UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

AILSON ALEXANDRE DA SILVA MORAIS

EDUQUIZ: UMA ABORDAGEM DE CÓDIGO ABERTO PARA SISTEMAS DE RESPOSTA BASEADOS EM JOGOS

FORTALEZA

2023

AILSON ALEXANDRE DA SILVA MORAIS

EDUQUIZ: UMA ABORDAGEM DE CÓDIGO ABERTO PARA SISTEMAS DE RESPOSTA BASEADOS EM JOGOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Paulo Antônio Leal Rego

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M825e Morais, Ailson Alexandre da Silva.
Eduquiz : uma abordagem de código aberto para sistemas de resposta baseados em jogos / Ailson Alexandre da Silva Morais. – 2023.
65 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2023.
Orientação: Prof. Dr. Paulo Antônio Leal Rego.

1. Educação baseada em jogos. 2. Sistemas de respostas de estudantes. 3. Gamificação. 4. Aplicações em nuvem. I. Título.

CDD 621.39

AILSON ALEXANDRE DA SILVA MORAIS

EDUQUIZ: UMA ABORDAGEM DE CÓDIGO ABERTO PARA SISTEMAS DE RESPOSTA BASEADOS EM JOGOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Paulo Antônio Leal Rego (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal Do Ceará (UFC)

Prof. Dr. Fernando Antônio Mota Trinta
Universidade Federal Do Ceará (UFC)

À minha família, por sempre ter acreditado no meu potencial, até quando parecia não haver algum. Mamãe, foi você quem me deu a razão para estudar e sem você eu não teria nem começado. Papai, seu exemplo me deu a motivação e segurança para seguir em frente, para cima, para o lado, ou para qualquer outra direção que eu desejasse.

AGRADECIMENTOS

Inicialmente, gostaria de agradecer à minha família e amigos. Em especial ao meu pai Aildo e minha mãe Isabel, que mesmo sem nada, me entregaram tudo. À minha tia Luísa, que é uma segunda mãe para mim e me ajudou bastante sem nem ter percebido. À minha vó Dora e às minhas tias Joilda, Sandra e Lívia, que sempre me fizeram acreditar que eu poderia ser inteligente. Aos meus amigos fieis que, apesar de não serem muitos, me deram suporte em toda hora e em todo lugar, em diversas línguas.

Gostaria de agradecer ao Cristóbal, que foi minha âncora no momento mais difícil da minha vida e continua a ser meu amigo, mesmo depois de tudo, a 6 mil quilômetros de distância.

Um grande obrigado a todos que me ajudaram diretamente ou indiretamente neste projeto, que me deram conselhos, que me orientaram, ou que somente reclamaram junto comigo. Em especial, ao Samuel, que foi imprescindível para a realização deste trabalho e sua ajuda está além de todas as menções.

Aos meus professores durante toda a minha formação, o meu grande obrigado! Somente a didática e perseverança de vocês para me manter motivado e atento em sala de aula.

Ao meu orientador, que sabendo das minhas dificuldades e contratempos, não mediu esforços para me orientar.

Aos professores da banca examinadora, que leram de julgaram meu trabalho de forma justa e coerente.

A mim mesmo, que não desisti.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

Este trabalho de conclusão de curso explorou o desenvolvimento e implementação do “Eduquiz”, uma aplicação de educação de código aberto baseada em jogos. O “Eduquiz” foi projetado para ser uma alternativa gratuita e de código-aberto, com o objetivo de aumentar a participação e engajamento do aluno. O desenvolvimento do “Eduquiz” envolveu a criação de um front-end responsivo e um back-end em microsserviços, utilizando princípios conhecidos de *UX/UI* e técnicas de desenvolvimento de aplicações em nuvem. Além disso, a aplicação implementa funcionalidades diversas, incluindo a possibilidade de criar questionários, gerenciar salas, e fornecer *feedback* em tempo real para os participantes. A arquitetura do back-end do “Eduquiz” é fortemente apoiada pelos serviços do *Firebase*, empregando um banco de dados *NoSQL Firestore* e funções *serverless* para um desempenho escalável. A experiência do usuário e o design da interface foram de extrema importância na construção do “Eduquiz”, onde foi utilizada uma abordagem de design iterativa. Com o auxílio do Figma, o design da interface buscou a combinação de cores e tipografia que promovessem um ambiente de aprendizado amigável e envolvente. Através deste estudo, busca-se destacar o potencial das aplicações baseadas em jogos na educação e o valor da abertura e flexibilidade oferecidas pelo software de código aberto.

Palavras-chave: Educação baseada em jogos. Sistemas de respostas de estudantes. Gamificação. Aplicações em nuvem.

ABSTRACT

This undergraduate thesis explored the development and implementation of “Eduquiz”, an open-source, game-based education application. “Eduquiz” was designed to be a free and open-source alternative aimed at increasing student participation and engagement. The development of “Eduquiz” involved creating a responsive front-end and a microservices back-end, utilizing well-known UX/UI principles and cloud application development techniques. Moreover, the application implements a variety of features, including the ability to create quizzes, manage rooms, and provide real-time feedback to participants. The back-end architecture of “Eduquiz” is strongly supported by Firebase services, employing a NoSQL Firestore database and serverless functions for scalable performance. User experience and interface design were paramount in the construction of “Eduquiz”, where an iterative design approach was used. With the aid of Figma, the interface design sought a combination of colors and typography that would promote a friendly and engaging learning environment. Through this study, we aim to highlight the potential of game-based applications in education and the value of the openness and flexibility offered by open-source software.

Keywords: Game-based education. Student response systems. Gamification. Cloud applications.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
SDK	Software Development Kit
SRBJ	Sistemas de Resposta Baseados em Jogos
UI	*User Interface*
UX	*User Experience*

SUMÁRIO

1 INTRODUÇÃO

O estudo é uma atividade que demanda tempo e esforço, muitas vezes com resultados perceptíveis somente a longo prazo e, por consequência disso, se pode tornar rapidamente cansativo e maçante. Esse cenário se agrava ainda mais quando se leva em perspectiva o método aplicado em muitas instituições de ensino, com aulas monótonas e expositivas com pouca ou nenhuma participação ativa dos estudantes. De fato, como evidenciado em Carine George D. Kuh (2006) existe uma ligação forte entre engajamento e aprendizado estudantil. Praticamente proporcionais, alunos engajados tendem a absorver melhor o conteúdo e se concentrar mais nos tópicos abordados, levando a uma melhor compreensão do todo e fundamentos do estudo. O contrário também é válido, pouco engajamento pode ocasionar sentimentos de cansaço e desinteresse.

Toda essa relação entre engajamento e aprendizado torna-se ainda mais pertinente dado o contexto atual da recente pandemia da COVID-19, que mudou drasticamente a maneira de como o ensino era tradicionalmente visto e introduziu com urgência a necessidade de métodos alternativos. O ensino a distância e por computador realçou a importância da utilização da tecnologia voltada à facilitação do aprendizado remoto (BAO, 2020). Com a aplicação de ambientes virtuais e aulas muitas vezes assíncronas, o valor de estratégias pedagógicas que podem aumentar o engajamento e a motivação dos alunos se torna essencial. Nessa perspectiva, a gamificação, definida como a aplicação de elementos e dinâmicas de jogos em contextos não-lúdicos, surgiu como uma dessas estratégias eficazes.

