupo de pessoas, e em seguida, uma análise quantitativa foi elaborada a partir dos resultados obtidos e de informações fornecidas pelos participantes como idade, escolaridade e fontes frequentes de busca de informação. Por fim, a temática do *quiz* centralizou-se na área da saúde, por estar em alta devido à pandemia do Covid-19, sendo certo que, diariamente diversas informações foram compartilhadas sobre o assunto. O sistema proposto compõe-se de uma aplicação *web* fundamentada em API REST, banco de dados relacional e computação em nuvem.

Palavras-chave: Sistema Web, *Fake News*, Covid-19, *Cloud Computing*

Aos meus pais Cleomar e Eunice, meu irmão Júlio e minha esposa Maria.

Sumário

1 Introdução

A evolução da tecnologia proporciona diversos benefícios ao estilo de vida atual da humanidade. Um dos exemplos é na comunicação, o meio pelo qual o ser humano transmite uma informação a outro ser humano. Hoje é possível transmitir uma informação de forma rápida em razão dos diversos meios de comunicação existentes, como redes sociais, uma página Web, um software aplicativo, uma TV ou rádio, de modo que há um elevado consumo de informação a todo momento.

Nem sempre essas informações que são consumidas possuem alguma veracidade, algumas delas são verdadeiras, outras são parcialmente verdadeiras e outras ainda são totalmente falsas. Com o intuito de tornar cada vez mais pública tal informação as pessoas sequer procuram saber se tal fato é verdadeiro ou não, ou seja, não questionam sua autenticidade, apenas lêem e repassam, seja pelo desejo de compartilhar rapidamente aquela informação ou de imediato já considerá-la verdadeira pelo simples fato de coincidir com sua a opinião sobre o assunto em questão.

O compartilhamento de uma notícia em determinado meio de comunicação pode conter inúmeros objetivos, como ganhos financeiros, consolidação da credibilidade em fornecer informações, popularidade entre grupos sociais, prejudicar a imagem de uma marca ou governo, comprovar uma opinião pessoal, etc. Para alcançar alguns dos objetivos, pode-se compartilhar uma notícia não verídica, seja por ignorância ou até mesmo para provocar instabilidade ou discórdia.

De acordo com TJPR (2020), o conceito de *fake news* ganhou forma devido à enorme quantidade de notícias fraudulentas que circulam nos meios de comunicação. A publicação destaca que, em 2018, o IPSOS (2018) divulgou um estudo intitulado "*Fake news, filter bubbles, post-truth and trust* (Notícias falsas, filtro de bolhas, pós-verdade e verdade)", que revela dados importantes. De acordo com o levantamento, 62% dos entrevistados do Brasil admitiram ter acreditado em notícias falsas, valor acima da média mundial que é de 48%.

Desde o dia 30 de janeiro de 2020, de acordo com Pan-Americana (2020), data em que a Organização Mundial de Saúde (OMS) declarou que o surto do novo coronavírus (COVID-19), constitui uma Emergência de Saúde Pública de Importância Internacional (ESPII), tem surgido a todo instante notícias sobre o assunto. Como consequência, diversas *fake news* são espalhadas a todo momento, relacionadas à forma de contágio, tratamentos, vacinas, eficácia de tratamentos/vacinas, declarações de governos e opiniões pessoais sem embasamento científico.

A análise realizada em Barcelos (2021) mostra que até a data de 30 de junho de

2020, *fake news* veiculadas e relacionadas com a pandemia da COVID-19 no Brasil somam 329, relatadas nos seguintes sites: 253 no G1[1] e 76 no Ministério da Saúde[2]. O site G1 disponibiliza em sua plataforma, um serviço de monitoramento e checagem de conteúdos duvidosos que esclarece o que é falso ou verdadeiro em mensagens disseminadas pelo celular e pela internet.

Com o propósito de automatizar a coleta de informações checadas do site G1 e realizar um levantamento quantitativo de dados na tentativa de mapear perfis suscetíveis a serem ludibriados por *fake news*, propõe-se neste trabalho, a criação de um sistema capaz de capturar notícias de forma automatizada do site em tela, passar por um processo de filtragem de texto, realizar o armazenamento dessas notícias e disponibilizá-las através de uma página Web em forma de um *quiz* para coleta de informações de cada participante, armazenando assim tais informações para gerar os relatórios finais.

Apresenta-se então, o Quiz Fato ou Fake?[3], um sistema Web que recebe como entrada informações pessoais de cada participante, como idade, nível de escolaridade e fontes de pesquisa. O sistema capta respostas de um *quiz* que contém dez notícias disseminadas e checadas pelo site G1, e somente pelo título e primeiro parágrafo o participante deve determinar se a notícia é um fato ou *fake*. Ao final do processo, é disponibilizado, além da pontuação obtida, o endereço eletrônico de cada notícia respondida a fim de conscientizar o participante sobre a veracidade ou não de tal informação.

[1] *g1.globo.com*
[2] *www.gov.br/saude/pt-br*
[3] *www.quizcovid19.com.br*

1.1 Objetivos

1.1.1 Objetivo Geral

Este projeto de graduação tem como objetivo geral o desenvolvimento de um sistema Web, que seja capaz de se integrar de forma automatizada ao site G1 para a busca de notícias checadas que viralizaram nos meios de comunicação e disponibilizar aos participantes uma sequência destas notícias em formato de *quiz*. O participante responde ao *quiz* fornecendo sua percepção em relação à cada uma das notícias e o sistema coleta as respostas para em seguida calcular o resultado do acerto da percepção do participante. Os resultados poderão ser utilizados para mapear perfis suscetíveis a serem ludibriados por uma notícia falsa. A implementação do sistema concentrou-se, no primeiro momento, no tema da saúde, devido a alta taxa de informações referentes à pandemia do coronavírus. Porém, o sistema pode facilmente ser adaptado para outros temas específicos, como política, meio ambiente e economia.

1.1.2 Objetivos Específicos

Como objetivos específicos deste projeto de graduação apresentam-se:

- Utilizar tecnologias e *frameworks* atuais que atendam ao objetivo geral proposto;
- Pormenorizar e desenvolver aplicações *back-end*, *front-end* e banco de dados para armazenamento das informações de integração;
- Implantar o modelo de arquitetura hexagonal na aplicação *back-end*;
- Empregar toda aplicação a um provedor de *cloud computing* para integração dos componentes;
- Testar o sistema a partir da disponibilização do quiz a um conjunto de participantes em suas redes sociais como Facebook e Whatsapp; e
- Validar o sistema implementado, analisando quantitativamente os resultados obtidos, para mapear perfis suscetíveis a serem ludibriados por uma notícia falsa.

1.2 Metodologia

Com o propósito de contemplar a organização e planejamento mais efetivo de trabalho, foram adotadas técnicas ágeis de gestão de projeto, especificamente o SCRUM, que segundo o guia SCRUM (2020) desenvolvido por seus cocriadores, Ken Schwaber e Jeff Sutherland, é um *framework* leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos. Para aplicação do *framework*, foi utilizado o aplicativo de gerenciamento de projeto Trello[4]

Para melhor entendimento do contexto, foram utilizadas *sprints* com duração de três semanas. Antes de iniciar a primeira *sprint*, foi necessária a criação do *Product Backlog* pelo Dono do Produto (neste caso o papel foi dividido entre o aluno e a orientadora) levantando todas as necessidades para a entrega total do projeto.

Com o *backlog* estabelecido, iniciou-se o planejamento das *sprints* para identificar o desenvolvimento a ser realizado a cada período de três semanas. A cada *sprint* três breves encontros semanais de quinze minutos foram realizados entre o aluno e a orientadora para alinhar os pontos que estavam sendo desenvolvidos. Estes encontros são conhecidos como *Daily Scrum* e foram realizados via *Whatsapp* e reuniões virtuais. Ao final de cada *sprint*, apresentou-se o desenvolvimento realizado e as lições aprendidas no período dessas três semanas. Estes processos que acontecem na finalização da etapa são denominados, respectivamente Revisão e Retrospectiva de *sprint*. O ciclo dos *sprints* se repetiu até o fim do projeto.

Para a criação do *backlog* foram realizados estudos, levantamento de tecnologias e configurações necessárias a serem utilizadas para alcançar o objetivo do projeto. Para a aplicação *back-end* foi escolhida a linguagem de programação Java, versão 11, em conjunto com o *framework Spring Boot*. Para aplicação *front-end* a linguagem de programação utilizada foi a JavaScript junto com a biblioteca *React* (SILVA, 2021). A arquitetura de software definida para o projeto foi a Arquitetura Hexagonal proposta por Cockburn (2005) e o modelo REST (*Representational State Transfer*). Como plataforma de *cloud computing* utilizou-se a Amazon Web Services[5], que disponibiliza diversos recursos em nuvem, a partir dos quais é possível hospedar toda aplicação em um só lugar.

[4] *www.trello.com*
[5] *aws.amazon.com*

1.3 Contribuições

As contribuições deste projeto dividem-se em: (i) construção de um Sistema Web capaz de integrar-se ao site do G1 para coleta de notícias, (ii) apresentação de notícias ao cidadão em formato de quiz para registro de respostas e, (iii) possibilidade de análise dos dados coletados para o mapeamento de perfis suscetíveis a serem ludibriados por uma notícia falsa.

1.4 Organização do Trabalho

Para uma melhor separação e compreensão do conteúdo, os próximos capítulos deste trabalho estão organizados da seguinte maneira:

- O Capítulo 2 apresenta os conceitos básicos necessários para compreensão do trabalho, como, Engenharia de Software, Método Ágil, Banco de Dados Relacional, *Design Pattern*s, Arquitetura Hexagonal, API REST, Aplicação Web e *Cloud Computing*, e também são apresentados trabalhos relacionados;
- O Capítulo 3 mostra como o trabalho foi desenvolvido desde a etapa de descrição e pré-processamento dos dados até a parte de implementação e implantação propriamente do projeto.
- O Capítulo 4 expõe as conclusões e considerações finais do trabalho, além de propostas para continuação do mesmo.

2 Fundamentação teórica

Neste capítulo são apresentados os conceitos básicos necessários para compreensão do trabalho, bem como os trabalhos relacionados.

2.1 Conceitos básicos

2.1.1 Engenharia de Software

A Engenharia de Software é definida pelo IEEE (*Institute of Electrical and Electronics Engineers*) (1990) como "aplicação de uma abordagem sistemática, disciplinada e quantificável, para desenvolvimento, operação e manutenção do software". Segundo Pressman e Maxim (2016), a Engenharia de Software abrange um processo, composto por um conjunto de métodos (práticas) e ferramentas que possibilitam aos profissionais desenvolverem software de altíssima qualidade, o processo não é rígido nem deve ser seguido à risca. Deve ser ágil e adaptável (ao problema, ao projeto, à equipe e à cultura organizacional).

No decorrer deste projeto foram aplicados os princípios estabelecidos pela Engenharia de Software para uma melhor organização e planejamento do projeto. Em específico, foi utilizado o modelo de processo, Métodos Ágeis, para auxiliar na aplicação desses princípios.

2.1.2 Método Ágil

Com o objetivo de estabelecer princípios mais eficientes e flexíveis, segundo Highsmith (2001), de 11 a 13 de fevereiro de 2001, dezessete pensadores independentes autointitulados como "*The Agile Alliance*" se reuniram para desenvolver o Manifesto para Desenvolvimento Ágil de Software. Algumas de suas principais diretrizes para direcionar ao caminho ágil, são: entrega contínua e adiantada de software com valor agregado, mudanças nos requisitos são bem-vindas, pessoas de negócios e desenvolvedores devem trabalhar diariamente juntas.

O livro de Hazzan e Dubinsky (2009) mostra que existem várias metodologias ágeis de desenvolvimento de software, como SCRUM[1], *Extreme Programming*[2], DSDM, Crystal (COCKBURN, 2004), FDD, ASD, entre outros. Neste projeto foi empregada a metodologia ágil SCRUM, que segundo o guia SCRUM (2020), é um framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.

[1] https://www.scrum.org/
[2] http://www.extremeprogramming.org

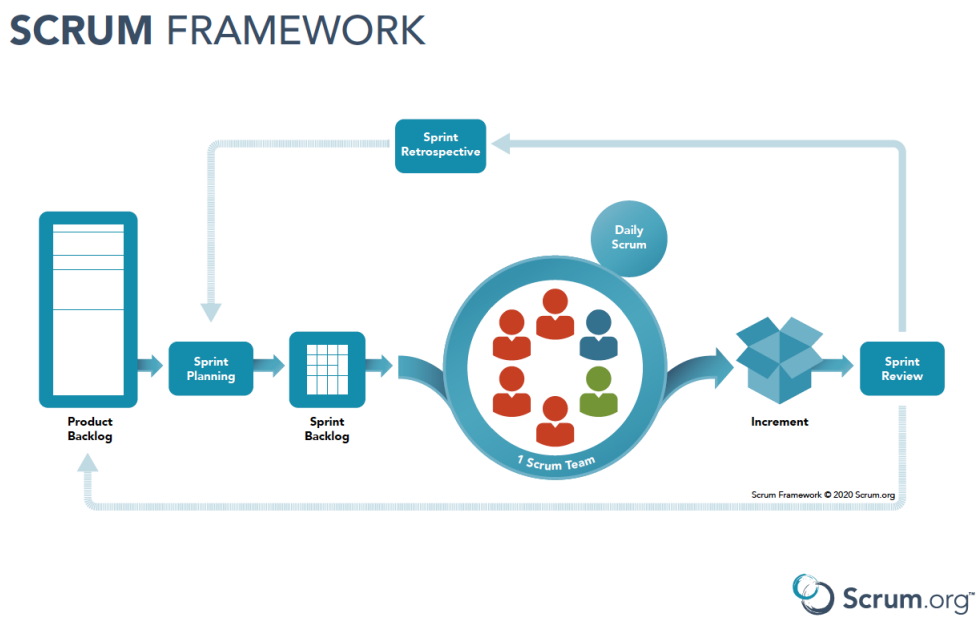

Figura 1 – Ciclo de vida do framework Scrum

Fonte: (SCRUM.ORG, 2020)

2.1.3 Banco de Dados Relacional

Segundo Ramakrishnan e Gehrke (2008), banco de dados é uma coleção de dados que, tipicamente, descreve as atividades de uma ou mais organizações relacionadas. Um sistema de gerenciamento de bancos de dados ou SGBD, é um software projetado para auxiliar a manutenção e utilização de vastos conjuntos de dados. O SGBD serve como um intermediário entre o usuário e o banco de dados.

Um sistema de gerenciamento de bancos de dados relacionais ou SGBDRs segundo Garcia-Molina, Ullman e Widom (2011), usa um modelo simples para armazenar e manipular dados, que consiste em tabelas de duas dimensões, denominadas de "relação". Tais relações representam os dados usando três princípios: estrutura do dado, suas operações e suas restrições.

Neste projeto foi aplicado o MySQL[3] como o sistema de gerenciamento de banco de dados relacional. Para sua funcionalidade é utilizado um conjunto de instruções SQL (*Structured Query Language*), que se trata de uma linguagem de alto nível dedicada a representar os dados e manipulá-los por meio de querys, com ações como ler, criar, atualizar e deletar dados. Também existem outros SGBDRs como Oracle[4], Amazon Aurora, PostgreSQL[5], MariaDB [6], entre outros.

[3] *www.mysql.com*
[4] *www.oracle.com/br/database*
[5] *www.postgresql.org*
[6] *mariadb.org*

2.1.4 Design Patterns

O conceito de padrão de projeto foi criado na década de 70 pelo arquiteto Christopher Alexander e apresentado em seus livros Alexander et al. (1977) e Alexander e Alexander (1979). Em um certo momento Christopher Alexander afirma que: "*cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira*".

Em 1994, um grupo de quatro pessoas intitulado como *Gang of Four* (GOF) escreveu o livro Gamma et al. (1994), o qual teve sua origem quando observaram que os mesmos problemas apareciam diversas vezes em diferentes situações. Começaram então a catalogar a solução para quando ocorresse determinado problema. Em contexto específico, padrões de projeto são descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular. Em geral, um padrão tem quatro elementos essenciais: nome do padrão, problema, solução e consequências.

Os padrões de projeto GOF são classificados em três categorias, a saber: Criação, Estrutural e Comportamental. Existem soluções propostas para cada uma:

- **Criacão**: Singleton, Abstract Factory, Builder e Prototype
- **Estrutural**: Decorator, Facade, Composite e Adapter
- **Comportamental**: Strategy, Chair of Responsability, State e Observer

Como apresentado na Figura 2, é possível visualizar o exemplo de um padrão comportamental, o Padrão Strategy.

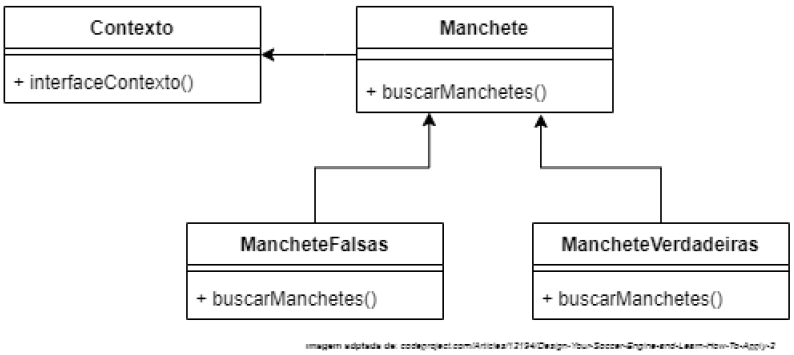

Figura 2 – - Exemplo de diagrama para o Padrão Strategy

Fonte: Próprio autor

2.1.5 Arquitetura Hexagonal

Segundo Sommerville (2011), o projeto de arquitetura está preocupado com a compreensão de como um sistema deve ser organizado e com a estrutura geral desse sistema. Como em padrões de projeto, existem os padrões arquiteturais e individualmente cada um com sua peculiaridade. É possível compreender alguns modelos citados no livro, como, Arquitetura em camadas, Arquitetura de repositório, Arquitetura cliente-servidor e MVC (Modelo-Visão-Controlador).

Em meados dos anos 90, o conceito de Arquitetura Hexagonal foi proposto por Alistair Cockburn, em um artigo publicado no primeiro site editável, WikiWikiWeb[7], onde o conteúdo da wiki é escrito pelos próprios usuários. A ideia para essa arquitetura é de construir sistemas que favorecem reusabilidade de código, alta coesão, baixo acoplamento, independência de tecnologia e que são mais fáceis de serem testados.

Para garantir esse desacoplamento, as classes de domínio não devem depender de classes relacionadas com infraestrutura, tecnologias ou sistemas externos. A comunicação entre esses dois grupos é realizada através de adaptadores. Visualmente, a arquitetura é representada por dois hexágonos concêntricos. As classes de domínio ficam no hexágono interno, enquanto os adaptadores ficam no hexágono externo. A comunicação entre os adaptadores e as classes de domínio são realizadas pelas interfaces, denominadas, portas, conforme apresentado na Figura 3. Segundo VALENTE (2020), a justificativa de Alistair Cockburn para o modelo de hexágonos é que "*cada face do hexágono representa um motivo pelo qual o sistema deve se comunicar com o mundo exterior. É por isso que são hexágonos concêntricos e não círculos concêntricos*".

Mais tarde, em 2012, Robert C. Martin (Uncle Bob) propôs a Arquitetura Limpa, conforme apresentado na Figura 4. Segundo MARTIN (2012), é uma tentativa de integrar arquiteturas como a Arquitetura Hexagonal[8], proposta por Alistair Cockburn, adotada por Steve Freeman e Nat Pryce em seu livro *Growing Object Oriented Software*, com a junções de outras arquiteturas como a Onion Architecture[9] de Jeffrey Palermo, cujo todas elas têm o mesmo objetivo, a separação de interesses.

No presente trabalho foi aplicada a Arquitetura Hexagonal, na justificativa de aprimorar a organização do código, facilitar a manutenção e garantir o desacoplamento. Por ser uma arquitetura orientada a camadas, cada camada possui sua responsabilidade, seguindo as recomendações de Evans, Fowler e Evans (2004) sobre modelagem de software, Domain Driven Design(DDD).

Para melhor entendimento do esquema apresentado na Figura 5, do lado esquerdo tem-se o que podemos chamar de *inbounds*. Eles são compostos por portas e adaptadores.

[7] http://wiki.c2.com/?WikiWikiWeb
[8] *alistair.cockburn.us/hexagonal-architecture*
[9] *jeffreypalermo.com/2008/07/the-onion-architecture-part-1*

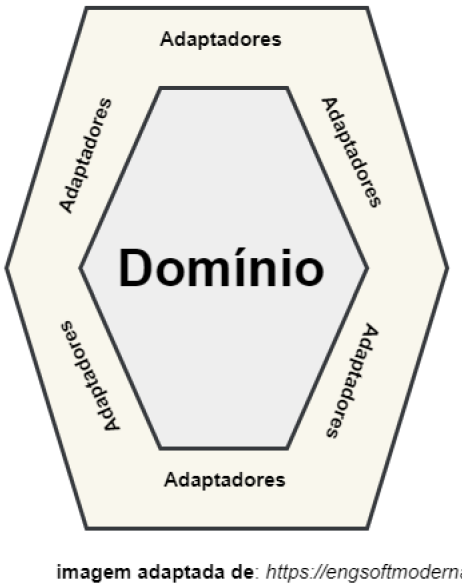

Figura 3 – Arquitetura Hexagonal proposta por Alistair Cockburn

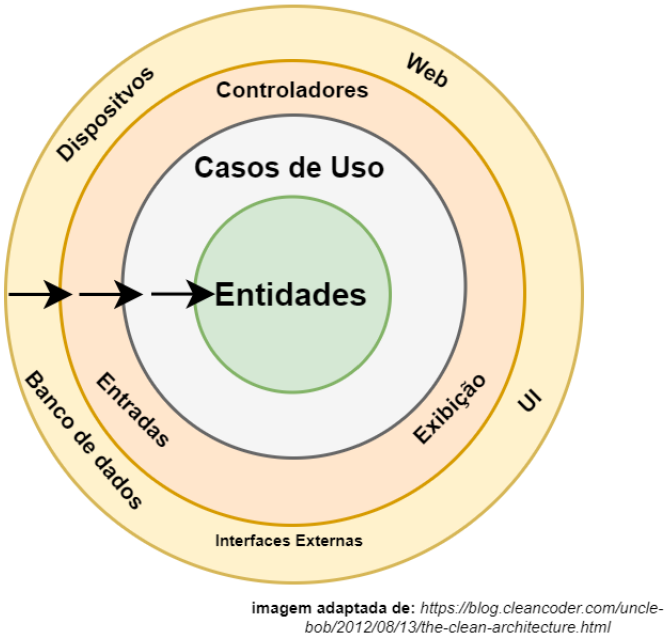

Figura 4 – Arquitetura Limpa proposta por Robert C. Martin

Por exemplo, a porta *inbound* pode ser entrada USB de um computador, o qual possui um formato específico (no caso este seria o contrato da porta) e os adaptadores são hardwares que se conectam a essa mesma porta, como mouse, teclado e pendrive.

A camada do centro é quem recebe estes dados da porta *inbound* e realiza as implementações de negócio. Continuando este exemplo, supõe-se que exista como entrada um pendrive com um arquivo .pdf. A camada centro receberia este arquivo e o trataria de acordo com as necessidades específicas.

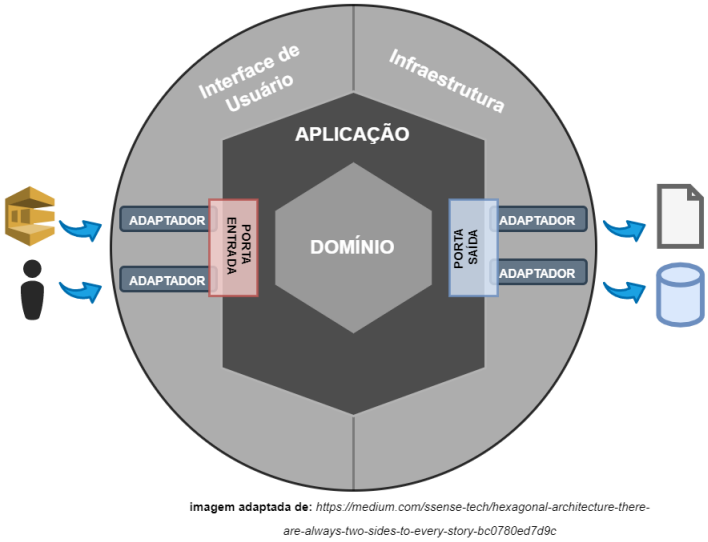

Figura 5 – Arquitetura hexagonal - Ports e Adapters

Fonte:(MARTINEZ, 2021)

Por fim, do lado direito tem-se o que podemos chamar de *outbound*, que também é composto por portas e adaptadores. Ainda no mesmo exemplo, a porta *outbound* receberia os dados do centro, no caso, o arquivo .pdf modificado. Então, pode-se imaginar que a porta *outbound* seria uma saída LTP (no caso este seria o contrato da porta) e o adaptador seria uma impressora por exemplo.

2.1.6 API REST

Application Programming Interface (API) é uma aplicação que provê serviços para que outras aplicações possam usá-los. Segundo Masse (2011), de um modo geral, uma API expõe um conjunto de dados e funções para facilitar as interações entre programas de computador e permitir que eles troquem informações. Fielding (2000), em sua dissertação de doutorado "*Architectural Styles and the Design of Network-based Software Architectures*", descreveu um estilo arquitetural para a Web, composto de algumas restrições e o nomeou de Representational State Transfer (REST).

O modelo REST explora o protocolo HTTP (*Hypertext Transfer Protocol*) para prover um modelo de comunicação seguro e padronizado. Além do HTTP, podem ser utilizadas tecnologias como XML (*eXtensible Markup Language*), JSON (*Javascript Object Notation*) e URI (*Uniform Resource Identifier*) que permitem desenvolver aplicações Web distribuídas de forma rápida e robusta.

Neste contexto, pode-se desenvolver uma API para um servidor Web utilizando a arquitetura REST. Essa arquitetura é formada por regras, princípios e limitações para aplicações Web. Ainda segundo Masse (2011), uma Web API em conformidade com o estilo arquitetural REST, é uma REST API.

Como dito anteriormente, algumas regras são estabelecidas para a arquitetura REST, uma delas é o uso do *Uniform Resource Identifiers* (URIs), é uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet. Ele permite a identificação uniforme de recursos por meio de um conjunto extensível de esquemas de nomenclatura seguindo o formato de documento RFC 3986, segundo Berners-Lee (2005), o qual possui a seguinte estrutura:

URI = scheme “://” authority “/” path [“?” query] [“#” fragment]

Seguindo a ideia de como podem ser mapeadas as pastas em nosso sistema operacional, e que deseja-se recuperar informações de um determinado aluno da Faculdade de Computação da UFU (FACOM), teríamos o seguinte exemplo: cidade/universidade/-faculdade/facom.

http://www.portal.facom.ufu.br/graduacao/bsi/aluno?matricula=11611BCC037

2.1.7 Aplicação Web

De acordo com Caetano (2019), uma aplicação Web é um software executado em um servidor Web por meio de requisições advindas de um browser, que utiliza de tecnologias Web para atuar no papel de cliente de um sistema cliente-servidor. É encarregado da parte visual e lógica do cliente, sendo também quem conduz a comunicação entre o cliente e o servidor. Tais aplicações são desenvolvidas através do uso de algumas tecnologias comuns, como o protocolo de comunicação *Hypertext Transfer Protocol* (HTTP), a *Hypertext Markup Language* (HTML), a *Cascading Style Sheets* (CSS) e script languages.

O Hypertext Transfer Protocol (HTTP) é o protocolo que os programas usam para se comunicar pela World Wide Web. Existem muitas aplicações de HTTP, mas o HTTP é mais famoso por conversas bidirecionais cliente-servidor. (HTTP). A utilização deste padrão permite que o cliente efetue requisições de recursos ao servidor.

O *HyperText Markup Language*(HTML) é uma linguagem de marcação, usada para definir a estrutura dos elementos de uma página, como parágrafos, links, títulos, tabelas, imagens e até vídeos. Segundo (CAETANO, 2019), a anatomia da estrutura HTML, consiste em tag de abertura, tag de fechamento, conteúdo e elemento (MDN, 2022a).

O *Cascading Style Sheet* (CSS) é uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML ou em XML (incluindo várias linguagens em XML como SVG, MathML ou XHTML). É padronizada em navegadores Web de acordo com as especificações da W3C (BOS, 1994) e pode ser adicionado a documentos HTML de 3 maneiras: inline, externo, interno.

```
1 <!DOCTYPE html>
2 <html>
```

```
3     <head>
4        <style>
5           body {
6             background-color: lightgrey;
7             color: black;
8           }
9             div {
10            background-color: white;
11            color: black;
12            height: 100px;
13            width: 100%;
14            text-align: center;
15            padding: 10px;
16            border-radius: 2px;
17          }
18       </style>
19    </head>
20    <body>
21       <h1 style="color:green;text-align:center;"> Projeto de Graduacao </h1>
22       <div>
23          <h1 style="color:black;"> Toda esta pagina foi criada utilizando HTML & CSS </h1
                >
24       </div>
25    </body>
26 </html>
```

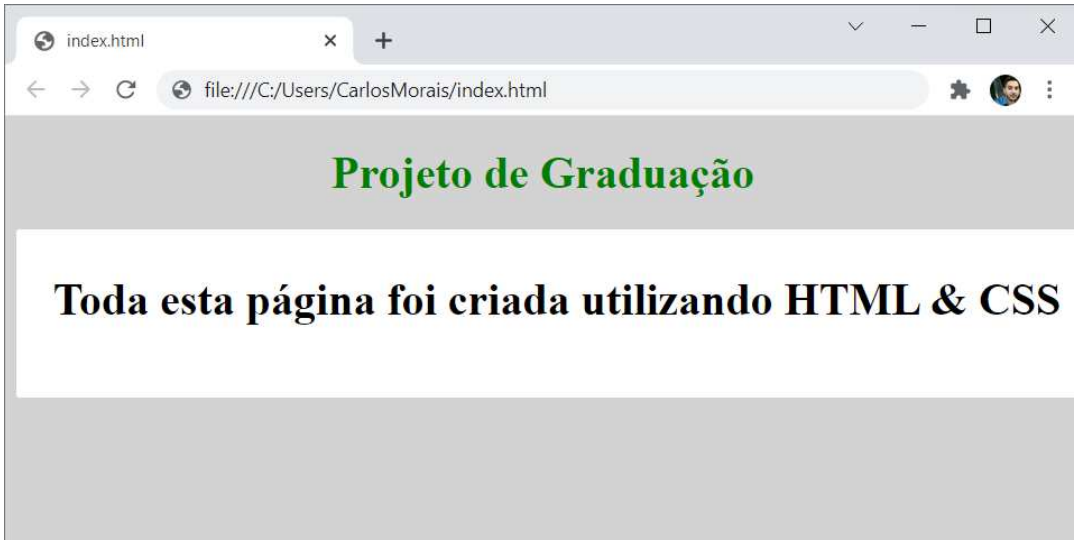

Figura 6 – Exemplo de página criada com HTML e CSS

Fonte: Próprio autor

Segundo Flanagan (2011), a linguagem HTML é utilizada para especificar o conteúdo de páginas Web, o CSS é utilizado para especificar a apresentação da página e o JavaScript é utilizada para especificar o comportamento delas.

Anualmente são criadas bibliotecas que ajudam no desenvolvimento de scripts como, fornecimento de funções prontas que por trás são funções gigantescas. A biblioteca React é um exemplo, criada em 2011 pelo engenheiro do Facebook, Jordan Walke, e usada pela primeira vez no feed de notícias da empresa (SILVA, 2021).

2.1.8 *Cloud Computing*

Computação em nuvem, ou do inglês, Cloud Computing, é um termo para descrever um ambiente de computação baseado em uma imensa rede de servidores, sejam virtuais ou físicos. Segundo Taurion (2009), uma definição simples seria "um conjunto de recursos com capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na internet".

Com a necessidade de manter um ambiente cliente-servidor, foi utilizada uma plataforma de computação em nuvem, com os benefícios de manter a aplicação sempre online realizando assim a integração entre cliente-servidor e disponibilizando um banco de dados para armazenamento com integração ao servidor. Nos dias atuais existem diversas plataformas que disponibilizam este serviço, sendo que as mais conhecidas são, Amazon Web Services (AWS), Google Cloud Platform[10], Microsoft Azure[11], Oracle Cloud[12] e IBM Cloud[13].

Há três tipos principais de computação em nuvem: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS). Na Infrastructure as a Service (IaaS), fica liberado o acesso a toda infraestrutura de nuvem, do sistema operacional aos recursos de segurança, com maior automonia para manter a solução em nuvem. Por sua vez, a *Platform as a Service* (PaaS), diferentemente do IaaS, tem seu foco na aplicação, não necessitando de preocupação com questões relacionadas a infraestrutura de nuvem, mas sim com a plataforma que irá utilizar. Por fim, a *Software as a Service* (SaaS) está relacionada mais a processos de negócios como fornecimento de produtos, um exemplo é a utilização de um *CoreBank* para sua aplicação.

Para a Amazon Web Services (AWS), computação em nuvem é a entrega de recursos de TI sob demanda por meio da Internet com definição de preço de pagamento conforme o uso. É possível acessar serviços de tecnologia, como capacidade computacional, armazenamento e bancos de dados conforme a necessidade, usando um provedor de

[10] *cloud.google.com*
[11] *azure.microsoft.com*
[12] *oracle.com/br/cloud*
[13] *cloud.ibm.com*

nuvem como a AWS.

Neste projeto, foi utilizada a AWS, pois, além de oferecer ferramentas e serviços extremamente poderosos para desenvolvimento, rede, processamento e gerenciamento de dados, ela disponibiliza uma conta gratuita por 12 meses, contemplando diversos recursos essenciais que foram utilizados neste projeto:

1. Amazon API Gateway: segundo Amazon (2022), é um serviço que permite que desenvolvedores criem, publiquem, mantenham, monitorem e protejam APIs em qualquer escala com facilidade.

2. Amazon Elastic Compute Cloud (Amazon EC2): segundo Amazon (2022), é um serviço Web que disponibiliza capacidade computacional segura e redimensionável na nuvem. O Amazon RDS for MySQL, gerencia as tarefas de administração de banco de dados, incluindo backups, aplicação de patches de software, monitoramento, dimensionamento e replicação.