A gamificação no ambiente educacional visa melhorar a experiência do aluno, promovendo a motivação, aumentando o engajamento e proporcionando um método de aprendizado mais interativo e estimulante (DETERDING, 2011). Essa abordagem se tornou a norma em muitas instituições durante a pandemia e se consolidou fortemente como uma alternativa para aumentar o engajamento em sala de aula e proporcionar uma boa experiência para os estudantes em relação ao ensino tradicional expositivo.

Na temática da gamifição em ambientes educacionais, existe o paradigma conhecido como SRBJ. Este é um método que combina o potencial dos jogos digitais com a flexibilidade e interatividade dos sistemas de resposta do aluno. Ao incorporar elementos de jogos, como competição, colaboração, recompensas e *feedback* imediato em tais sistemas de resposta, cria-se um ambiente de aprendizado altamente envolvente e motivador para os estudantes. Diversos estudos mostram que a implementação de SRBJ em salas de aula pode resultar em melhorias

significativas na atenção, motivação e resultados de aprendizagem (HUNG *et al.*, 2014). Assim, o SRBJ surge como um mecanismo promissor para potencializar os benefícios da gamificação na educação, particularmente no cenário atual de ensino e aprendizagem digital.

Neste ambiente digital emergente com novas relação entre aluno e ensino, a plataforma *Kahoot!* se destaca como um SRBJ bem-sucedido que utiliza conceitos de gamificação para transformar o aprendizado em uma atividade divertida e competitiva, incentivando a participação ativa dos alunos e facilitando a retenção de informações (WANG; TAHIR, 2020). De modo mais concreto, trata-se de um aplicativo baseado em jogos, tanto para *web* quanto para dispositivos móveis, usado para aprender e revisar conteúdos em um ambiente online em tempo real por professores e estudantes. O aplicativo cria um sessão de jogo virtual no qual o professor é o anfitrião e os estudantes devem responder anonimamente a diferentes tipos de perguntas de forma semelhante a um *quiz*. Baseado fortemente em princípios de gamificação e em um ambiente competitivo, os estudantes ganham pontos ao responder corretamente às perguntas em um curto espaço de tempo.

Contudo, mesmo com a ampla adoção de soluções como o *Kahoot!*, existe uma limitação importante que deve ser levada em conta: o custo. A exemplo do *Kahoot!*, para ter acesso a todas as funcionalidades da aplicação, como número ilimitado de alunos em cada sala de jogo, diferentes tipos de questionários e itens de customização, é necessário adesão a um plano mensal pago. Além disso, existem outros pontos relacionados como manutenção, novos conteúdos e flexibilidade de personalização dessa plataforma que dificultam o adesão plena por parte de uma instituição acadêmica.

A necessidade de superar essas barreiras ressalta a relevância de alternativas gratuitas e de código aberto na educação, que oferecem maior controle, flexibilidade e acessibilidade. Diante deste contexto, este trabalho apresenta o "Eduquiz", uma proposta de sistema web de resposta gratuito de código aberto totalmente baseado em gamificação. Desenvolvido com as tecnologias *Vue.js 3* e *Vuetify* para o front-end, e *Firebase* para o back-end, o "Eduquiz" propõe um ambiente de aprendizagem interativo e personalizável, sem fins lucrativos e de fácil configuração. Assim, a aplicação busca proporcionar uma interação agradável em sala de aula para que o engajamento do estudante possa se manter elevado.

1.1 Objetivos

Este projeto visa a implementação de um sistema *web* de resposta baseado em jogos que proporcione as funcionalidades principais de plataformas desse tipo de forma gratuita e de código aberto. Tais funcionalidades incluem: a possibilidade de múltiplos alunos conectados ao mesmo tempo, criação e gerenciamento de questionários e sistema de *feedback*. Além disso, como objetivos específicos, têm-se:

- Fazer um levantamento e comparar as diferentes soluções presentes no mercado baseadas em gamificação e sistemas de reposta baseados em jogos.
- Realizar uma pesquisa com utilizadores de tais sistemas.
- Delimitar os requisitos funcionais e não-funcionais.
- Propor uma arquitetura de Front-end pautando-se em conceitos de *UI/UX*.
- Propor uma arquitetura de Back-end em microsserviços.

Além disso, todo o código-fonte da aplicação foi disponibilizado gratuitamente na plataforma digital *github*.

Este TCC tem como público-alvo a academia em geral e a desenvolvedores interessados no conceito de gamificação e que desejam aprofundar-se em projetos de código aberto.

1.2 Organização e capítulos

Este trabalho divide-se nas seguintes partes:

1. **Introdução:** breve capítulo introdutório e uma básica contextualização da problemática e dos conceitos de gamificação.
2. **Contextualização teórica:** fundamentação teórica das técnicas de gamificação e das tecnologias empregadas no projeto.
3. **Metodologia:** aprofundamento da arquitetura geral do projeto com o detalhamento do design com UI/UX, lógicas do front-end e back-end, e também do banco de dados utilizado.
4. **Resultados:** exposição do estado final da aplicação, juntamente com seus aspectos positivos e negativos.
5. **Conclusão:** uma visão geral do projeto abrindo também para perspectivas de incrementos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Dada a característica densa do trabalho, é importante atentar-se a alguns conceitos chaves que serão empregados durante todo o texto. As discussões e conclusões acerca do "Eduquiz" somente poderão ser facilmente realizadas mediante o entendimento dos conceitos apresentados neste capítulo. Desse modo, são apresentadas quatro secções distintas: gamificação, sistemas de resposta baseados em jogos, front-end e back-end.

2.1 Gamificação

O conceito de gamificação resume-se ao processo pelo qual elementos e características de jogos são aplicados em contextos pouco convencionais, por exemplo, em ambientes de trabalho, educação, negócios, saúde, entre outros (DETERDING, 2011). O objetivo principal desta técnica é incentivar e aumentar o engajamento dos participantes na aquisição de conhecimentos e habilidades, através da introdução de elementos lúdicos que podem tornar tarefas rotineiras ou complexas mais agradáveis e motivadoras (KAPP, 2012).

De forma sucinta, com o termo cunhado por *Nick Pelling*, a gamificação baseia-se na mecânica de jogos como pontos, missões, recompensas, sistema de nível e competições para mudar a forma de que determinada atividade ou tarefa é realizada (ZICHERMANN; CUNNINGHAM, 2011). No entanto, essa técnica vai além: o design e a experiência do usuário é igualmente importante para manter altos níveis de engajamento. A criação de sistemas que sejam por si só prazerosos e gratificantes de jogar é uma boa forma de aumentar o engajamento e a produtividade de maneira divertida e interativa (DENNY, 2013).