3. Amazon Route 53: segundo Amazon (2022), é um serviço *Web Domain Name System* (DNS) projetado para oferecer direcionamento de usuários finais a recursos da AWS ou aplicações da internet.

4. AWS Amplify: segundo Araujo (2021), oferece um fluxo CI/CD baseado em Git para criar, implantar e hospedar aplicações SPA Web ou sites estáticos com *serveless backends*.

2.2 Trabalhos Relacionados

O termo *fake news* se iniciou na década de 1890. Como retratado em Kalsnes (2018) por mais de um século, tem sido usado para indicar falsidade impressa como notícia. Em 1890 e 1891, o dicionário Merriam-Webster [14] usaram o termo fake news em artigos em conexão com informações falsas.

Atualmente, é possível encontrar diversos trabalhos no tema das *fake news* relacionadas à Covid-19. O motivo é que a pandemia trouxe uma ampla quantidade de informações de fatos e suposições, gerando assim diversas opiniões entre os indivíduos. Por exemplo, o trabalho de Galhardi (2021), intitulado *Fato ou Fake? Uma análise da desinformação frente à pandemia da Covid-19 no Brasil*, apresenta uma reflexão sobre as notícias falsas a respeito do novo coronavírus (Sars-CoV-2) mais disseminadas nas redes sociais e mostra como podem causar prejuízos à saúde pública.

As análises de Galhardi (2021) sobre as notícias falsas recebidas entre 17 de março e 10 de abril de 2020, revelam que 65% delas ensinavam métodos caseiros para prevenir

[14] *merriam-webster.com*

o contágio da Covid-19; 20% mostravam métodos caseiros para curar a doença; 5,7% se referiam a golpes bancários; 5% faziam menção a golpes sobre arrecadações para instituição de pesquisa; e 4,3% diziam respeito ao uso do novo coronavírus como estratégia política. A pesquisa apontou que 10,5% das notícias falsas foram publicadas no Instagram, 15,8% no Facebook e 73,7% circularam via WhatsApp. Como consequência, as notícias falsas disseminadas pelas plataformas digitais influenciam o comportamento da população e colocam em risco a adesão do cidadão aos cuidados cientificamente comprovados.

Outro artigo, de Barcelos (2021), publicado na Revista Panamericana de Salud Pública em 9 de junho de 2021 com o título *Análise de fake news veiculadas durante a pandemia de COVID-19 no Brasil* realiza uma busca na base de dados do site do G1 e no Ministério da Saúde para encontrar as *fakes news* coletadas até 30 de Junho de 2020 que circularam na mídia e categorizadas de acordo com o seu conteúdo. Foram identificadas 329 fake news (253 no G1 e 76 no Ministério da Saúde) relacionadas à pandemia da COVID-19. As categorias temáticas mais frequentes foram: política (por exemplo, governantes falsificando a vacinação contra a COVID-19, com 20,1%), epidemiologia e estatística (proporção dos casos e óbitos, 19,5%) e prevenção (16,1%). Conforme o Google Trends, houve um aumento de 34,3% nas buscas que utilizavam termos presentes nas fake news. O maior aumento nas buscas ocorreu no Sudeste (45,1%) e Nordeste (27,8%).

A questão *Quais são as ferramentas utilizadas para detecção precoce das fake news?* foi estudada no trabalho de Beppler et al. (2021). A revisão da literatura identificou 13 propostas de ferramentas de identificação de fake news, das quais 84,61% usavam detecção de texto da língua inglesa e 69,23% baseavam-se em aprendizado de máquina sobre diversas bases de dados, sendo o Twitter a rede social mais utilizada como ambiente para a extração de notícias falsas. Dentre as ferramentas levantadas, destacou-se a BRENDA, desenvolvida por Botnevik, Sakariassen e Setty (2020), que consiste em uma extensão de navegador para detecção de fake news. A ferramenta utiliza uma arquitetura de rede neural para identificar a necessidade de checagem automática de uma informação e apresenta ao usuário a classificação e o resultado da checagem, sem que o usuário precise sair da página Web em que está lendo a notícia.

3 Desenvolvimento

Neste capítulo serão apresentadas todas as etapas necessárias para alcançar o objetivo deste projeto, explicando o processo de desenvolvimento, modelagem dos dados, implementação do sistema, implantação do projeto e apresentando a aplicação final.

Para este trabalho, foi utilizado no *backend* a linguagem Java 11 em conjunto com o framework Spring Boot, que nos fornece a criação de *Controllers*. O backend foi hospedado na *cloud Amazon Web Services* (AWS) e para integração entre a aplicação Web e o backend, é necessário passar pelo API Gateway, recurso que a AWS fornece para integração externa.

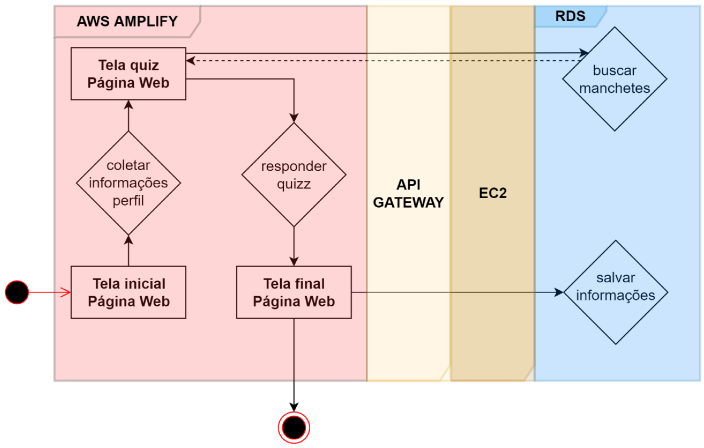

Figura 7 – Fluxograma do projeto

Fonte: Próprio autor

Como já mencionado na seção 1.2, todo desenvolvimento foi realizado utilizando o framework ágil Scrum, percorrendo toda esteira necessária proposta pelo framework, conforme apresentado na Figura 1, as dividindo em sprints.

Como ferramenta para aplicar o framework, utilizamos o aplicativo de gerenciamento de projeto Trello, conforme apresentado na Figura 8, sendo, que existem outras ferramentas como Jira Software[1] e Azure Devops[2]

[1] https://www.atlassian.com/br/software/jira
[2] https://azure.microsoft.com/pt-br/services/devops/

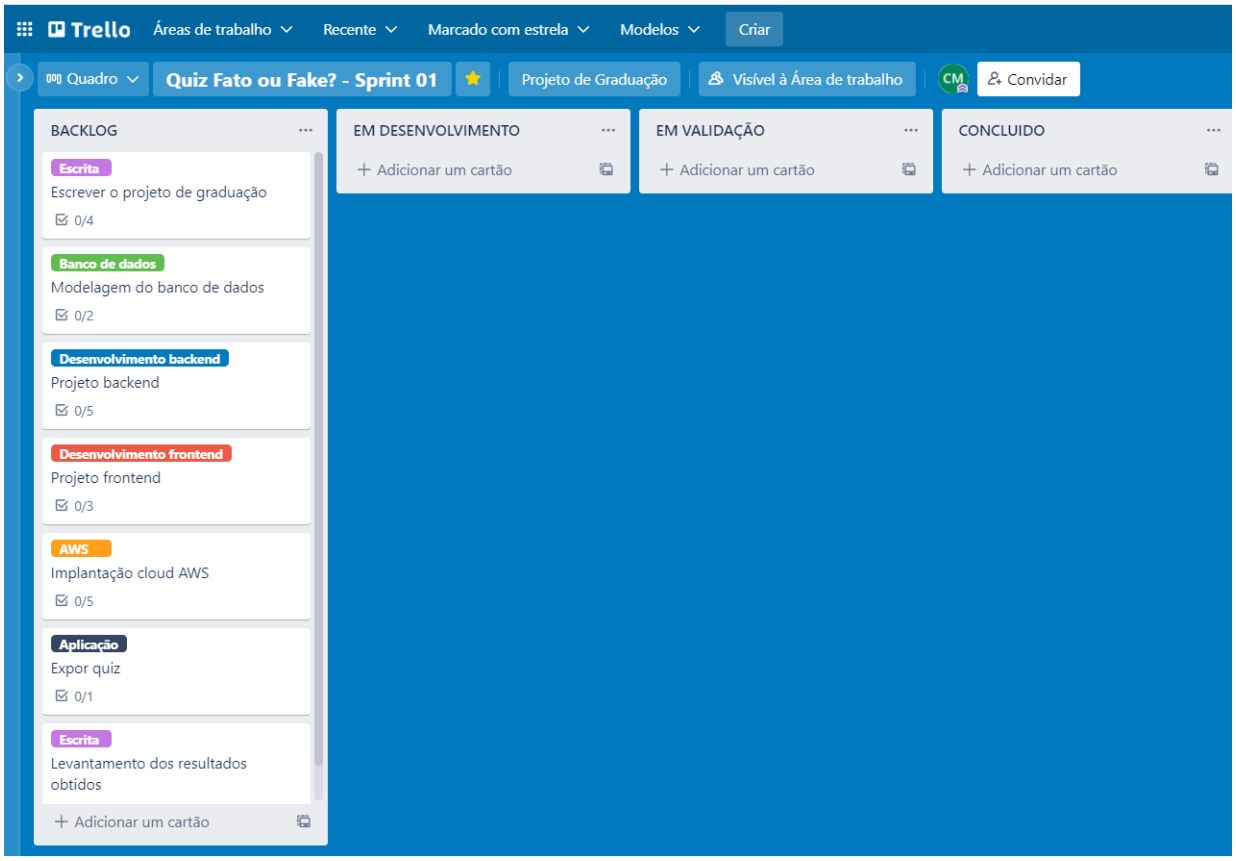

Figura 8 – Modelo final do Product Backlog no Trello

Fonte: Próprio autor

O processo de desenvolvimento iniciou-se com a criação do Product Backlog, sendo estudado e definido qual seria o tema deste projeto. Foram mapeados todos os objetivos necessários para alcançar a finalização deste projeto, podendo, então, iniciar a primeira sprint do projeto.

Desta feita, no presente trabalho foi implantado o framework Scrum utilizando o aplicativo de gerenciamento Trello. A estrutura do framework neste projeto ficou estabelecido da seguinte forma:

Estrutura Scrum

- **Daily**: três vezes semanais com duração de 15 minutos para acompanhamento das demandas;
- **Scrum Master**: Orientadora;
- **Desenvolvedor**: Aluno;
- ***Product Owner***: Aluno/Orientadora;
- ***Sprint***: duração de três semanas por ser o tempo estimado de entrega de uma demanda;

- ***Review***: Apresentação ao Product Owner do desenvolvimento realizado durante a sprint;
- ***Retrospective***: Lições aprendidas e melhorias para próxima sprint;
- ***Product Backlog Item*:** Demandas a serem realizadas para entrega do projeto;
- ***Tasks***: Sub-demandas que em conjunto integra o Product Backlog Item;
- ***Planning***: Definição de quais Product Backlog Item entrara na sprint;

Product Backlog Itens

- PBI - Modelagem do banco de dados
  - *Task* - Modelagem do banco;
  - *Task* - Criação banco RDS na AWS;
- PBI - Projeto backend
  - *Task* - Estruturação projeto utilizando arquitetura hexagonal, java 11 e spring boot;
  - *Task* - Integração com site G1;
  - *Task* - Filtragem de texto;
  - *Task* - Conexão com banco de dados;
  - *Task* - Expor API Rest para frontend;
- PBI - Projeto frontend
  - *Task* - Esrtuturação do projeto utilizando; JavaScript + React;
  - *Task* - Criação das telas;
  - *Task* - Integração com backend;
- PBI - Escrever o projeto de graduação
  - *Task* - Introdução;
  - *Task* - Fundamentação teórica;
  - *Task* - Desenvolvimento;
  - *Task* - Conclusão;
- PBI - Implantação cloud AWS
  - *Task* - Criação de conta gratuita;

  - *Task* - Configuração máquina EC2;
  - *Task* - Configuraçào AWS Amplify;
  - *Task* - Configuração Route DNS;
  - *Task* - Configuraçào API GATEWAY;

- PBI - Expor quiz
  - *Task* - Divulgar link do quiz;

- PBI - Levantamento dos resultados obtidos
  - *Task* - Apresentação;

3.1 Sprint 1

Para o início da primeira sprint, foram estabelecidas as atividades de modelagem de banco de dados e o início da escrita deste projeto, conforme apresentado na Figura 9.

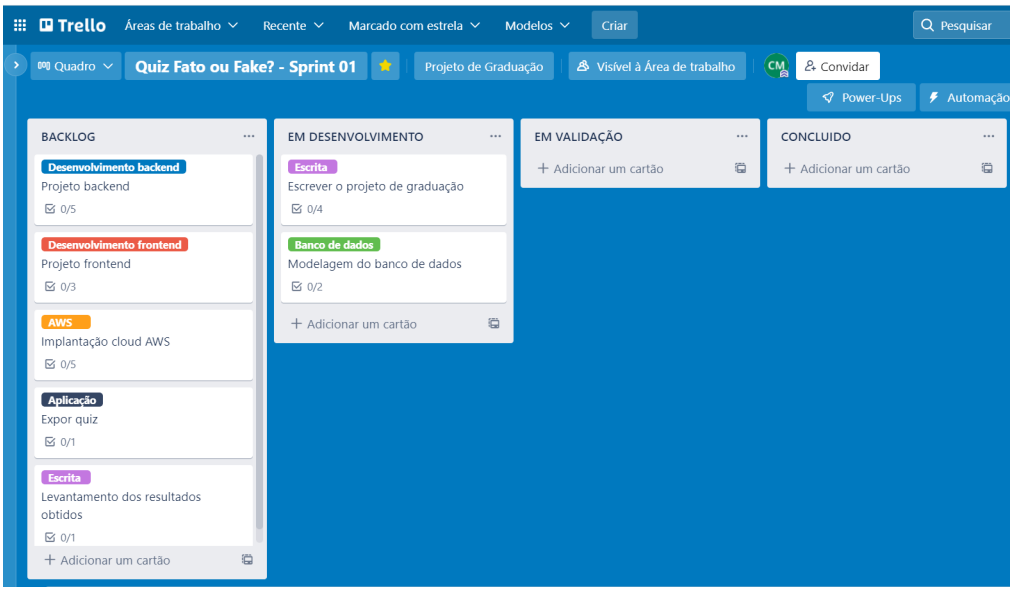

Figura 9 – Board do ínicio da primeira sprint

Fonte: Próprio autor

3.1.1 Modelagem do banco de dados

Realizando o levantamento das informações que deveriam ser armazenadas no banco de dados e o relacionamento entre essas informações, foi decidido utilizar o banco de dados relacional Mysql para armazenamento e requisições. Como serviço que disponibiliza a criação de banco de dados em nuvem foi utilizado o recurso Amazon Relational Database Service (RDS). Então, o Driagrama de Entidade-Relacionamento do banco de dados relacional, ficou como apresentado na Figura 10.

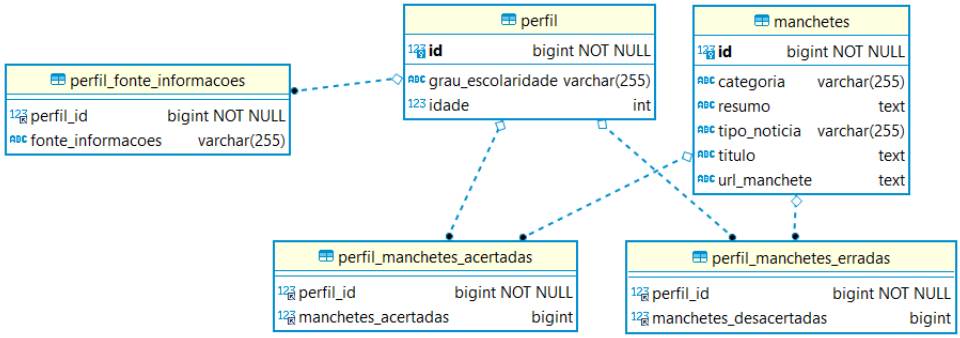

Figura 10 – Diagrama Entidade-Relacionamento do banco de dados do projeto

Fonte: Próprio autor

Foi realizada a busca de manchetes ao site do G1, sendo que, inicialmente foram levantadas quais as informações seriam mantidas sobre cada manchete para então disponibilizá-las aos participantes em formato de quiz. Como resultado, os seguintes itens foram selecionados:

url_manchete: refere-se a url da manchete no site do G1;

titulo: refere-se ao título da manchete no site do G1;

tipo_noticia: refere-se ao tipo FATO ou FAKE da manchete no site do G1;

resumo: refere-se ao primeiro parágrafo da manchete no site do G1;

categoria: refere-se a categoria (saúde, política, economia) da manchete.

Como a ideia era coletar de maneira anônima as informações dos participantes e suas respostas sobre as manchetes, para só então realizar um mapeamento dos perfis mais suscetíveis a serem ludibriados por uma fake news, definiu-se que as informações a serem armazenadas em relação a cada perfil seriam:

grau_escolaridade: refere-se ao nível de escolaridade do participante, podendo ser: Ensino Fundamental Incompleto/Completo, Ensino Médio Incompleto/Completo, Ensino Superior Incompleto/Completo;

idade: refere-se a idade do participante, a partir de 10 anos de idade;

fonte_informacoes: refere-se ao local em que participante geralmente busca informações no dia a dia, podendo ser: redes sociais, TV, Rádio, Jornais/Revistas, Portal Notícias ou Podcasts.