Especialmente em contextos educacionais, a gamificação vem mostrando resultados interessantes no aumento de motivação e engajamento dos alunos, com a promoção de uma aprendizagem menos passiva e melhores rendimentos acadêmicos (HANUS; FOX, 2015).

Na perspectiva do "Eduquiz", a gamificação é o elemento mais importante da premissa, que busca permitir uma maneira fluída e intuitiva de aprendizado.

No entanto, é importante considerar que, apesar dos benefícios potenciais, a gamificação também tem suas críticas e desafios. Alguns estudos sugerem que a gamificação pode, paradoxalmente, diminuir a motivação intrínseca dos alunos ao longo do tempo, pois a motivação inicial gerada por recompensas e prêmios pode dar lugar a uma dependência desses incentivos (NICHOLSON, 2015). Isto é, ao concentrar-se demasiadamente na competição e nas

recompensas, a gamificação pode transformar o processo de aprendizado em uma corrida por pontos, diminuindo o foco na compreensão profunda do conteúdo (MOLONEY, 2011).

Além disso, a gamificação pode contribuir para a criação de um ambiente de aprendizado desigual, em que alunos mais competitivos ou mais confortáveis com a mecânica de jogos se sobressaem, enquanto outros podem se sentir desencorajados ou marginalizados (ROBERTS, 2015). A gamificação também pode ser vista como uma forma de manipulação, uma vez que utiliza técnicas de engajamento desenvolvidas na indústria de jogos para motivar comportamentos desejados, o que pode levar a questionamentos éticos sobre o controle e a autonomia dos alunos (BOGOST, 2011).

Por fim, a implementação efetiva da gamificação em contextos educacionais requer recursos significativos e habilidades específicas de design de jogos, o que pode representar um obstáculo para instituições e professores (HAMARI *et al.*, 2014). Isso reforça a necessidade de uma abordagem cuidadosa e consciente na implementação de estratégias de gamificação, considerando tanto seus benefícios quanto seus desafios potenciais.

2.2 Sistemas de resposta baseados em jogos

Dentro dos conceitos de gamificação, os sistemas de resposta baseados em jogos utilizam elementos de jogos para tornar a educação mais interativa e atraente. Eles representam uma mudança na pedagogia tradicional, onde os estudantes são considerados participantes ativos em vez de receptores passivos de informação (WANG; TAHIR, 2020).

Um SRBJ é essencialmente uma plataforma digital que permite aos estudantes responder a perguntas propostas pelos professores em tempo real durante a aula. Estes sistemas são geralmente baseados em tecnologias *web* ou *mobile* e oferecem uma variedade de formatos de perguntas, incluindo escolha múltipla, verdadeiro ou falso, resposta curta, entre outros (GEE, 2003).

Ao incorporar elementos de jogos, como pontos, tabelas de classificação, e feedback instantâneo, os SRBJ conseguem engajar os estudantes de uma maneira que os métodos de ensino tradicionais muitas vezes não conseguem. Segundo Wang e Tahir (2020), estes elementos gamificados podem encorajar a participação, aumentar a motivação e promover uma competição saudável entre os estudantes.

Há uma crescente evidência empírica que sugere que o uso de SRBJ na sala de aula pode ter impactos positivos sobre o envolvimento dos estudantes e os resultados de aprendizagem.

Eles têm se mostrado particularmente eficazes em aumentar a atenção dos estudantes, melhorando a compreensão do conteúdo, e estimulando a interação entre os estudantes e os professores (OWEN; LICORISH, 2020).

No entanto, também é importante ressaltar que o uso efetivo de SRBJ requer um planejamento cuidadoso por parte dos professores. A inclusão de perguntas bem formuladas, a consideração do ritmo da aula, e a integração efetiva do sistema no currículo são fatores cruciais para maximizar os benefícios desses sistemas (RANIERI *et al.*, 2021).

Sistemas que utilizam-se dessa técnica, como o *Kahoot!*, vêm crescendo atualmente. Questionários em tempo real são uma simplificação fácil do sistema de respostas baseado em jogos e têm resultados que comprovam um bom melhoramento do envolvimento dos alunos e o desempenho acadêmico (WANG; TAHIR, 2020).

2.3 *Front-end*

O conceito de *front-end*, no contexto de uma aplicação *web*, refere-se à camada de interface com o usuário, a qual é renderizada no navegador *web* do cliente. Ele engloba a parte da aplicação com a qual o usuário interage diretamente, incluindo os aspectos visuais e de interação. O desenvolvimento *front-end* envolve a criação de elementos interativos, como botões, menus, formulários, e a estilização desses elementos para proporcionar uma experiência de usuário eficaz e agradável (LAWSON; SHARP, 2011).

Do *front-end*, dois conceitos importantes precisam ser extraídos. O primeiro refere-se ao *UX/UI design*, de fato como o usuário irá interagir e ver a aplicação e o segundo diz respeito a como a aplicação foi construída.

2.3.1 Conceitos de UX/UI design

O design de *User Interface* (UI) e *User Experience* (UX) são componentes fundamentais em produtos digitais. O conceito de UI refere-se somente à estética do produto e o seu respectivo layout. UX, por sua vez, é um termo mais abrangente e refere-se a toda a experiência que o usuário tem ao interagir na aplicação. (TULLIS; ALBERT, 2008)

O conceito de UI concentra-se majoritariamente na aparência visual da aplicação e como o usuário utiliza os interagíveis da aplicação, isto é: botões, sliders, ícones, tabelas, etc. Esses elementos são responsáveis por transmitir a estética do produto, como cores e tipografia; e

também facilitar na usabilidade.

No que diz respeito ao UX, o foco é na criação da experiência completa. Por exemplo, na consideração de fatores como facilidade de uso, valor agregado e na certificação de que todas as jornadas de usuário estão sendo satisfeitas.

Segundo Norman (2002), os conceitos básicos de design UI/UX podem ser sumarizados em três pontos: a usabilidade, *affordance* e *feedback*. O primeiro diz respeito à facilidade de uso da aplicação. O segundo, que traduzido ao português significa "possibilidade de ação", remete à transparência que um objeto tem para o usuário, i.e, o quão fácil é perceber as diversas possibilidades de ação que determinado componente tem. Por sua vez, feedback refere-se às respostas que a aplicação da face às ações do usuário.

2.3.2 Frameworks de desenvolvimento front-end: Vue.js 3 e Vuetify 3

Os *frameworks* de desenvolvimento têm uma função essencial na construção de aplicações robustas, escaláveis e eficientes. Eles não somente tornam o processo de desenvolvimento algo prazeroso e fácil de estruturar, mas também fornecem um conjunto de regras e padrões já estabelecidos que facilitam a colaboração dentro de uma equipe e a manutenção do código. A grande maioria dos *frameworks* contam com pacotes de códigos já escritos que aceleram o planejamento e prototipagem, dando à possibilidade de concentrar-se somente nas características únicas de cada aplicação (YOU, 2020).