3.2 Sprints 2 e 3

Conforme apresentado nas Figuras 11 e 15, a segunda e a terceira sprints mantiveram a continuação de escrita do projeto tendo sido realizadas as implementações com desenvolvimento *backend* e *frontend*.

3.2.1 Implementação

Neste projeto a implementação foi separada em duas frentes. A primeira é o backend, responsável pela conexão com o banco de dados, integração com o site do G1, filtragem das informações e exposição de serviços através de APIs. A segunda é o *frontend*, responsável por buscar informações através de APIs disponibilizadas pelo *backend*, coletar informações dos participantes e disponibilizar todo o quiz de forma visual através de uma página Web.

Essa separação traz consigo alguns benefícios, por exemplo, códigos separados e independentes de tecnologia. Pode-se mudar a tecnologia de um sem afetar o outro, é possível hospedar o *frontend* em um servidor e o *backend* em outro servidor e diversos desenvolvedores podem trabalhar em projetos diferentes com base em suas habilidades, diminuindo-se assim o conflito entre as implementações.

Para toda implementação, foi empregado um software de controle de versão distribuído, o Git. O controle de versão é uma maneira de salvar as alterações ao longo do tempo sem substituir as versões anteriores (COMMUNITY, 2021). Para interagir com o Git, fez-se uso da plataforma de hospedagem de código-fonte e arquivos com controle de versão GitHub.

3.2.2 Sprint 2

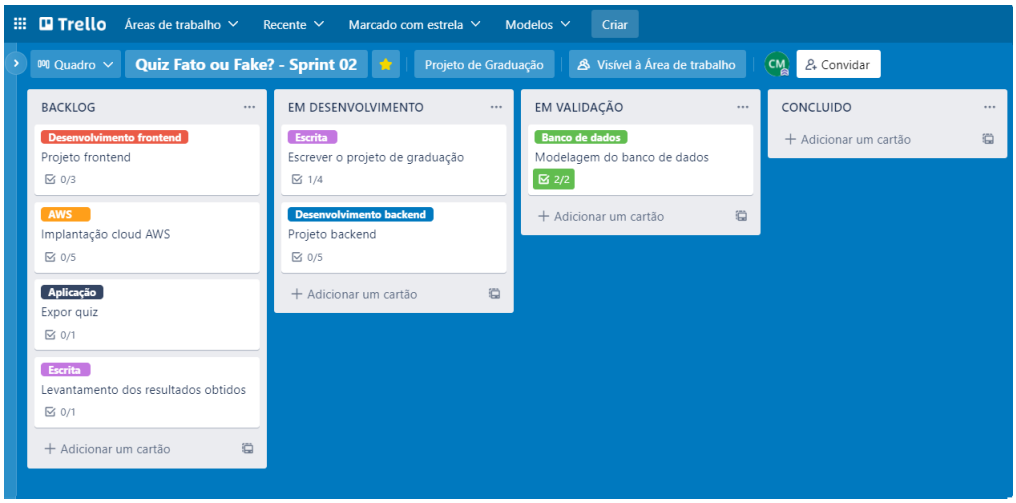

Figura 11 – Início da segunda Sprint

Fonte: Próprio autor

3.2.2.1 Desenvolvimento *backend*

A construção do sistema *backend* aderiu-se à arquitetura Hexagonal, orientada a camadas. Essa modularização foi realizada utilizando-se as seguintes tecnologias: Java 11 como linguagem de programação orientada a objetos; *framework* Spring Boot com o objetivo de facilitar o processo de construção de aplicações Java e a ferramenta de automação de compilação, Apache Maven.

Podem existir diversas tecnologias de entrada para aplicação como, filas SQS, streams DynamoDB, SOAP, etc. Contudo, para o nosso projeto, o único adaptador de entrada existente foi REST, conforme apresentado na Figura 12. Desta feita, foi necessária a criação de dois Controllers para este adaptador, sendo ambos desenvolvidos utilizando-se o formato URI e métodos do padrão HTTP, como GET, POST, PUT e DELETE.

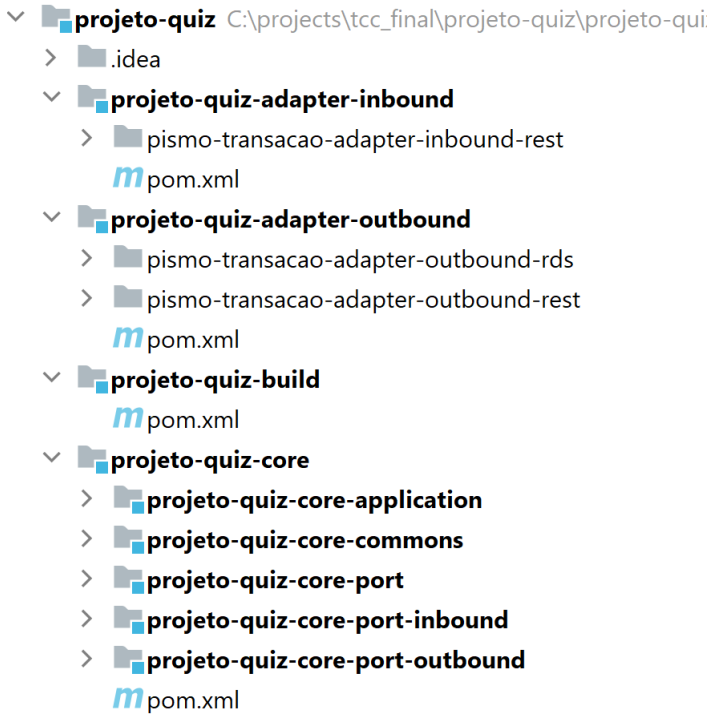

Figura 12 – Estrutura do projeto utilizando Arquitetura Hexagonal

Fonte: Próprio autor

O primeiro Controller, com caminho "/manchetes/G1", foi desenvolvido com o intuito de se integrar com o site do G1 para buscar e salvar as manchetes na base de dados. Possuindo dois métodos HTTP POST, sendo um responsável por buscar/salvar manchetes verdadeiras com caminho "/fatos" e o outro responsável por buscar/salvar manchetes falsas com caminho "/fakes". Assim, foram buscadas, tratadas e armazenadas 83 manchetes no total (51 Fakes e 32 Fatos).

O ciclo para buscar/salvar fatos, inicia-se no adaptador de entrada REST com método HTTP POST e caminho "/manchetes/G1/fatos". O mesmo é reconhecido pela porta de entrada, e então encaminhado para o domínio da aplicação, onde primeiramente é realizada a leitura página a página do site do G1 e a cada página buscada, é realizado um filtro em cima de todas as manchetes encontradas daquela página antes de armazená-la no banco de dados. Essa filtragem é responsável por formatar o título e primeiro parágrafo e categorizar a notícia como Fato.

Para a realização das filtragens de texto, foi aplicado o padrão de projeto strategy, pois foi criada a classe abstract Manchetes que tem como método de busca Manchetes() e duas classes concretas de manchetes, respectivamente, ManchoteFatos e ManchoteFakes, onde as mesmas implementam o método buscar manchetes, porém usando parâmetros e regras diferentes.

Já o ciclo para buscar/salvar fakes, também se inicia no adaptador de entrada

REST com método HTTP POST e caminho "/manchetes/G1/fakes". Também reconhecido pela porta de entrada, é então encaminhado para o domínio da aplicação, onde segue o mesmo fluxo de leitura de página para a busca de fatos. Porém, a coleta de manchetes fakes é um pouco mais detalhada do que a de manchetes fatos, isto porque os títulos das manchetes fakes sempre iniciam-se com "É FAKE que", enquanto as manchetes fatos, não possuem este início.

Assim, para este ciclo é realizada uma filtragem de texto um pouco mais completa antes de armazená-la no banco. Primeiramente, é retirado do título este início de "É FAKE que" e substituído por vazio. Com essa retirada, também é necessário colocar o primeiro caractere do título com letra maiúscula. Outra filtragem é no primeiro parágrafo, onde, alguns casos se inicia com termos que já apresentar ser fakes, exemplos, "Circula nas redes" , "Circula uma foto", todas essas informações são identificadas e corrigidas de forma automática. Após o armazenamento no banco, foi necessário validá-las individualmente para garantir a coerência.

Conforme explanado acima, para ambos os ciclos do primeiro *Controller*, há dois adaptadores de saídas. O primeiro adaptador de saída vem através da porta de saída, e é responsável por se comunicar com o site do G1 através do adaptador de saída de tecnologia REST com método HTTP GET. O segundo adaptador de saída, que também vem através da porta de saída, é responsável por armazenar as informações no banco de dados MySQL. Essa comunicação entre aplicação e banco de dados é feita através do JpaRepository, que é uma extensão do repositório JPA (*Java Persistence API*). Ela contém uma API completa para operações de consulta, inserção, atualização e remoção no banco de dados. O JpaRepository é fornecido pelo *framework* Spring Boot.

O segundo *Controller*, com caminho "/quiz", foi desenvolvido com o objetivo de fornecer e receber informações do frontend, possuindo também dois métodos, sendo o primeiro, um método HTTP GET com caminho "/manchetes", responsável por fornecer ao frontend as manchetes tratadas e armazenadas no banco de dados. O retorno dessa chamada é em formato JSON(*JavaScript Object Notation*). A resposta, conforme apresentado na Figura 13, é sempre uma lista contendo dez manchetes buscadas de forma aleatórias, cinco fatos e cinco fakes, para garantir um padrão na análise dos resultados de cada participante.

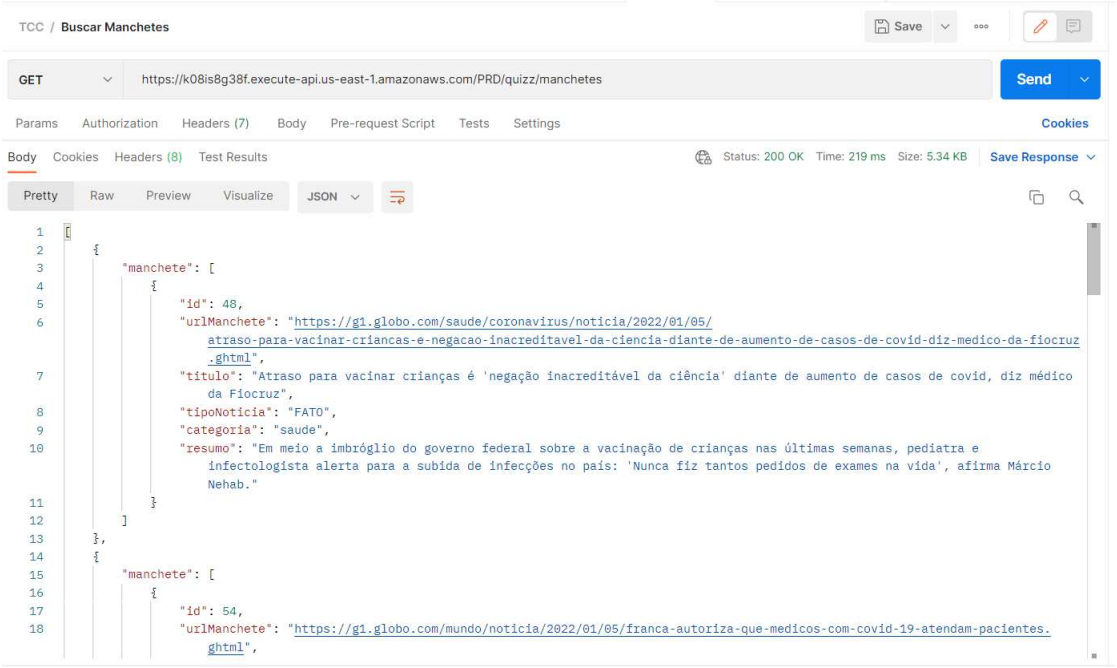

Figura 13 – Chamada ao backend para busca de manchetes do banco de dados

Fonte: Próprio autor

O segundo método, é HTTP POST com caminho "/perfil", responsável receber do frontend os dados de perfil de cada participante e suas respectivas respostas do quiz. O envio dessas informações são realizadas pelo body da chamada e também possui o formato JSON, conforme apresentado na Figura 14.

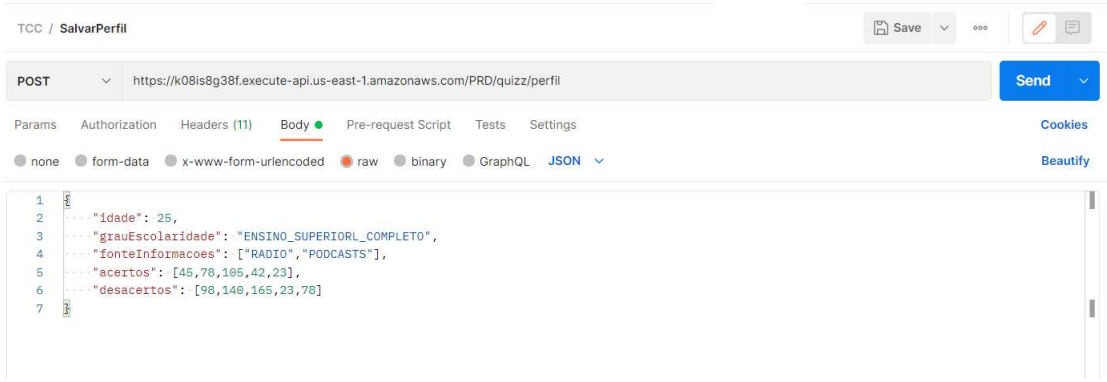

Figura 14 – Chamada ao backend para salvar perfil no banco de dados

Fonte: Próprio autor

A segunda sprint é terminada com sucesso, porém, ainda com status validação, pois, é necessário a conclusão das *PBI's* de desenvolvimento *frontend* e implantação com AWS para realização dos testes fim a fim.

3.2.3 Sprint 3

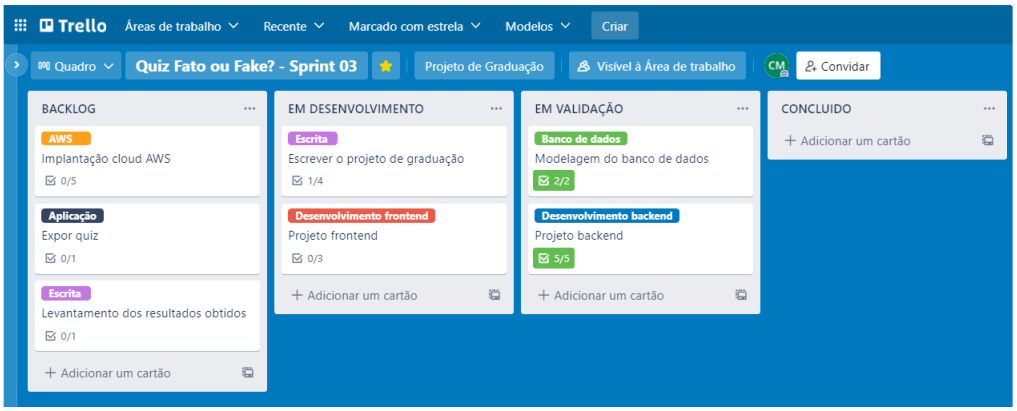

Figura 15 – Inicio da terceira Sprint

Fonte: Próprio autor

3.2.3.1 Desenvolvimento frontend

O módulo responsável pelo lado do cliente, foi desenvolvido utilizando-se a linguagem JavaScript em conjunto com a biblioteca React, para criação de interfaces de usuário. A React utiliza uma extensão alternativa ao JavaScript para descrever seus componentes. Esta sintaxe é chamada de JSX. Basicamente, foi adotada no nosso projeto para facilitar o desenvolvimento e aumentar a velocidade de entrega do frontend, isso porque, o JSX é uma linguagem desenvolvida com uma junção de HTML e JS, assim, já temos diversos métodos prontos. A estrutura da aplicação, conforme apresentado na Figura 16, foi dividida em três partes principais, a Home, Quiz e o Finish.

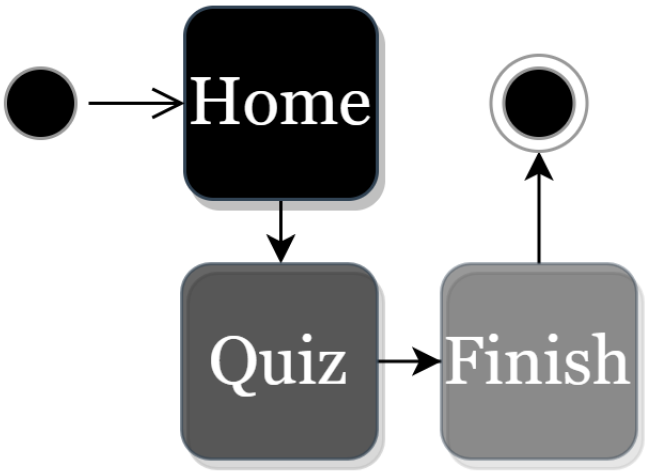

Figura 16 – Ciclo de vida da aplicação Web

Fonte: Próprio autor

A tela Home, conforme apresentado na Figura 17, é a tela inicial do Quiz. Ela é composta por um título e um breve texto explicativo do que se trata o quiz. Neste primeiro momento, são coletadas do participante algumas informações, como: idade, grau de escolaridade e fontes onde o participante geralmente busca se manter informado. Estes dados são armazenados através de API Web Storage que fornece mecanismos pelos quais os navegadores podem armazenar pares de chaves/valores (MDN, 2022b). Esses dados são armazenados temporariamente para serem usados na tela Finish.