Nesse âmbito, dentro da miríade incontável de *frameworks* dentro do ecossistema *JavaScript* - uma das principais linguagens de desenvolvimento para web - destaca-se o *Vue.js*. Sem estar sendo desenvolvidas por grandes empresas como outras tecnologias, o *Vue.js*, agora em sua terceira versão, emerge como uma solução de *framework* inteiramente de código-aberto. Originalmente concebido por Evan You, um ex-engenheiro do Google, a ferramenta apresenta grande flexibilidade e simplicidade no desenvolvimento de UI para usuários finais, permitindo que os desenvolvedores criem desde aplicações web de página única (SPA) até complexas aplicações de larga escala (YOU, 2020). O grande diferencial do *Vue.js* é a utilização de componentes reutilizáveis de HTML/CSS, que possibilita uma abordagem mais metódica na concepção de aplicativos *web* e uma estrutura modular que torna mais simples a organização e manutenção do código.

Em termos de design de *UI*, o *vuetify* caracteriza-se como um *framework* de componentes para *Vue.js*, seguindo os princípios do *Material Design* do Google. Em termos simples,

o *Vuetify* oferece uma série de componentes já pré-programados que oferecem responsividade, design e funcionalidade sem a necessidade de configuração por parte dos desenvolvedores. Essa ferramenta simplifica o processo de desenvolvimento de interfaces, tornando a experiência do usuário mais agradável e consistente sem o esforço de desenvolvimento e investimento em códigos CSS.

Baseado nessas características, o *front-end* do "Eduquiz" foi inteiramente feito utilizando *Vue.js 3* e *Vuetify 3*, com pequenas modificações para oferecer uma melhor estética.

2.4 *Back-end*

O back-end de uma aplicação web é definido como a camada de servidor, que compreende os aspectos do software que não estão visíveis diretamente para o usuário final, mas são responsáveis por processar e gerenciar os dados necessários para o funcionamento da aplicação (SHULL *et al.*, 2002). É aqui que reside a lógica de negócios, operações de banco de dados, autenticação de usuários e integrações com outros sistemas, como servidores de e-mail e sistemas de pagamento. Além disso, o back-end é também responsável por receber as solicitações do front-end, processá-las e enviar as respostas apropriadas de volta (FIELDING, 2000).

No que se refere ao "Eduquiz", toda sua arquitetura tem como principais elementos do *back-end*: arquitetura de microsserviços, *WebSockets* e *Firebase* com banco de dados *NoSql*.

2.4.1 Arquitetura de Microsserviços

A arquitetura de microsserviços é basicamente uma forma de planejar e desenvolver um projeto no qual o software é visto como um conjunto de pequenos serviços independentes que interagem entre si (NEWMAN, 2015). Na maioria das vezes, esses serviços estão hospedados em diferentes servidores, em diferentes regiões, e trocam informações por *Application Programming Interface* (API)s. Cada microsserviço encapsula uma lógica própria e isolada, cada um com sua infraestrutura de base própria, i.e, com banco de dados próprios e também são desenvolvidos, implantados e escalados de forma independente (DRAGONI *et al.*, 2017).

A ideia contrária de uma arquitetura de microsserviços é dita uma arquitetura monolítica, na qual todas as funcionalidades do sistema são interdependentes e organizadas em um único processo, onde cada componente do software, desde a interface do usuário até o acesso ao banco de dados, é parte de uma única aplicação unificada (NEWMAN, 2015). Embora essa

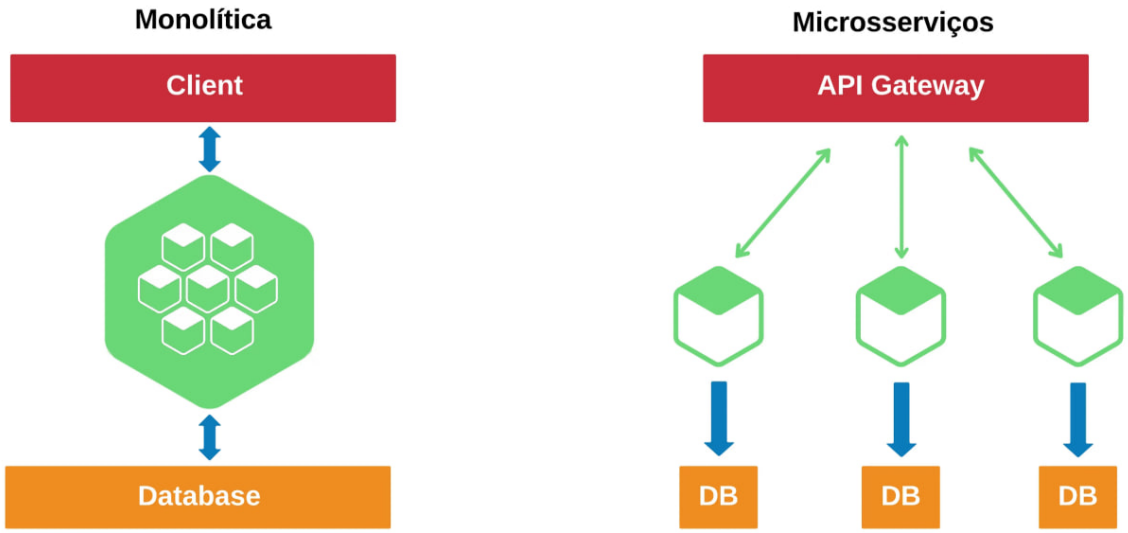
abordagem possa ser mais simples de desenvolver e testar inicialmente, especialmente para projetos de pequeno a médio porte, ela tem suas desvantagens. Dentre elas, destacam-se a dificuldade de escalar de maneira granular, de acordo com as necessidades de diferentes partes do sistema, a dependência entre as equipes de desenvolvimento, que precisam coordenar e entender toda a base de código, e o potencial impacto no sistema como um todo caso ocorra uma falha em algum componente.

Por outro lado, a arquitetura de microsserviços proporciona maior modularidade ao dividir uma aplicação em uma coleção de serviços menores que são desenvolvidos, implantados e escalados de forma independente. Cada microsserviço é responsável por um conjunto de funcionalidades relacionadas e tem a capacidade de funcionar de forma autônoma, permitindo a escalabilidade e a resiliência do sistema como um todo. Além disso, cada serviço pode ser escrito em diferentes linguagens de programação e utilizar diferentes tecnologias de armazenamento de dados, adaptando-se melhor às necessidades específicas de cada componente.

Essa arquitetura se mostra particularmente benéfica ao ser utilizada em combinação com tecnologias de computação em nuvem, como Amazon Web Services (AWS), Google Firebase e Microsoft Azure, que fornecem a infraestrutura necessária para gerenciar e escalar de maneira eficiente esses microsserviços (BLINOWSKI *et al.*, 2022).

A figura 1 abaixo ilustra de forma básica a diferença entre arquiteturas monolíticas e de microsserviços.

Figura 1 – Diferença entre arquiteturas monolíticas e de microsserviços.



Arquitetura de Microsserviços

Fonte: Gonçalves (2020).

Como comentado, A estrutura de microsserviços no “Eduquiz” permite uma separação dos componentes do sistema, como a base de dados em tempo real, o banco de dados *NoSql*, e *Cloud Storage*. Tais componentes podem escalar independentemente e suprir a demanda.

2.4.2 WebSockets

Quando se pensa em aplicações de tempo real como o *Kahoot!*, os *WebSockets* geralmente veem associados. Essa tecnologia fornece uma forma de conexão bidirecional contínua e eficiente entre o cliente e o servidor de uma aplicação, em oposição ao padrão HTTP em que o cliente inicia a conexão e aguarda uma reposta do servidor (FETTE; MELNIKOV, 2011).

A vantagem principal do *WebSocket*, protocolo estabelecido em 2011, é a possibilidade de transferência contínua de informações entre o cliente e o servidor sem a necessidade de múltiplas requisições HTTP, o que evita sobrecarregar a rede e reduz a latência da comunicação. Em um relação cliente-servidor comum no protocolo HTTP, geralmente o cliente realiza uma espécie de *polling*, no qual fica em espera esperando um determinado dado do servidor. Essa estratégia causa uma sobrecarga, uma vez que diversas conexões HTTP são criadas, com seus respectivas mensagens de espera e cabeçalhos trocados.

Baseado em Fette e Melnikov (2011), o protocolo *WebSocket* pode ser sumarizado da seguinte forma:

- Com um *handshake* inicial, semelhante ao HTTP, uma conexão TCP entre cliente e servidor é instaurada.
- A conexão é atualizada para um canal de comunicação persistente bidirecional, uma vez estabelecida.
- Toda a troca de mensagens entre cliente e servidor é feita sobre o mesmo canal aberto inicialmente.
- A conexão é encerrada até que a ligação seja finalizada por uma das partes ou por *timeout*.

Esse protocolo é bastante utilizado tipos de aplicações nas quais a comunicação em tempo real se faz necessária. Exemplos de uso vão desde sistema de chat em tempo real, até jogos *multiplayer*, atualizações de conteúdo ao vivo e sistemas de dados críticos em tempo real.

Na perspectiva do “Eduquiz”, o protocolo de *WebSocket* é uma das bases do desenvolvimento. A comunicação em tempo real entre cliente e servidor garante uma experiência de jogo fluida com *feedbacks* instantâneo e dinamismo nas partidas. Além disso, todo o sistema

de alocação de jogadores e hospedagem de salas de jogo são fortemente construídas sobre o protocolo. Definitivamente, esse nível de interação em tempo real pode ajudar a aumentar o engajamento e fomentar um melhor aprendizado dos estudantes. Não obstante, o protocolo permite uma comunicação eficiente entre múltiplos clientes em tempo real com o servidor, garantindo que as informações cheguem de forma uniforme para todos os jogadores. O *WebSocket* permite que uma grande quantidade de jogadores participe das partidas com o mínimo de latência possível.

2.4.3 Firebase

Firebase é uma das principais plataformas de computação em nuvem disponível atualmente. Desenvolvido pelo Google, a ferramenta dispõe de uma variedade de serviços que a maioria das aplicações precisam: sistemas de autenticação e autorização, armazenamento em nuvem, banco de dados relacionais e não-relacionais, serviços de tempo real, e formas de rodar código *serverless*. Essas são apenas algumas características que tornam o *Firebase* uma ferramenta viável de prototipagem e de desenvolvimento rápido de aplicações web.

A principal vantagem do *Firebase* é a possibilidade de desenvolver aplicativos seguindo os padrões de desenvolvimento de forma ágil e sem riscos, onde toda a infraestrutura é gerenciada pelo Google e os custos são limitados somente às utilizações dos serviços. Conforme a aplicação cresce em escopo e em complexidade, serviços podem ser removidos ou novos podem ser adicionados, sem necessidade de re-estruturar o código-fonte da aplicação. Outro ponto positivo é que toda a parte de escalabilidade e segurança da aplicação é de responsabilidade do Google, o que torna a aplicação resiliente e menos suscetiva a falhas.

Dentre as soluções propostas, chama a atenção em especial as seguintes, que foram de extrema importância para o desenvolvimento de "Eduquiz" :

- ***Firebase Authentication:*** é um serviço que permite a autenticação e o gerenciamento de usuários cadastrados no aplicativo de forma transparente e livre de escrita de código pelo desenvolvedor da aplicação. O serviço suporta diferentes tipos de autenticação: autenticação anônima, autenticação por e-mail e senha, autenticação por telefone e também por *Oauth2* sem a necessidade configurar um servidor de autenticação proprietário.
- ***Firestore:*** é o principal serviço de bancos de dados disponível no *Firebase*. Trata-se de um banco de dados não-relacional do tipo *NoSql* baseado em nuvem que oferece armazenamento em escala global e modularizado. Esse sistema é ideal para sistemas

distribuídos e com arquitetura de microsserviços que necessitam de sincronização. Base de dados *NoSql* garante uma grande velocidade de acesso e escrita de forma segura para sistemas grande e complexos.

- ***Realtime Database:*** é uma outra base de dados não-relacional adaptada para o acesso em tempo real. Para sistemas que sustentam-se sobre o protocolo de *WebSocket*, esse tipo de serviço é indispensável.
- ***Cloud Functions:*** é onde toda a lógica do back-end pode ser implementada de maneira *serverless*. Baseado em eventos, as funções podem ser acionadas por meio de interações com o banco de dados, requisições HTTP, acessos ao sistema de autentificação ou programadas para execução periódica.

3 METODOLOGIA

Este capítulo aborda de fato como o “Eduquiz” foi concebido, tendo como base a contextualização teórica descrita anteriormente. Em particular, será discutido os requisitos funcionais e não funcionais do sistema, que servem como as especificações básicas para o desenvolvimento da aplicação. Os requisitos funcionais referem-se à funcionalidade que a aplicação deve oferecer e como ela deve reagir a entradas específicas, enquanto os requisitos não funcionais são as propriedades e restrições sob as quais a aplicação deve operar.

Além disso, será explorado o processo de design realizado utilizando o software Figma, uma ferramenta digital de design de interface de usuário (UI) e experiência do usuário (UX) que permite conceituar e visualizar a aplicação antes de sua implementação efetiva.

Finalmente, também será discutido a arquitetura da aplicação, examinando como o sistema foi estruturado e como suas diferentes partes se comunicam e interagem entre si para fornecer a funcionalidade desejada. A figura 2 abaixo ilustra todas as etapas de desenvolvimento que serão discutidas nesta secção.

Figura 2 – Etapas de desenvolvimento da aplicação.

Fonte: Figura do autor (2023).

3.1 Levantamento de Requisitos

A identificação precisa dos requisitos funcionais e não funcionais é um componente crítico para a construção bem-sucedida de um software. Esta etapa predetermina o conjunto central de funcionalidades que a aplicação deve fornecer para estabelecer sua singularidade em meio à concorrência. Além disso, a relevância de qualquer aplicação é medida pela sua capacidade de resolver problemas e ser adotada por seus usuários. Levando isso em consideração, os requisitos do “Eduquiz” foram determinados seguindo duas etapas principais:

1. Análise das plataformas de questionários existentes que utilizam o paradigma de aprendi-

zado baseado em jogos.