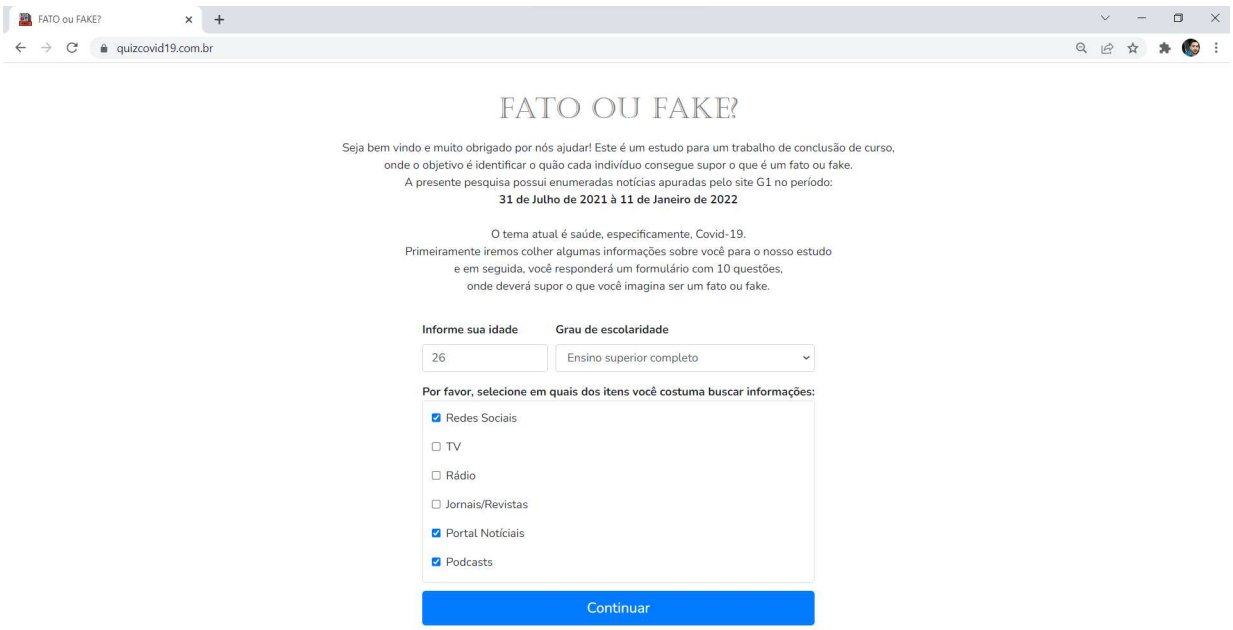

Figura 17 – Tela Home do “Quiz Fato ou Fake?”

Fonte: Próprio autor

A próxima tela é a do Quiz, conforme apersentado na Figura 18. Antes dessa tela ser apresentada, é realizada no servidor backend através do caminho “/quiz/manchete” uma busca de manchetes para a montagem da tela. Essa chamada retorna com uma lista com 10 notícias, sendo que cada manchete pode ser Fato ou Fake, assim, a cada resposta do participante é possível informar se o mesmo errou ou acertou aquela notícia, conforme as Figuras 19 e 20.

Figura 18 – Tela Quiz do "Quiz Fato ou Fake?"

Fonte: Próprio autor

Como dito anteriormente, é reconhecido pela aplicação se a manchete é FATO ou FAKE. Assim, se por um acaso o participante errar, ou seja, registrar FATO em uma notícia FAKE ou o contrário, será demonstrado se a manchete é FATO ou FAKE e caso ele tenha errado a borda ficará vermelha.

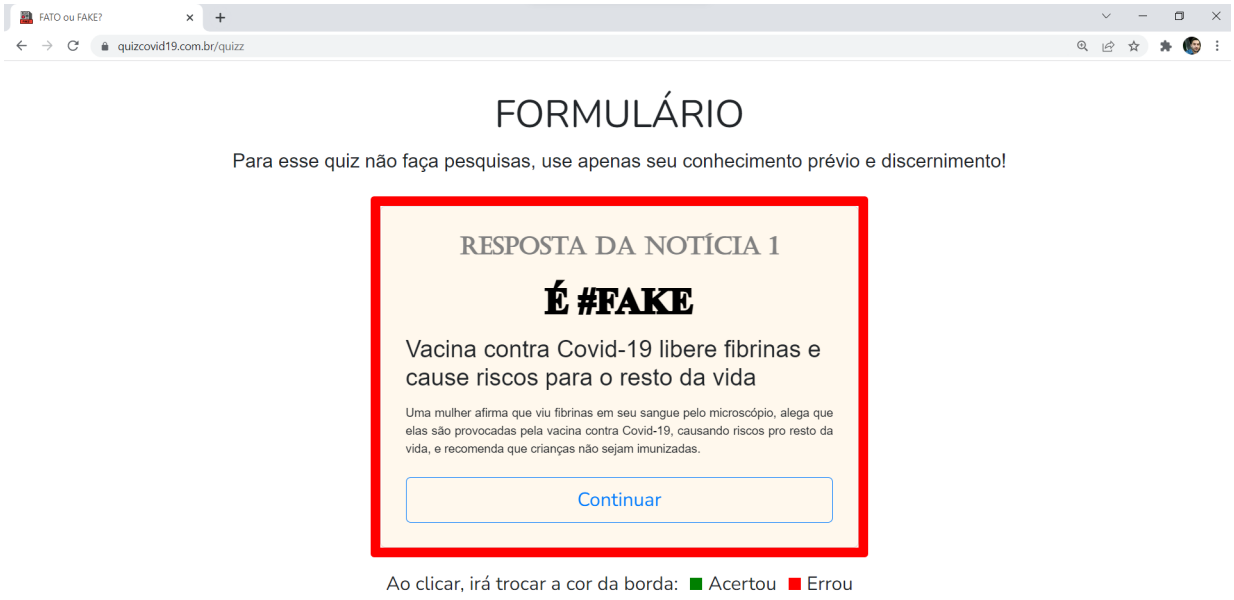

Figura 19 – Tela Quiz do "Quiz Fato ou Fake?" quando participante não acerta

Fonte: Próprio autor

O mesmo acontece quando o participante acerta a manchete. Desta feita, também será demonstrado se o participante acertou ou errou se a manchete em questão era Fato ou Fake, assim, tendo acertado a borda ficará na cor verde.

Figura 20 – Tela Quiz do "Quiz Fato ou Fake?" quando participante acerta

Fonte: Próprio autor

Para cada participante, é criado um objeto no Web Storage. Este objeto possui os dados iniciais que foram coletados na tela Home. Durante o quiz são criadas duas listas: acertos e desacertos. A cada manchete respondida pelo participante, é adicionado o id da manchete em alguma dessas listas. Por exemplo, foram fornecidas as manchetes com id: 14, 22, 25, 31, 36, 44, 45, 46, 70 e 75. O participante acertou as de id [36, 44, 75] e errou as de id [14, 22, 25, 31, 45, 46, 70]. Ao final do quiz, é enviado para o servidor backend através do caminho "/quiz/perfil" as informações do participante, que se encontra no Web Storage em conjunto com as listas de acertos e erros. Os participantes recebem como resultado, conforme apresentado na Figura 21, uma lista com as manchetes que o mesmo acertou e errou, e o link de cada uma delas no site do G1, afim de conscientizá-lo.

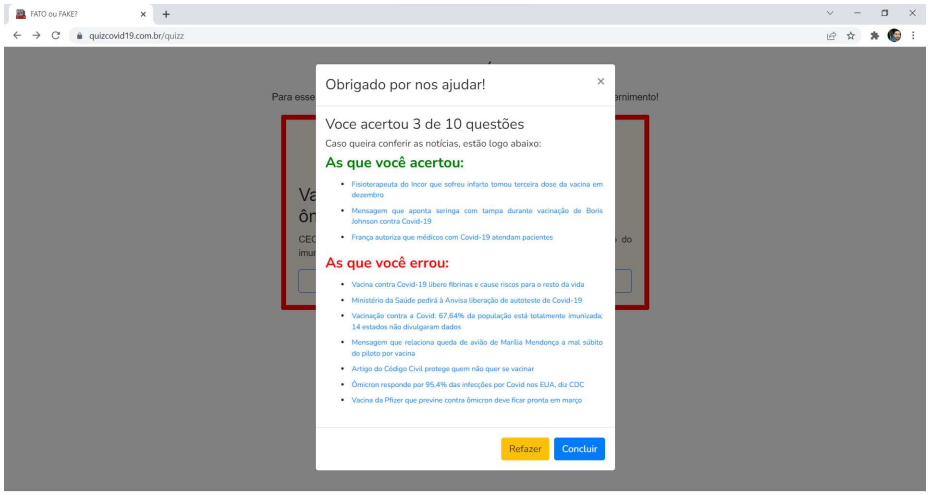

Figura 21 – Tela Quiz do "Quiz Fato ou Fake?" após finalizar respostas

Fonte: Próprio autor

Por fim, é apresentada a tela Finish, conforme demonstrado na Figura 22, onde contém um agradecimento ao participante por ter contribuído com a pesquisa.

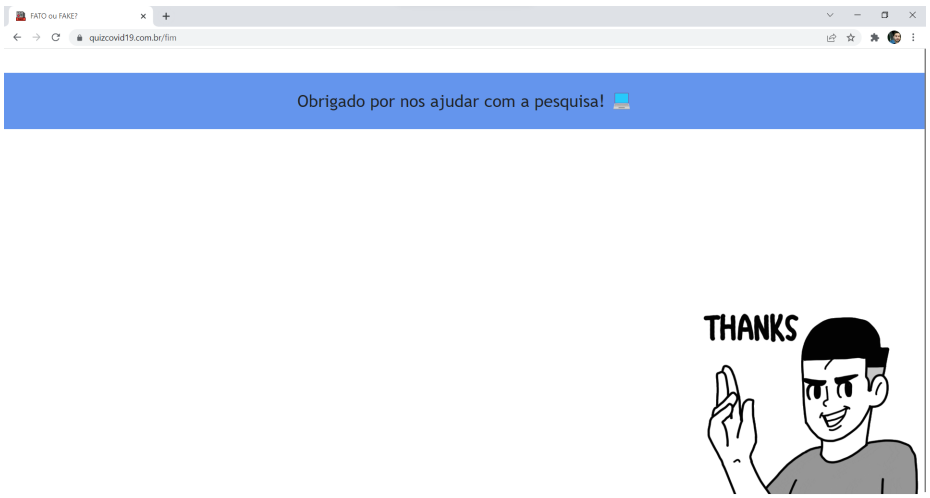

Figura 22 – Tela Finish do "Quiz Fato ou Fake?"

Fonte: Próprio autor

3.3 Sprint 4

A quarta Sprint, conforme a Figura 23, iniciou-se com as Pbi's para realização da implantação e integração do cliente-servidor.

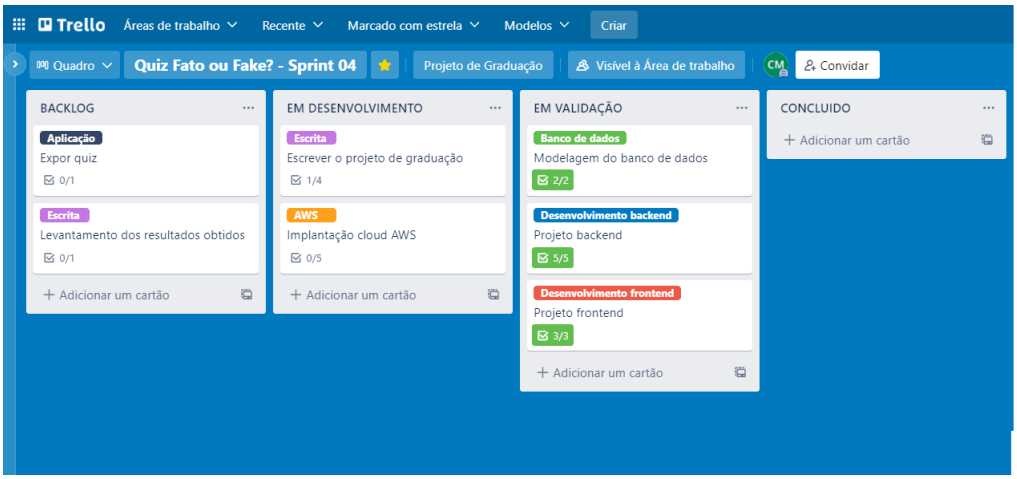

Figura 23 – Inicio da quarta Sprint

Fonte: Próprio autor

Como dito no capítulo "Fundamentação Teórica", seção 2.1.8, neste projeto foi adotada a utilização de uma plataforma de computação em nuvem. Os principais motivos para essa utilização foram: centralizar em um mesmo ambiente o cliente-servidor, manter o ambiente online a todo momento, recursos e serviços disponíveis pela plataforma e gratuidade para uso de alguns serviços, conforme apresentado na Figura 24. Com todos esses critérios, estabeleceu-se trabalhar com a Amazon Web Services (AWS).

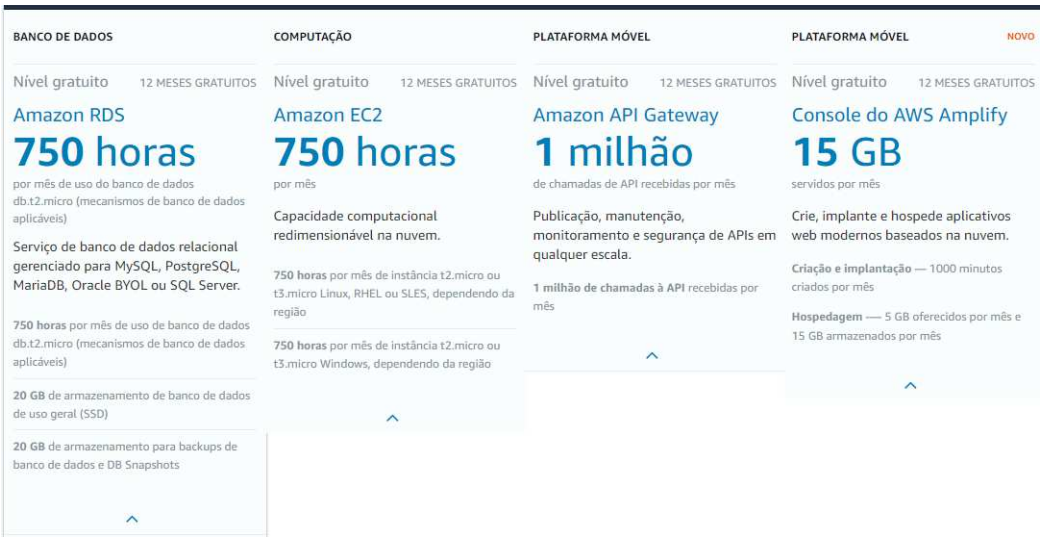

Figura 24 – Alguns serviços disponíveis na conta gratuita AWS

Fonte: (AMAZON, 2022)

3.3.1 Implantação

AWS possui diversos datacenters espalhados pelo mundo, conforme a Figura 25, chamados de zonas de disponibilidade. Assim, a flexibilidade de escolher geograficamente o local onde deseja armazenar as informações, é possível. Alguns serviços são disponibilizados por zonas, enquanto para outros não é necessário definir a zona. Para esse projeto o RDS está rodando na zona us-east-1 (Norte da Virgínia). Outros serviços como Amazon S3, roda em zona Global.