2. Investigação das funcionalidades centrais do ponto de vista do usuário, realizada através de uma pesquisa de campo.

Combinando uma análise abrangente da concorrência com uma pesquisa de campo focada no usuário, foi possível definir uma lista robusta e direcionada de requisitos funcionais e não funcionais para o “Eduquiz”.

3.1.1 Plataformas SRBJ existentes

Existe um número considerável de projetos que tentam implementar soluções usando conceitos de gamificação. Para reduzir o escopo da pesquisa, somente os mais relevantes foram utilizados, baseando-se no número de usuários ativos e funcionalidades apresentadas. São eles:

1. **Kahoot!** - Líder de mercado e pioneiro na popularização dos SRBJ. Apresenta um grande esforço em *design* e grande possibilidade de personalização de questionários. No entanto, é uma plataforma paga.
2. **Squizzy** - Utiliza um back-end privado (sanity.io), o que limita a adaptabilidade e a personalização por parte dos desenvolvedores.
3. **Quizizz** - Embora siga a mesma ideia do Kahoot! e seja gratuito, o seu código não é de código aberto, restringindo a liberdade dos usuários para modificá-lo ou personalizá-lo para suas necessidades específicas.
4. **Classquiz** - Apesar de ser de código aberto, falta rigor no seu design e implementação. Trata-se de uma aplicação monolítica hospedada em um servidor físico na Alemanha sem a implementação de serviços em nuvem. Isso significa que não está adaptado para escalabilidade, e seu uso pode ser problemático. A interface do usuário não é moderna e fluida e não segue padrões de UI/UX reconhecidos, o que pode dificultar a sua adoção e utilização pelos usuários.

De forma resumida, segue abaixo a tabela 1, referente ao apurado das diferenças e semelhanças dos softwares apresentados:

Tabela 1 – Comparações entre plataformas existentes.

Característica	Kahoot!	Squizzy	Quizizz	Classquiz
Gratuito	Não[a]	Sim	Sim	Sim
De código-aberto	Não	Somente *front-end*	Não	Sim
Apoio da comunidade	Não	Não	Não	Não
Funcionaliades extras[b]	Sim	Sim	Não	Não
Conceitos de UI/UX[c]	Sim	Sim	Não	Não

[a] O *Kahoot!* possui um modo gratuito até 10 jogadores, com funcionalidades bastante limitadas.

[b] A possibilidade de criar aulas por meio de slides, salvar conteúdo e criar roteiros, fazer upload de vídeos e exibir conteúdo em forma de blog. Além de ser pago, o Kahoot! oferece uma grande variedade de opções para engajar os usuários e ajudar os estudantes no processo de aprendizagem.

[c] Um dos recursos mais importantes que esses tipos de aplicativos precisam ter é um sistema de UI/UX bem construído para transformar as sessões em uma partida semelhante a um jogo. Com componentes de UI/UX fluidos e intuitivos, o Kahoot! é a aplicação mais ambiciosa e bem executada em comparação com as outras, o que, em termos, justifica a taxa elevada requerida.

Fonte: elaborado pelo autor (2023).

Os dados apresentados demonstram a necessidade de um equilíbrio entre acessibilidade, adaptabilidade, escalabilidade e aderência a padrões modernos de design de UI/UX, um equilíbrio que o "Eduquiz" procura alcançar.

3.1.2 Entrevista de levantamento de requisitos

A pesquisa de levantamento de requisitos foi dividida nas seguintes partes:

- Formulário online respondido de forma assíncrona por professores e estudantes universitários. Todas as questões tratavam-se de perguntas objetivas elencadas de 0 a 5, baseando-se na escala *Likert*, conforme a relevância da funcionalidade apresentada.
- Entrevista rápida de 5 minutos com professores e estudantes com perguntas abertas para recolher opiniões precisas dos potenciais usuários.

Ao todo, houve um total de 5 participantes, dentre eles professores e alunos, que já trabalharam ou utilizaram ferramentas semelhantes no passado. Para um expositivo completo da entrevista usada, consultar a secção de apêndices.

3.1.3 Requisitos funcionais e não-funcionais

Da análise das plataformas já existentes e da entrevista de levantamento de requisitos, as principais funcionalidades da aplicação foram delimitadas. Essas funcionalidades serviram como ponto de partida para o desenvolvimento da aplicação e como referencial para as versões

estáveis de lançamento.

3.1.3.1 Requisitos funcionais

Os requisitos funcionais foram classificados em duas categorias distintas segundo uma análise de complexidade de implementação: complexidade elevada e complexidade normal. Esta análise foi realizada baseando-se em critérios de percepção de dificuldade e incerteza, estabelecidos pelo próprio desenvolvedor do trabalho.

Requisitos categorizados como de Complexidade Elevada são aqueles identificados pelo desenvolvedor como mais desafiantes para implementar, quer devido à maior incerteza em torno de como proceder, quer devido ao grau de esforço de engenharia exigido. Em contraste, os requisitos classificados como de Complexidade Normal são aqueles para os quais o desenvolvedor já possui uma clareza maior sobre as etapas de implementação, tendo, portanto, uma complexidade percebida como menor.

- Complexidade Elevada:
  - Participação no questionário: Os usuários devem poder participar dos questionários criados por outros.
  - Feedback em tempo real: O aplicativo deve fornecer feedback em tempo real para os participantes sobre suas respostas ao questionário com um sistema de pontuação.
  - Suporte a vários jogadores: O aplicativo deve permitir que vários participantes joguem o mesmo questionário ao mesmo tempo com uma conexão de rede celular. Mínimo de 30 jogadores.
  - Os estudantes e professores precisam ser capazes de criar cartões de memorização para fixação de conteúdo. Esses cartões podem ser anexados a pastas e seções para serem melhor organizados.
- Complexidade Normal:
  - Registro e login do usuário: para professores e alunos.
  - Criação de questionários: Usuários cadastrados devem poder criar questionários com perguntas, opções de resposta e respostas corretas.
  - Gerenciamento de questionários: Os usuários cadastrados devem poder gerenciar suas salas, incluindo criar, editar, excluir e compartilhá-las com outras pessoas com ou sem uma conta.
  - Hospedagem de questionários: Os usuários cadastrados devem poder hospedar seus

questionários para que outros possam participar, com ou sem uma conta.

– Tabela de classificação: O aplicativo deve exibir uma tabela de classificação para mostrar como os participantes estão se saindo no questionário em comparação com os outros. O tipo de classificação deve ser definido pelo proprietário do questionário.

3.1.3.2 Requisitos não-funcionais

- As páginas do front-end devem ser reativas e responsivas, i.e, as páginas não somente devem refletir mudanças de estado e ações performadas por usuários, como também se adaptar a diferentes tamanhos de tela.
- O tempo de resposta deve ser suficiente para permitir uma sessão de jogo de modo fluído. Tempo de resposta máximo de *300ms* em todas as ações performadas.
- Os dados e o estado da sala devem ser salvos se ocorrer uma interrupção de conexão.
- As páginas do front-end devem ser roteadas com base na funcionalidade e com poucos elementos seguindo as recomendações do *Material Design*.
- O front-end e o back-end devem ter sua infraestrutura dinamicamente escalável com balanceadores de carga.

Desta forma, a lista apresentada resume os principais requisitos funcionais e não-funcionais identificados para a construção do "Eduquiz". No entanto, é importante salientar que, em projetos de software, a complexidade pode mudar à medida que se ganha uma compreensão mais profunda do problema ou quando os requisitos evoluem com base nas necessidades dos usuários.

3.2 Definição do Front-end e Experiência de usuário da aplicação

A decisão de utilizar o framework *Vue.js* e a biblioteca *Vuetify 3* na arquitetura do front-end do "Eduquiz" foi impulsionada por uma combinação de fatores, sobretudo, a familiaridade prévia com estas ferramentas. O entendimento dos nuances destas tecnologias, decorrente de experiências anteriores, facilitou a capacidade de explorar as funcionalidades de modo eficaz, otimizando assim o processo de implementação.

Vue.js, por natureza, é conhecido pela sua estrutura modular e capacidade de manter uma base de código de fácil manutenção. Paralelamente, o *Vuetify 3* oferece uma extensa biblioteca de componentes pré-desenhados seguindo as diretrizes do *Material Design*, permitindo

uma consistência visual na aplicação. Essa combinação de estética e funcionalidade busca uma experiência de usuário refinada e intuitiva, tornando o “Eduquiz“ mais acessível.

Ademais, a abordagem proporcionada pela “Options API“ no “Vue.js“ simplifica o processo de declaração reativa de dados e composição de lógica de componentes, facilitando a construção de modo reutilizável. Esta estrutura é especialmente útil para desenvolvedores familiarizados com o *Vue.js*. Na *Options API*, a lógica do componente é dividida em seções predefinidas, como *data*, *methods*, *computed*, etc. Isso fornece uma estrutura organizada e fácil de entender, especialmente para projetos menores ou com um time menor de desenvolvimento.

O código-fonte abaixo ilustra a utilização da *Options API* para o gerenciamento de dados e métodos de um componente. O componente em questão é o *App.vue*, ponto de partida da aplicação “Eduquiz”.

Código-fonte 1 – Options API em Vue.js

```
<template>
  <v-app>
    <div class="mx-auto w-100 w-md-75" style="max-width:
       1100px">
      <MainHeader :isLoggedIn="isLoggedIn"></MainHeader>
    </div>
    <router-view></router-view>
    <MainFooter></MainFooter>
  </v-app>
</template>

<script>
import MainHeader from "./components/UI/MainHeader.vue";
import MainFooter from "./components/UI/MainFooter.vue";
import { getAuth, onAuthStateChanged, signOut } from "
   firebase/auth";
export default {
  components: {
    MainHeader,
```

```
18      MainFooter,
19    },
20    data() {
21      return { isLoggedIn: true };
22    },
23    created() {
24      let auth = getAuth();
25      onAuthStateChanged(auth, (user) => {
26        if (user) {
27          this.$store.state.userId = user.uid;
28          this.isLoggedIn = true;
29        } else {
30          this.$store.state.userId = null;
31          this.isLoggedIn = false;
32        }
33      });
34    },
35  };
36  </script>
```

O estado global da aplicação é gerenciado pela biblioteca externa *Vuex*, uma biblioteca de gerenciamento de estado que atua como um unidade central para todos os componentes em um aplicativo. A maturidade e a robustez do Vuex, como uma biblioteca amplamente adotada e bem estabelecida na comunidade Vue.js, também foram fatores decisivos. Enquanto outras opções como Pinia, por exemplo, podem oferecer certas vantagens ou funcionalidades adicionais, o Vuex já se provou ser uma solução confiável para gerenciamento de estado em aplicações Vue.js de diferentes escalas. Esta confiabilidade, juntamente com sua capacidade de facilitar a comunicação entre componentes e manter o estado da aplicação previsível, foram consideradas prioridades para o projeto “Eduquiz”.“

Além disso, o roteamento da aplicação é implementado usando o *Vue Router*. Essa biblioteca possibilita a criação de *Single Page Applications (SPA)*, caracterizadas por carregar dinamicamente diferentes partes da aplicação conforme o usuário navega, evitando a necessidade

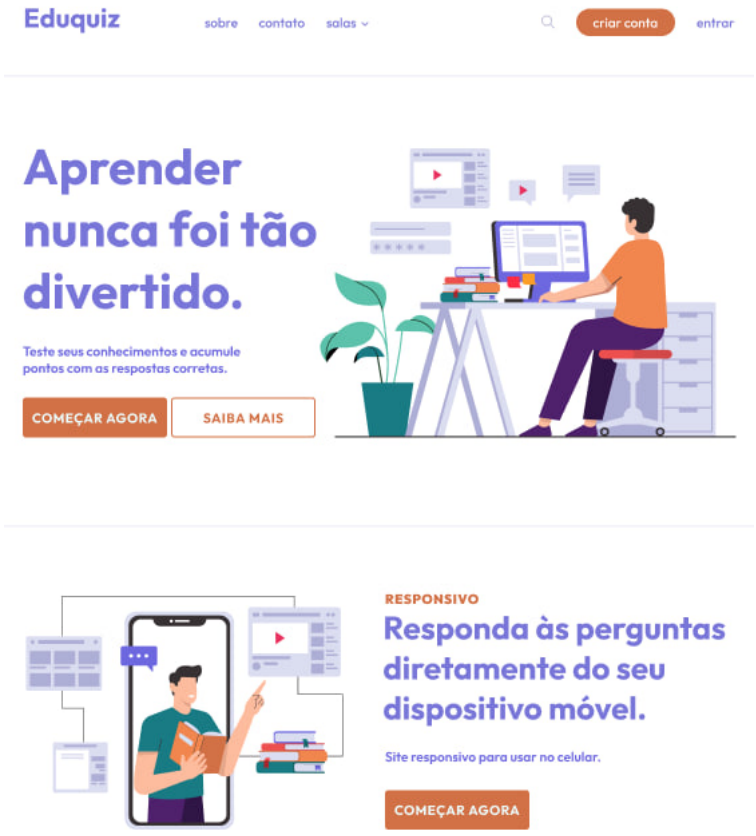
de atualizações de página completas. Por fim, o *Vue Router* oferece a possibilidade de criar rotas protegidas, garantindo a segurança da navegação.

Como mencionado em capítulos anteriores, a experiência de usuário é bastante importante para manter o engajamento dos usuários e, no contexto da educação, esse engajamento reflete-se diretamente na absorção de conteúdo. Baseado nessa premissa, conceitos de UI/UX foram utilizados na concepção das diferentes telas do aplicativo.

Todas as principais telas do aplicativo foram feitas utilizando o *Figma*, afim de planejar com detalhes a experiência de usuário antes do desenvolvimento. Para uma visão mais aprofundada de todas as telas, consultar o apêndice.