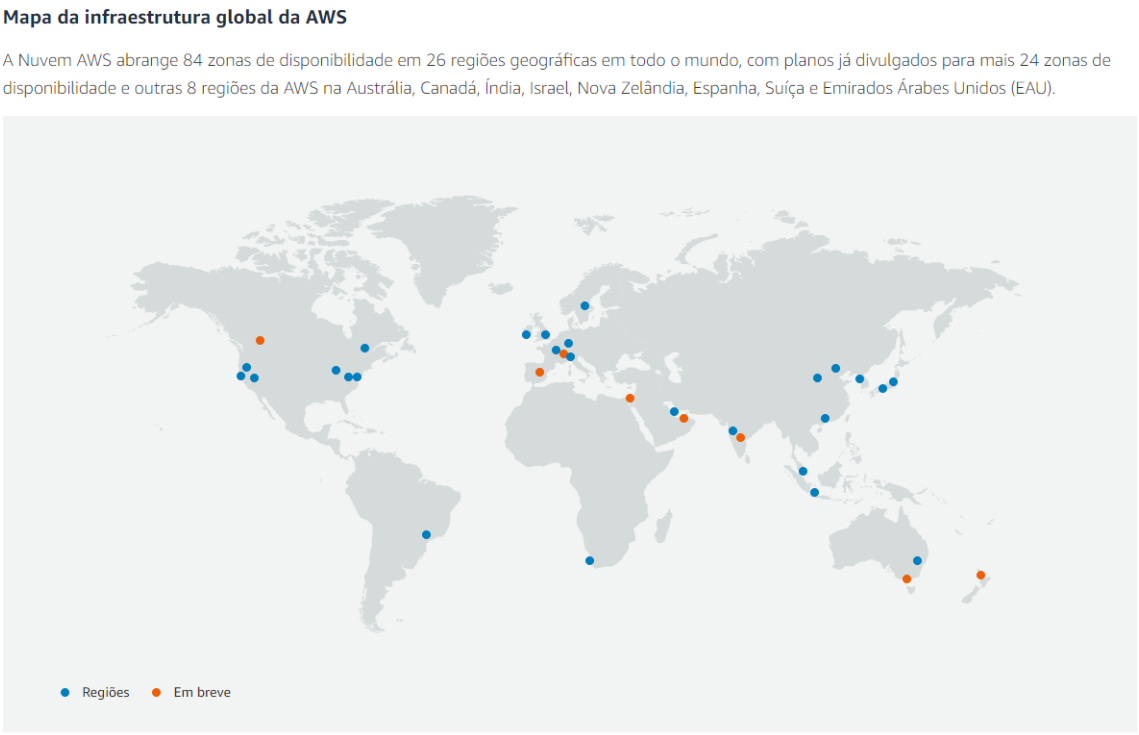

Figura 25 – Mapa da infraestrutura global da AWS

Fonte: (AMAZON, 2022)

Com a modelagem do banco de dados já realizada, foi iniciado o processo de criação do banco de dados dentro da AWS. Foi utilizado o serviço RDS para criação do banco de dados. O processo de criação é bem intuitivo. Na primeira etapa foi definido o tipo de mecanismo do banco, como dito anteriormente em 2.1.3, foi utilizado o MySQL versão 8.0.27, conforme apresentado na Figura 26.

A segunda etapa foi a escolha do modelo do banco de dados, dividido em três tipos: Produção, Desenvolvimento/Teste e Nível gratuito. Foi utilizado o modelo de Nível gratuito por já atender a nossa demanda e por não ter nenhum custo. Logo em seguida, foram definidas as configurações do banco, como, usuário e senha, nome do database, público ou privado e zona de disponibilidade.

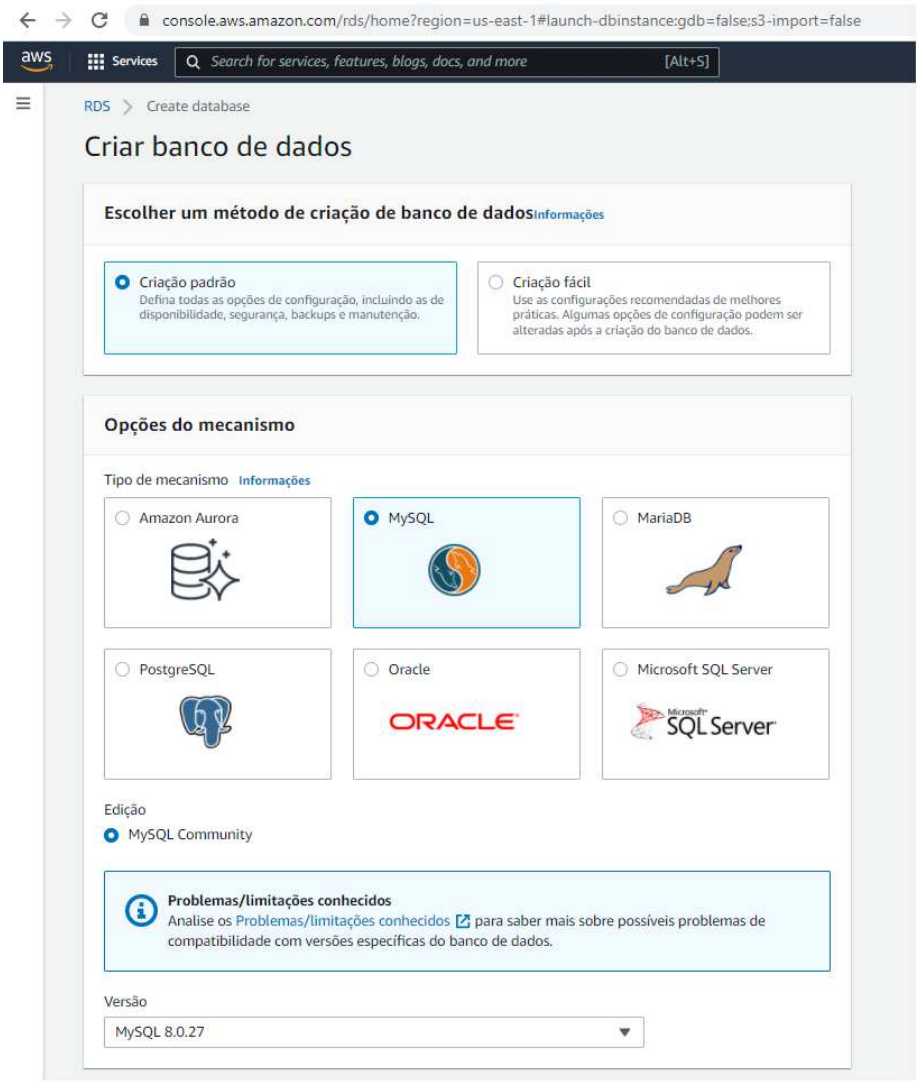

Figura 26 – Exemplo inicial de criação de banco de dados RDS

Fonte: (AMAZON, 2022)

Com a instância do banco de dados já criada, foi possível registrar no backend os seguintes valores para conexão com o banco de dados:

- **host**= *fakenews.cnynbvymjamd.us-east-1.rds.amazonaws.com*
- **port**= *3306*
- **database**=*quizz*

A criação das tabelas com suas colunas, tipos de colunas, chaves primárias e chaves secundárias foram realizadas através do código backend, por meio do pacote JavaX Persistence, que possui operações de mapeamento para banco de dados.

```
1  import javax.persistence.*;
2  import java.io.Serializable;
3
4  @Entity
5  @Table(name = "MANCHETES")
6  @SequenceGenerator(name = "MANCHETES_SEQ", sequenceName = "MANCHETES_SEQ",
       initialValue = 1, allocationSize = 1)
7  public class ManchetesEntity implements Serializable {
8
9      @Id
10     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "
           MANCHETES_SEQ")
11     @Column(name = "id")
12     private Long id;
13
14     @Column(name = "url_manchete", columnDefinition = "TEXT")
15     private String urlManchete;
16
17     @Column(name = "titulo", columnDefinition = "TEXT")
18     private String titulo;
19
20     @Column(name = "tipo_noticia")
21     private String tipoNoticia;
22
23     @Column(name = "categoria")
24     private String categoria;
25
26     @Column(name = "resumo", columnDefinition = "TEXT")
27     private String resumo;
28 }
```

Como a aplicação backend é executada através de um arquivo com extensão .jar, é fundamental que a máquina da aplicação contenha os requisitos necessários para a execução. Utilizamos Amazon EC2 para a criação de uma instância contendo as ferramentas: JDK 11, Maven e MySQL para a execução da aplicação. Para a criação da instância EC2 ficou definido o sistema operacional Ubuntu Server 20.04 LTS, conforme apresentado na Figura 27, contudo, é possível escolher outros sistemas operacionais, como Microsoft Windows Server 2019, macOS Monterey 12.2.1, Red Hat Enterprise Linux 8, Amazon Linux 2 AMI, entre outros.

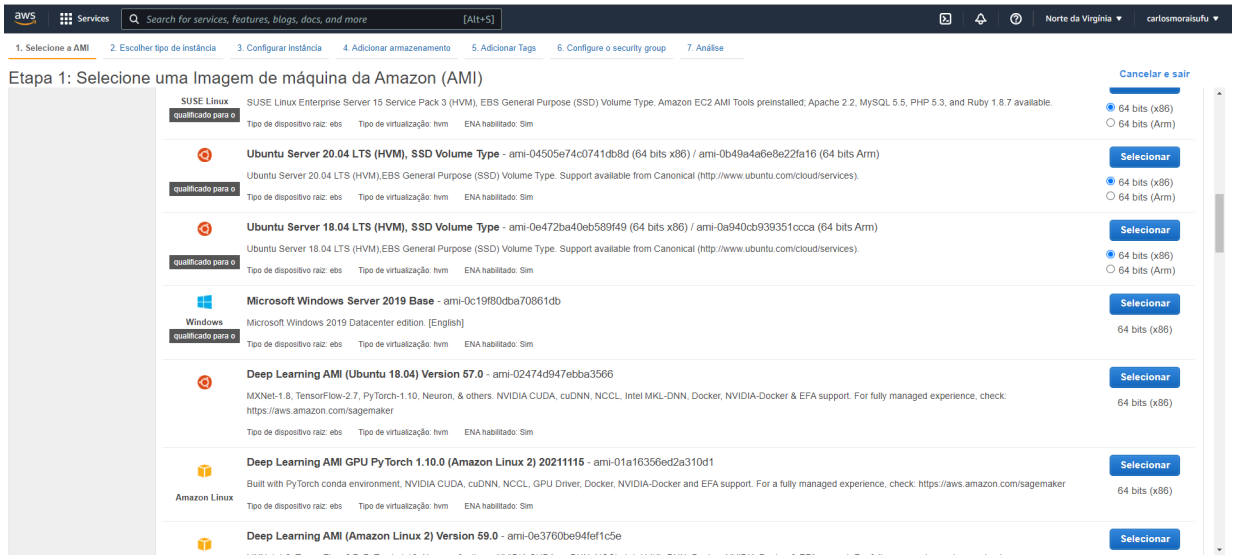

Figura 27 – Exemplo criação de instância na Amazon EC2

Fonte: (AMAZON, 2022)

Com a criação da instância EC2 é definida uma chave.pem. Essa chave é utilizada para a conexão com a instância que também recebe um identificador que em conjunto com a chave.pem é possível acessá-la. Para este projeto a instância possui as seguintes informações:

- **chave**=*aw-spring-tcc-fake-news.pem*
- **instância**=*ec2-35-172-134-198.compute-1.amazonaws.com*

Com a instância online, foi necessário seguir alguns passos como, enviar o arquivo jar para a instância, se conectar com a instância, instalar as ferramentas necessárias e executar o jar da aplicação backend, como demonstrado no script abaixo:

```
1  carlosmorais@local
2  $ chmod 400 aw-spring-tcc-fake-news.pem
3
4  carlosmorais@local
5  $ scp -i aw-spring-tcc-fake-news.pem tcc-0.0.1-SNAPSHOT.jar ubuntu@ec2
       -35-172-134-198.compute-1.amazonaws.com:~
6
7  carlosmorais@local
8  $ ssh -i aw-spring-tcc-fake-news.pem ubuntu@ec2-35-172-134-198.compute-1.
       amazonaws.com
9
10 ubuntu@ip-172-31-88-198:~$ sudo apt update
11 ubuntu@ip-172-31-88-198:~$ sudo apt install maven
12 ubuntu@ip-172-31-88-198:~$ sudo apt install default-jdk
13 ubuntu@ip-172-31-88-198:~$ sudo apt install mysql-server
14 ubuntu@ip-172-31-88-198:~$ nohup java -jar tcc-0.0.1-SNAPSHOT.jar &
```

A parte final da implantação foi a hospedagem do frontend (cliente) que se encontra no repositório do GitHub[3], já que o backend (servidor) está online. Foi utilizado o Amazon Amplify, que oferece um fluxo CI/CD para implantar e hospedar aplicações. Quando conectado a um repositório Git, o Amplify determina automaticamente as configurações de compilação para a estrutura do front-end e implanta automaticamente as atualizações a cada deploy.

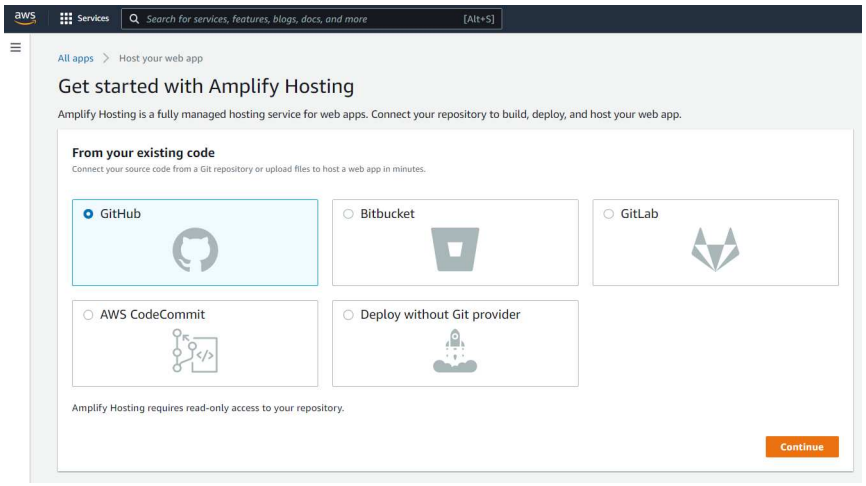

Figura 28 – Exemplo de criação de aplicação no AWS Amplify

Fonte: (AMAZON, 2022)

Conforme apresentado na Figura 28, foi iniciada uma aplicação no Amplify, onde é necessária a conexão com algum repositório, GitHub, BitBucket[4], GitLab[5] ou CodeCommit[6]. Neste caso foi utilizado o repositório onde se encontra o frontend, o GitHub, sendo então definidas as configurações de compilação. Após o término da configuração, o processo foi encaminhando para esteira, onde foi realizada a implantação e gerado um link aleatório acessível à aplicação Web:

https://master.d2h40h4wvmmjsm.amplifyapp.com

3.3.2 Integração

Dando prosseguimento à quarta Sprint, foi necessário definir os detalhes das aplicações, como:

- Camada API Gateway entre cliente e servidor;
- Nome de domínio a aplicação Web;

[3] *github.com*
[4] *bitbucket.org*
[5] *about.gitlab.com*
[6] *aws.amazon.com/pt/codecommit*

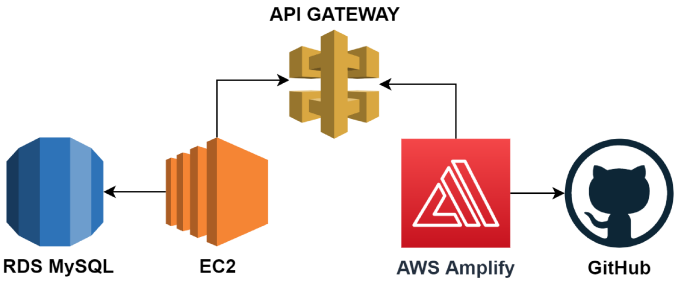

Figura 29 – Estrutura da AWS utilizada nesse projeto

Fonte: Próprio autor

Com todas as configurações realizadas, o cliente consegue expor a aplicação Web através de uma URL aleatória, na qual esta aplicação possui conexão com o servidor, entretanto, não é uma boa prática o cliente ter contato direto com o servidor. Sendo assim, foi decidido implantar um gateway, conforme a Figura 29, que serve como intermediário entre a aplicação cliente-servidor. Foi utilizado, o recurso que a AWS fornece, API GATEWAY. Com essa utilização, é possível estabelecer uma autenticação necessária do cliente ao servidor, aumentando assim a segurança da invocação ao servidor. Também é possível definir um contrato de entrada para o cliente, antecipando uma validação de contrato no servidor.

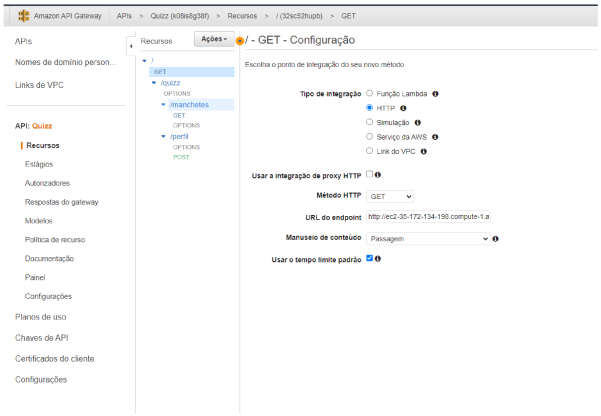

Figura 30 – Exemplo de criação de método GET no API GATEWAY

Fonte: Próprio autor

Para realizar essa configuração, foi necessário fornecer o caminho original de acesso ao backend, no caso, composto pelo identificador da instância EC2 da AWS, mais o domínio da AWS e o caminho do REST no backend, conforme apresentado na Figura 30. A partir de então, o cliente passa a realizar a seguinte chamada para buscar as manchetes no servidor, de

GET http://ec2-35-172-134-198.compute-1.amazonaws.com:8080/quizz/manchetes

para

```
GET https://k08is8g38f.execute-api.us-east-1.amazonaws.com/PRD/quizz/manchetes
```

Outro detalhe tratado foi em relação a URL da aplicação Web, uma vez que, por se tratar de uma URL aleatória não ficaria bem visível divulgar ao participante, pois, era de difícil leitura e não passava nenhuma credibilidade do site. Sendo assim, foi aplicada a hospedagem de domínio e para isso foi comprado o domínio de nome "quizcovid19.com.br" no site de hospedagem GoDaddy[7]. Por outro lado, domínios como: .tk, .ml, .cf, .ga, entre outros, podem ser encontrados de forma gratuita, contudo, a decisão de pagar por um domínio, foi estabelecida para gerar uma credibilidade maior do site.

Para transformar a URL do Amplify para uma nova URL, foi necessário cadastrar uma Zona de DNS e a AWS fornece este recurso através do serviço Route 53. Assim, foi gerado quatro valores DNS:

- ns-1958.awsdns-52.co.uk
- ns-1220.awsdns-24.org
- ns-359.awsdns-44.com
- ns-932.awsdns-52.net

O próximo passo foi vincular os quatro valores DNS acima demonstrado ao domínio comprado. Para isto, foi necessário acessar as configurações do domínio comprado dentro do site GoDaddy e cadastrar a lista de DNS ao domínio. Por fim, o último passo realizado foi acessar o AWS Amplify, ir nas configurações e alterar a URL de aplicação para "quizcovid19.com.br". Com essas etapas concluídas, o fluxograma da aplicação ficou conforme apresentado na Figura 7.

3.4 Sprint 5

A quinta e última Sprint, teve como demanda a divulgação do Quiz ao público, o cruzamento das informações recebidas, gerar os resultados da análise quantitativa dos dados e realizar o término da escrita do projeto de graduação, conforme apresentado na Figura 31.

3.4.1 Cruzamento dos dados

A divulgação do sistema Web ao público foi realizada entre a última semana de Janeiro de 2022 e a primeira semana de Fevereiro de 2022. Os meios de divulgação do "Quiz

[7] *godaddy.com*

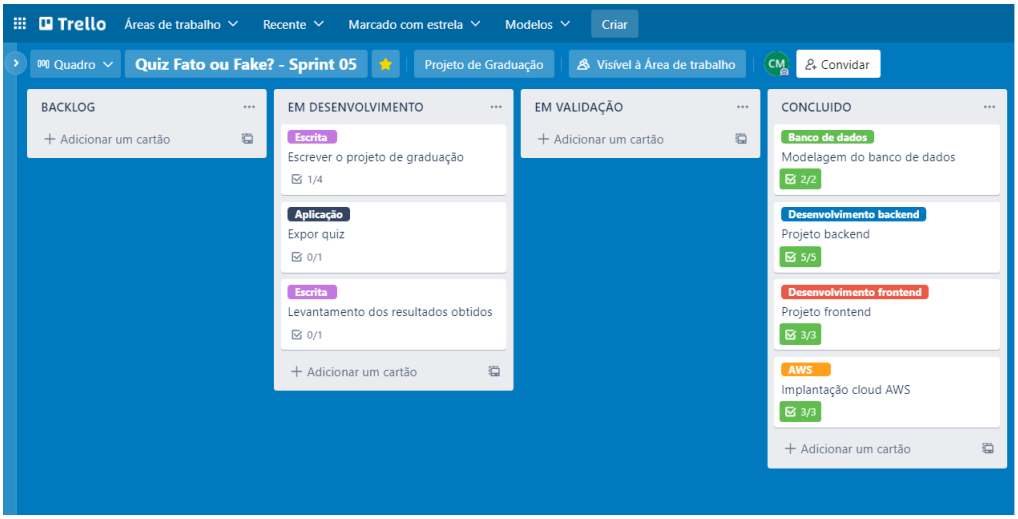

Figura 31 – Inicio da quinta e última Sprint

Fato ou Fake?", foi através de compartilhamento em Redes Sociais, como, WhatsApp[8], Instagram[9] e Facebook[10], e por software de comunicação, como, Google Chat[11].

Após o encerramento do período estabelecido, foram construídas consultas SQL diretamente no banco de dados para recolher as informações. Para a realização dessas buscas, foi utilizado o software/ferramenta gratuito de administração de banco de dados (DBEAVER, 2022) e sua conexão foi realizada com o banco de dados RDS MySQL desenvolvido para este projeto, para então executar as consultas necessárias para o cruzamento das informações.

Foram realizadas buscas como, quantidade de respostas obtidas, quantidade de acertos e erros, quantidade de acertos e erros pelo grau de escolaridade ou fonte de busca de informações ou por idade, relação entre o grau de escolaridade e a fonte de busca de informações, e manchetes que mais teve acertos ou erros. Todos os dados encontrados foram inseridos nas planilhas que constam em anexo a este projeto, tendo sido realizado o levantamento das informações em porcentagem com os respectivos gráficos.

3.4.2 Resultados obtidos no Teste de Sistema

Em um total, 131 participantes responderam ao "Quiz Fato ou Fake?", totalizando 1310 manchetes avaliadas, lembrando que cada participante analisa 10 manchetes. Os resultados obtidos foram separados em diversas perspectivas.

1) Relação de respostas por faixa etária

[8] *whatsapp.com*
[9] *instagram.com*
[10] *facebook.com*
[11] *chat.google.com*

Separamos os participantes em quatro faixas etárias, são elas:

- dos 15 aos 24 anos, total de 31 participantes;
- dos 25 aos 34 anos, total de 68 participantes;
- dos 35 aos 44 anos, total de 19 participantes;
- acima dos 44 anos, total de 13 participantes;

Após essa separação, foi mapeada a quantidade de manchetes acertadas e erradas por faixa etária, sendo realizada a transformação para porcentagem da relação entre quantidade de respostas e quantidade de acertos/erros para cada faixa etária, conforme os resultados apresentados na Figura 32.

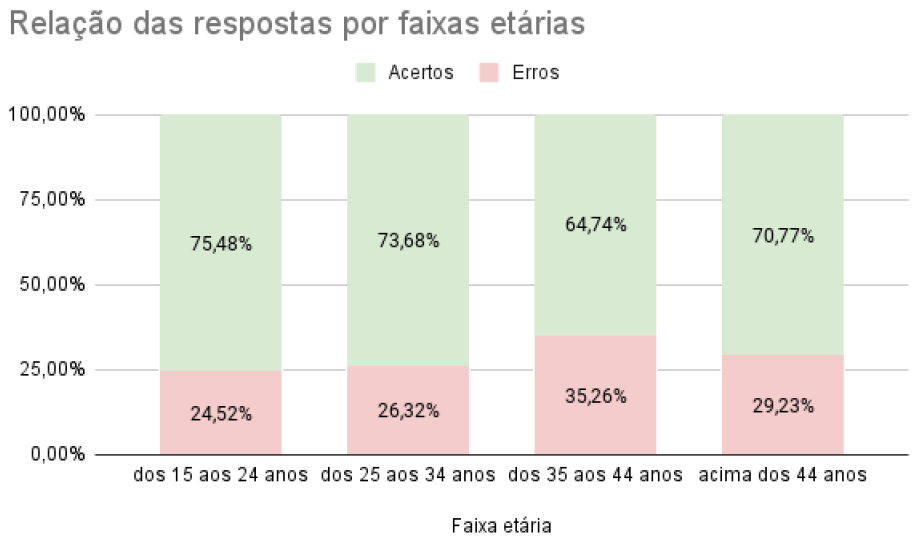

Figura 32 – Resultado: Relação das respostas por faixas etárias

Fonte: Próprio autor

2) Relação de respostas por fonte de informação

Cada participante pode se manter informado sobre o que acontece no mundo em mais de uma fonte de informação/notícia, como por exemplo o participante número 78 que busca informações em Redes Socias, TV e Podcasts. As fontes de informações foram separadas em seis categorias, são elas:

- Jornais/Revistas, total de 23 participantes;
- Podcasts, total de 26 participantes;
- Portal de Notícias, total de 73 participantes;
- Rádio, total de 16 participantes;
- Redes Sociais, total de 100 participantes;
- TV, total de 59 participantes;

Após essa separação, foi mapeada a quantidade de manchetes acertadas e erradas de acordo com a fonte de informação. Desta feita, foi realizada a transformação para porcentagem da relação entre quantidade de respostas e quantidade de acertos/erros para cada fonte de informação, conforme os resultados apresentados na Figura 33.

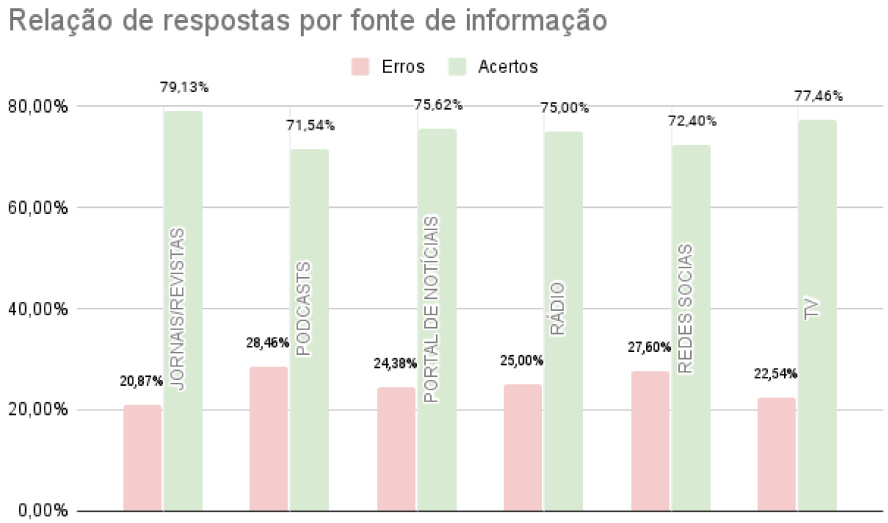

Figura 33 – Resultado: Relação das respostas por fontes de informação

Fonte: Próprio autor

3) Relação de respostas por escolaridade

O grau de escolaridade foi dividido em seis categorias, são elas:

- Fundamental Incompleto, total de 2 participantes;
- Fundamental Completo, total de 1 participante;
- Ensino Médio Incompleto, total de 8 participantes;
- Ensino Médio Incompleto, total de 25 participantes;
- Superior Incompleto, total de 29 participantes;
- Superior Completo, total de 66 participantes;

Como a amostra para Fundamental é muito pequena, foi decidido retirá-la dos resultados. Assim, após o mapeamento da quantidade de manchetes acertadas e erradas pelo grau de escolaridade, foi realizada a transformação para porcentagem da relação entre quantidade de respostas e quantidade de acertos/erros para cada grau de escolaridade, conforme os resultados apresentados na Figura 34.

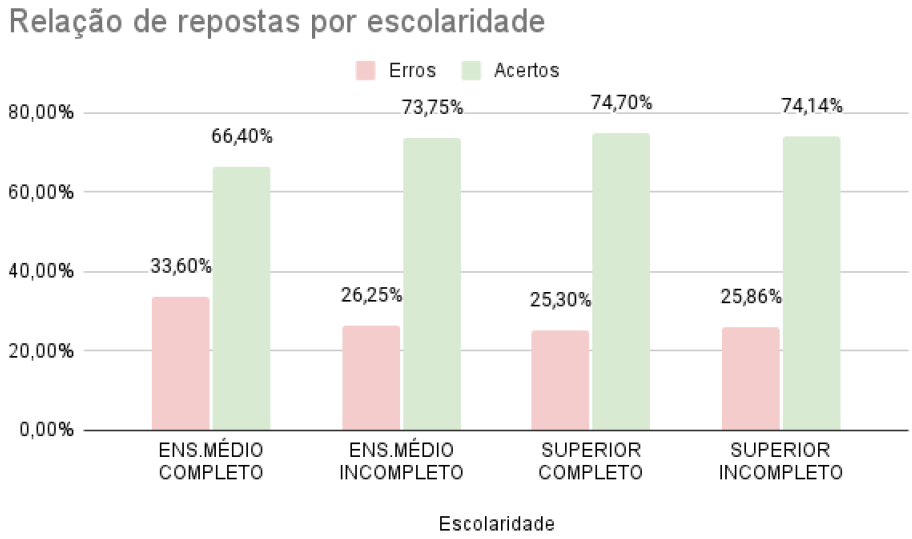

Figura 34 – Resultado: Relação das respostas por grau de escolaridade

Fonte: Próprio autor

4) Relação de grau de escolaridade por fonte de informação

Cada participante pode pertencer a diversos grupos de fonte de informação e ao realizar o cruzamento dos dados e ignorando o grau de escolaridade Fundamental, em razão da pequena quantidade de amostra, chegou-se aos seguintes resultados, conforme apresentados na Figura 35:

Tabela 1 – Relação grau escolaridade x fonte informações

Fonte	Redes sociais	Portal de Notícias	Tv	Podcasts	Rádio	Jornais
ENSINO MÉDIO INCOMPL.	100,00%	12,50%	87,50%	0,00%	12,50%	0,00%
ENSINO MÉDIO COMPLETO	76,00%	48,00%	44,00%	4,00%	8,00%	16,00%
ENSINO SUPERIOR INCOMPL.O	93,10%	44,83%	41,38%	34,48%	10,34%	27,59%
ENSINO SUPERIOR COMPLETO	66,67%	71,21%	40,91%	22,73%	15,15%	16,67%

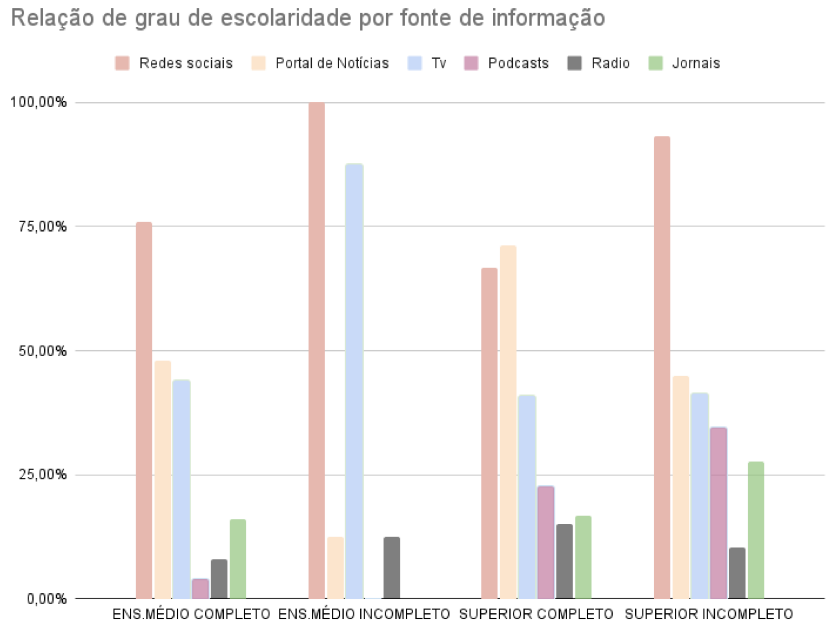

Figura 35 – Resultado: Relação grau de escolaridade por fonte de informação

Fonte: Próprio autor

É possível visualizar dados mais detalhados acessando a planilha[12], como, a relação de participantes que estão inclusos em diversos grupos de fontes de informação em relação ao grau de escolaridade.

[12] *docs.google.com/spreadsheets/d/e/2PACX-1vQ_ruQvbCleJ5sp-59DQqdkJrvFIrqBbOdatwo9rvoQRNxyITk9-DlsTSODoJ6Zfg/pubhtml*

5) Total de acertos em relação ao total de erros

Em relação à quantidade total de 1310 manchetes respondidas, se divergem em 950 acertos e 360 erros, respectivamente, 72,52% e 27,48%, conforme apresentados na Figura 36:

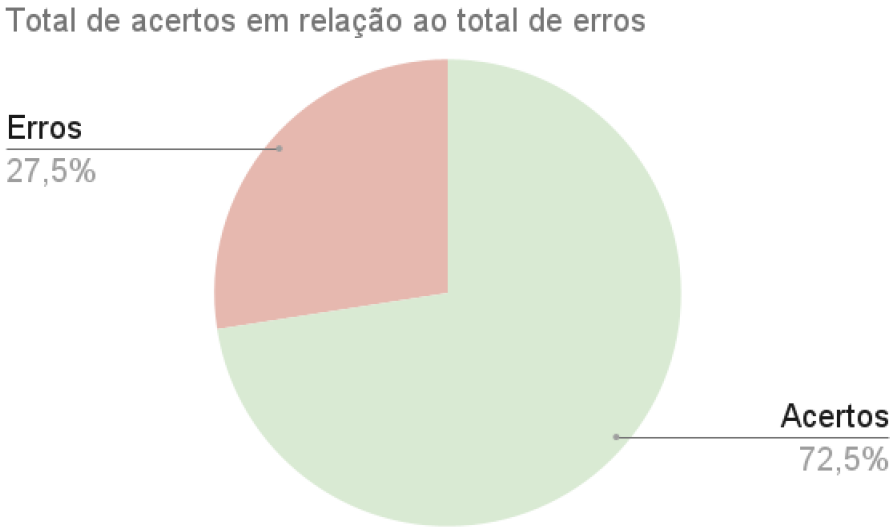

Figura 36 – Resultado: Total de acertos em relação ao total de erros

Fonte: Próprio autor

6) Top 3 manchetes mais acertadas

Título: Vacina da Pfizer raramente provoca evento adverso grave em crianças, diz relatório do CDC; (GLOBO, 2022c)
Primeiro Parágrafo: Órgão de saúde dos EUA analisou dados de mais de 42 mil crianças. Os eventos adversos mais comuns foram leves: dor no local da injeção, fadiga ou dor de cabeça.
Tipo da Manchete: FATO;
Quantidade de respostas: 23

Título: Escolas de Uganda reabrem após o fechamento mais longo do mundo devido à Covid; (GLOBO, 2022a)
Primeiro Parágrafo: Cerca de 15 milhões de crianças não frequentam as aulas desde março de 2020, quando as instituições de ensino fecharam devido à pandemia. Menos de 3% da população está vacinada.
Tipo da Manchete: FATO;
Quantidade de respostas: 23

Título: Pfizer registrou patente para rastrear pessoas vacinadas; (GLOBO, 2022e)
Primeiro Parágrafo: Uma mensagem que mostra um pedido de registro de patente de um sistema capaz de rastrear pessoas com doenças contagiosas por meio de dispositivos eletrônicos.
Tipo da Manchete: FAKE;
Quantidade de respostas: 21

7) Top 3 manchetes mais erradas

Título: Ômicron é mortal e não deve ser chamada de variante branda, diz OMS; (GLOBO, 2022f)
Primeiro Parágrafo: OMS alerta que variante ômicron está hospitalizando e matando pessoas.
Tipo da Manchete: FATO;
Quantidade de respostas: 14

Título: França autoriza que médicos com Covid-19 atendam pacientes; (GLOBO, 2022b)
Primeiro Parágrafo: Medida tenta conter a falta de mão de obra em um momento em que muitos profissionais estão afastados pela doença; apenas alguns trabalhadores, com poucos ou nenhum sintoma, poderão seguir em operação.
Tipo da Manchete: FATO;
Quantidade de respostas: 14

Título: Diretor-geral da OMS fez recomendação para o Brasil não festejar o carnaval em 2022; (GLOBO, 2022d)
Primeiro Parágrafo: Uma recomendação atribuída a Tedros Adhanom Ghebreyesus, diretor-geral da Organização Mundial da Saúde (OMS), para não haver Carnaval em 2022 no Brasil.
Tipo da Manchete: FAKE;
Quantidade de respostas: 12

4 Considerações finais

No transcorrer do desenvolvimento deste trabalho, foi idealizado e implementado um Sistema Web para fornecimento de uma pesquisa em formato de Quiz. Seu objetivo é recolher informações pessoais dos participantes e expor manchetes reais e verificadas pelo site do G1, com o intuito de coletar as respostas dos participantes em relação a essas manchetes classificando-as como fato ou fake para a realização de uma análise quantitativa.

Tal objetivo foi cumprido através do estudo e desenvolvimento do Sistema Web "Quiz Fato ou Fake?" que é uma aplicação composta por páginas responsáveis pela coleta de informações dos participantes como, idade, grau de escolaridade e fontes de pesquisa. Ainda, foram realizadas buscas e inserções no banco de dados para respectivamente, expor manchetes em formato de *Quiz* e armazenamento das informações do participante, para ao fim do processo conscientizá-lo com os links das manchetes respondidas.

Dessa forma, foi possível realizar um teste de sistema e uma posterior análise quantitativa dos resultados, com o cruzamento dos dados, e identificar o quanto determinado grupo de pessoas tende a ser ludibriado por fake news.

Tal feito somente foi possível com a junção de conhecimentos adquiridos no decorrer do curso de graduação em Ciência da Computação, relacionados às disciplinas como Engenharia de Software, Sistema de Banco de Dados, Sistema de Gerenciamento de Banco de Dados, Programação para Internet, Programação Orientada a Objetos 1 e 2.

4.1 Desafios encontrados

Durante o estudo e desenvolvimento do projeto surgiram algumas dificuldades já esperadas. A primeira era a forma de capturar as manchetes do site do G1 de forma automática, já que o site não disponibiliza uma API. Este problema foi resolvido aplicando os conhecimentos adquiridos em Programação para Internet, pois, foi possível inspecionar a página do G1 e verificar a API que fornece as informações ao mesmo. Como era uma resposta específica para G1, foi necessário realizar um filtro com as informações necessárias.

Outra dificuldade superada foi encontrar os recursos necessários para a integração entre o desenvolvimento backend, frontend e banco de dados, contudo, após pesquisas foi encontrado a plataforma de cloud computing Amazon Web Services(AWS). Esta plataforma fornece recursos esperados para esse projeto como, servidor para hospedar backend, banco de dados relacional, hospedagem para frontend e um gateway para integração entre

cliente-servidor. Foi uma oportunidade de explorar uma plataforma de cloud computing que está em alta nos dias atuais.

Também se fez necessário o estudo de frameworks e tecnologias utilizadas, e como estas facilitariam o desenvolvimento. Ter o conhecimento prévio de Programação Orientada, foi uma etapa fundamental para definições de padrões de projeto, mas ainda assim houve momentos de dificuldades como o entendimento da arquitetura hexagonal proposta para este projeto, já que foi decido utilizar uma arquitetura de baixo acoplamento e que está em alta na atualidade.

4.2 Trabalhos futuros

Como próximas etapas deste trabalho compreendem-se:

- Realização de análises qualitativas da amostra;
- Integrações com outros sites além do G1, como, Secretaria de Saúde do Distrito Federal, Instituo Butantan e outros jornais online;
- Abrangência de buscas em outras categorias além da saúde, como, economia, meio ambiente, política e educação;
- Desenvolvimento e treinamento de inteligência artificial para, conseguir categorizar manchetes automaticamente e buscar notícias na internet com base em uma frase para identificar se é fato ou fake; e
- Realizar integração através do WhatsApp para receber manchetes e analisar na internet a porcentagem da veracidade.

Referências

ALEXANDER, C.; ALEXANDER, P. *The Timeless Way of Building*. Oxford University Press, 1979. (Center for Environmental Structure Berkeley, Calif: Center for Environmental Structure series, v. 8). ISBN 9780195024029. Disponível em: <https://books.google.com.br/books?id=H6CE9hlbO8sC>. Acesso em: 24/01/2022. Citado na página 13.

ALEXANDER, C. et al. *A Pattern Language: Towns, Buildings, Construction*. OUP USA, 1977. (Center for Environmental Structure Berkeley, Calif: Center for Environmental Structure series). ISBN 9780195019193. Disponível em: <https://books.google.com.br/books?id=hwAHmktpk5IC>. Acesso em: 24/01/2022. Citado na página 13.

AMAZON. Aws documentation. *Amazon Web Services*, 2022. Disponível em: <https://docs.aws.amazon.com/index.html>. Acesso em: 08/01/2022. Citado 6 vezes nas páginas 20, 38, 39, 40, 42 e 43.

ARAUJO, C. Aws amplify + react: Criando e hospedando a aplicação. *DEV Community*, 2021. Disponível em: <https://dev.to/araujocristian/aws-amplify-react-criando-e-hospedando-a-aplicacao-4bo2>. Acesso em: 15/01/2022. Citado na página 20.

BARCELOS, T. d. N. e. a. de. Análise de fake news veiculadas durante a pandemia de covid-19 no brasil. *Revista Panamericana de Salud Pública [online]*, v. 45, n. 65, 2021. ISSN 16784561. Disponível em: <https://doi.org/10.26633/RPSP.2021.65>. Acesso em: 22/01/2022. Citado 2 vezes nas páginas 6 e 21.

BEPPLER, T. et al. Notícias falsas, dano real: Levantamento, análise e discussão de phishing, malware e fake news sobre covid-19. *Anais do Computer on the Beach*, v. 12, p. 080–087, 2021. Acesso em: 13/02/2022. Citado na página 21.

BERNERS-LEE, T. Uniform resource identifier (uri): Generic syntax. *T. Berners-Lee*, 2005. Disponível em: <https://www.rfc-editor.org/rfc/rfc3986.txt>. Acesso em: 07/02/2022. Citado na página 17.

BOS, B. Cascading style sheets. *WEC*, 1994. Disponível em: <https://www.w3.org/Style/CSS/#specs>. Acesso em: 13/02/2022. Citado na página 17.

BOTNEVIK, B.; SAKARIASSEN, E.; SETTY, V. Brenda: Browser extension for fake news detection. In: _____. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2020. p. 2117–2120. ISBN 9781450380164. Disponível em: <https://doi.org/10.1145/3397271.3401396>. Acesso em: 16/02/2022. Citado na página 21.

CAETANO, A. Sistema web para gerenciamento de eventos e emissão de documentos - sked. *UNIVERSIDADE FEDERAL DE UBERLÂNDIA*, 2019. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/28923/1/SistemaWebGerenciamento.pdf>. Acesso em: 17/01/2022. Citado na página 17.

COCKBURN, A. *Crystal clear: A human-powered methodology for small teams: A human-powered methodology for small teams*. [S.l.]: Pearson Education, 2004. Acesso em: 11/02/2022. Citado na página 11.

COCKBURN, A. *Hexagonal Architecture - The Pattern: Ports and Adapters*. 2005. Hexagonal architecture. Acessado em 14/02/2022. Disponível em: <https://alistair.cockburn.us/hexagonal-architecture/>. Acesso em: 14/02/2022. Citado na página 9.

COMMUNITY, G. Git guide. *GitHub, Inc*, 2021. Disponível em: <https://github.com/github/training-kit/blob/master/git-guides/git-overview.md>. Acesso em: 19/02/2022. Citado na página 28.

DBEAVER. *DBeaver Free Universal Database Tool*. 2022. DBeaver Community Edition. Acessado em 07/02/2022. Disponível em: <https://dbeaver.io/>. Acesso em: 15/01/2022. Citado na página 46.

EVANS, E.; FOWLER, M.; EVANS, E. *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2004. ISBN 9780321125217. Disponível em: <https://books.google.com.br/books?id=7dlaMs0SECsC>. Acesso em: 03/02/2022. Citado na página 14.

FIELDING, R. T. Architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California, Irvine*, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. Acesso em: 07/02/2022. Citado na página 16.

FLANAGAN, D. *JavaScript: O Guia Definitivo*. Bookman Editora, 2011. ISBN 9788565837484. Disponível em: <https://books.google.com.br/books?id=zWNyDgAAQBAJ>. Acesso em: 13/02/2022. Citado na página 19.

GALHARDI, C. P. e. a. Fato ou fake? uma análise da desinformação frente à pandemia da covid-19 no brasil. *Ciência & Saúde Coletiva [online]*, v. 25, 2021. ISSN 16805348. Disponível em: <https://doi.org/10.1590/1413-812320202510.2.28922020>. Acesso em: 22/01/2022. Citado na página 20.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994. (Addison-Wesley Professional Computing Series). ISBN 9780321700698. Disponível em: <https://books.google.com.br/books?id=6oHuKQe3TjQC>. Citado na página 13.

GARCIA-MOLINA, H.; ULLMAN, J.; WIDOM, J. *Database Systems: The Complete Book*. Pearson Education, 2011. ISBN 9780133002010. Disponível em: <https://books.google.com.br/books?id=gaEuAAAAQBAJ>. Acesso em: 23/01/2022. Citado na página 12.

GLOBO, G. *Escolas de Uganda reabrem após o fechamento mais longo do mundo devido à Covid*. 2022. Disponível em: <https://g1.globo.com/mundo/noticia/2022/01/10/escolas-de-uganda-reabrem-apos-o-fechamento-mais-longo-do-mundo-devido-a-covid.ghtml>. Acesso em: 19/01/2022. Citado na página 51.

GLOBO, G. *França autoriza que médicos com Covid-19 atendam pacientes*. 2022. Disponível em: <https://g1.globo.com/mundo/noticia/2022/01/05/franca-autoriza-que-medicos-com-covid-19-atendam-pacientes.ghtml>. Acesso em: 19/01/2022. Citado na página 52.

GLOBO, G. *Vacina da Pfizer raramente provoca evento adverso grave em crianças, diz relatório do CDC*. 2022. Disponível em: <https://g1.globo.com/saude/coronavirus/vacinas/noticia/2021/12/30/vacina-da-pfizer-raramente-provoca-evento-adverso-grave-em-criancas-diz-relatorio-do-cdc.ghtml>. Acesso em: 19/01/2022. Citado na página 51.

GLOBO, G. *É #FAKE que diretor-geral da OMS fez recomendação para o Brasil não festejar o carnaval em 2022*. 2022. Disponível em: <https://g1.globo.com/fato-ou-fake/coronavirus/noticia/2021/11/26/e-fake-que-diretor-geral-da-oms-fez-recomendacao-para-o-brasil-nao-festejar-o-carnaval-em-2022.ghtml>. Acesso em: 19/01/2022. Citado na página 52.

GLOBO, G. *É #FAKE que Pfizer registrou patente para rastrear pessoas vacinadas*. 2022. Disponível em: <https://g1.globo.com/fato-ou-fake/coronavirus/noticia/2021/11/05/e-fake-que-pfizer-registrou-patente-para-rastrear-pessoas-vacinadas.ghtml>. Acesso em: 19/01/2022. Citado na página 51.

GLOBO, G. *Ômicron é mortal e não deve ser chamada de variante branda, diz OMS*. 2022. Disponível em: <https://g1.globo.com/saude/coronavirus/noticia/2022/01/07/omicron-e-mortal-e-nao-deve-ser-chamada-de-variante-branda-diz-oms.ghtml>. Acesso em: 19/01/2022. Citado na página 52.

HAZZAN, O.; DUBINSKY, Y. *Agile Software Engineering*. Springer London, 2009. (Undergraduate Topics in Computer Science). ISBN 9781848001992. Disponível em: <https://books.google.com.br/books?id=kNFqQW3uZucC>. Acesso em: 17/01/2022. Citado na página 11.

HIGHSMITH, J. *History: The Agile Manifesto*. [s.n.], 2001. Disponível em: <https://agilemanifesto.org/history.html>. Acesso em: 17/01/2022. Citado na página 11.

IPSOS. 2018. Disponível em: <https://www.ipsos.com/sites/default/files/ct/news/documents/2018-08/fake_news-report.pdf>. Acesso em: 22/01/2022. Citado na página 6.

KALSNES, B. Fake news. In: *Oxford Research Encyclopedia of Communication*. [S.l.: s.n.], 2018. Citado na página 20.

MARTIN, R. C. The clean architecture. *Clean Code*, v. 45, n. 65, 2012. Disponível em: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>. Acesso em: 27/01/2022. Citado na página 14.

MARTINEZ, P. *Hexagonal Architecture, there are always two sides to every story*. 2021. Disponível em: <https://medium.com/ssense-tech/hexagonal-architecture-there-are-always-two-sides-to-every-story-bc0780ed7d9c>. Acesso em: 22/02/2022. Citado na página 16.

MASSE, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, 2011. ISBN 9781449319908. Disponível em: <https://books.google.com.br/books?id=eABpzyTcJNIC>. Acesso em: 03/02/2022. Citado na página 16.

MDN. Html: Linguagem de marcação de hipertexto. *MDN WEB DOCS*, 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 09/02/2022. Citado na página 17.

MDN. Usando a api web storage. *MDN WEB DOCS*, 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API>. Acesso em: 19/02/2022. Citado na página 34.

PAN-AMERICANA, O. d. S. *Histórico da pandemia de COVID-19*. 2020. Disponível em: <https://www.paho.org/pt/covid19/historico-da-pandemia-covid-19>. Acesso em: 23/02/2022. Citado na página 6.

PRESSMAN, R.; MAXIM, B. *Engenharia de Software - 8ª Edição*. [s.n.], 2016. ISBN 9788580555349. Disponível em: <https://books.google.com.br/books?id=wexzCwAAQBAJ>. Acesso em: 23/12/2022. Citado na página 11.

RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados - 3.ed.* McGraw Hill Brasil, 2008. ISBN 9788563308771. Disponível em: <https://books.google.com.br/books?id=COUJpkH5v38C>. Acesso em: 23/01/2022. Citado na página 12.

SCRUM, G. *Scrum Guide*. [s.n.], 2020. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-2.0.pdf>. Acesso em: 06/01/2022. Citado 2 vezes nas páginas 9 e 11.

SCRUM.ORG. *WHAT IS SCRUM?* 2020. Disponível em: <https://www.scrum.org/resources/what-is-scrum>. Acesso em: 11/01/2022. Citado na página 12.

SILVA, M. *React Aprenda Praticando: Desenvolva aplicações web reais com uso da biblioteca React e de seus módulos auxiliares*. Novatec Editora, 2021. ISBN 9786586057393. Disponível em: <https://books.google.com.br/books?id=MWkOEAAAQBAJ>. Acesso em: 13/02/2022. Citado 2 vezes nas páginas 9 e 19.

SOMMERVILLE, I. *Engenharia de software*. Pearson Prentice Hall, 2011. ISBN 9788579361081. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>. Acesso em: 26/01/2022. Citado na página 14.

TAURION, C. *Cloud Computing - Computação em Nuvem*. Brasport, 2009. ISBN 9788574524238. Disponível em: <https://books.google.com.br/books?id=mvir2X-A2mcC>. Acesso em: 14/02/2022. Citado na página 19.

TJPR, a. V. P. 2020. Disponível em: <https://www.tjpr.jus.br/noticias-2-vice/-/asset_publisher/sTrhoYRKnlQe/content/o-perigo-das-fake-news/14797>. Acesso em: 18/01/2022. Citado na página 6.

VALENTE, M. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. [s.n.], 2020. ISBN 9786500000771. Disponível em: <https://engsoftmoderna.info/>. Acesso em: 27/01/2022. Citado na página 14.