O software deve seguir a linha de *single-page application*, contando com uma tela inicial de boas-vindas na qual o usuário pode ter mais informações sobre a aplicação no geral. Observe a figura 3 abaixo.

Figura 3 – Página de boas-vindas “Eduquiz” no figma



Fonte: Figura do autor, (2023).

A figura 3 representa a página inicial da aplicação web. A escolha da paleta de cores principal foi uma parte integral do processo de design. Os tons de roxos são frequentemente associados à sabedoria e criatividade, enquanto os laranja são conhecidos por seus efeitos

energizantes e estimulantes.

Além disso, a tipografia arredondada transparece uma ideia amigável e acessível, o que é importante para ambientes educacionais. Na página inicial, os internautas podem obter mais informações gerais da aplicação e porquê ele seria ideal para o seu aprendizado.

Sendo uma *single-page application*, o software possui como principal elemento de navegação a barra de navegação, na qual todas as ações aceitas por cada página são exibidas.

Na página inicial, como mostra a figura 4, a barra de navegação exibe os botões padrão de “Entra” e “Login”. Esta configuração fornece aos novos usuários a opção de se registrarem ou fazerem login, tornando fácil para eles começarem a usar o aplicativo.

Figura 4 – Barra de navegação inicial

Fonte: Figura do autor, (2023).

Uma vez que o usuário esteja logado, como mostra a figura 5, a barra de navegação muda para exibir opções relevantes para usuários autenticados. Isso inclui botões para acessar seus questionários já existentes, criar um novo questionário e para acessar seu perfil. Essa mudança contextual na barra de navegação é uma prática recomendada em design de UI/UX, pois fornece aos usuários as opções que são mais relevantes para eles em um determinado momento.

Figura 5 – Barra de navegação usuário logado

Fonte: Figura do autor, (2023).

Na tela de criação de questionário, a barra de navegação adapta-se mais uma vez, como visto na figura 6, oferecendo novas opções de interação conforme o contexto exige. Isso garante que os usuários sempre tenham acesso às funcionalidades que precisam, independentemente de onde estejam no aplicativo.

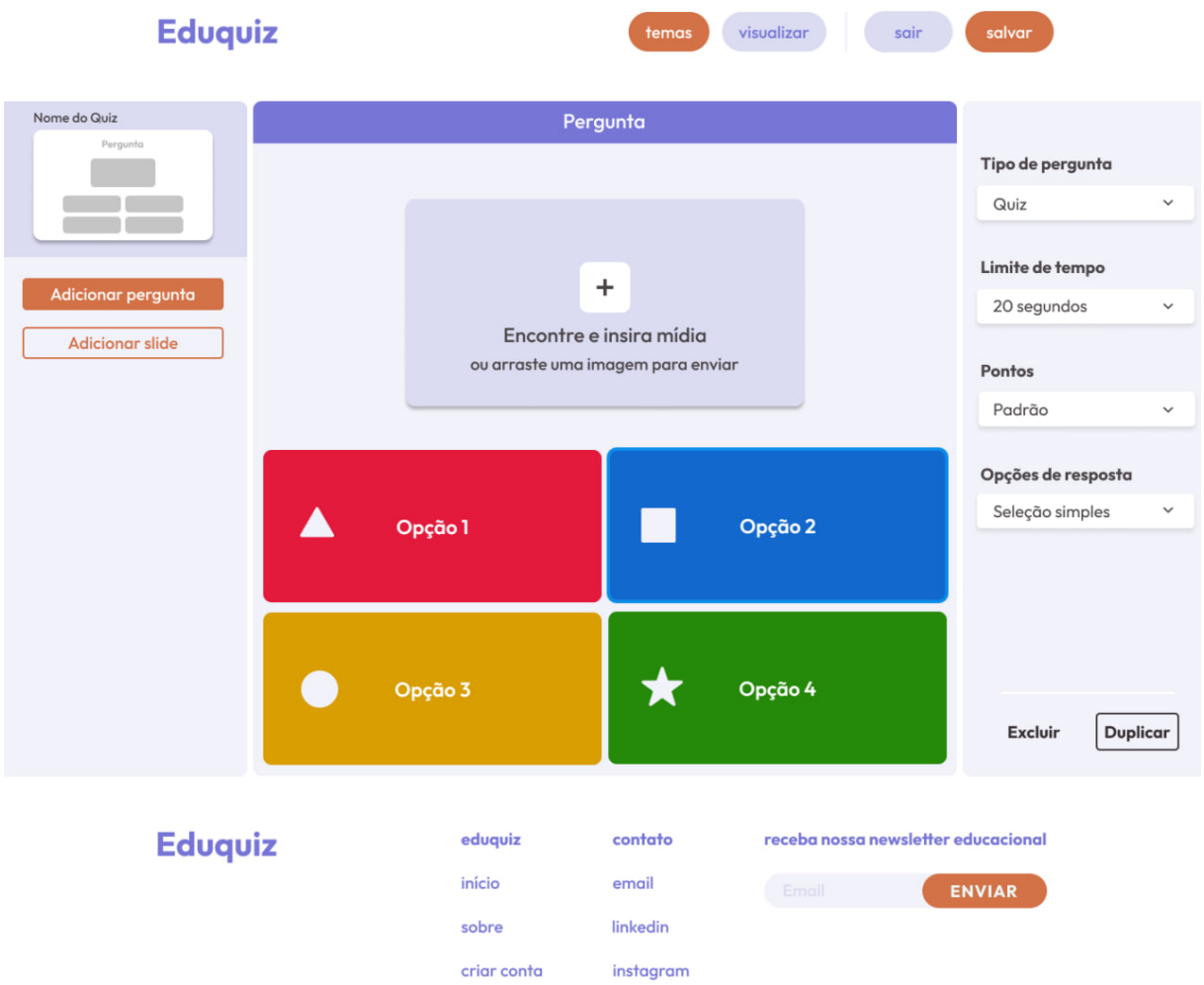
Figura 6 – Barra de navegação em criar questionário

Fonte: Figura do autor, (2023).

Outra página importante concebida no *Figma* foi a de criação de questionário. Esta página precisa ser fácil de utilizar e contar com todos os botões de ação para o usuário pode

escolher o tema do seu questionário, o tipo de pergunta que gostaria de inserir, as configurações gerais e respostas corretas. Além disso, o usuário precisa ser capaz de visualizar o seu questionário em edição. A figura 7 mostra a versão finalizada no *Figma* da página em questão.

Figura 7 – Página de criação de questionário



Fonte: Figura do autor, (2023).

3.3 Arquitetura do Back-end

Toda a arquitetura do "Eduquiz" foi feita usando microsserviços e computação em nuvem. Devido a natureza da aplicação, essa escolha determina o desempenho, escalabilidade e capacidade geral de atender às necessidades dos usuários. A arquitetura é baseada nos serviços do *Firebase*, mantidos pelo Google. A escolha do *Firebase* como plataforma em nuvem para o "Eduquiz" em detrimento de outras plataformas é pautada principalmente em dois aspectos: facilidade de uso e completude.

No que diz respeito à facilidade de uso, o *Firebase* se destaca por sua abordagem

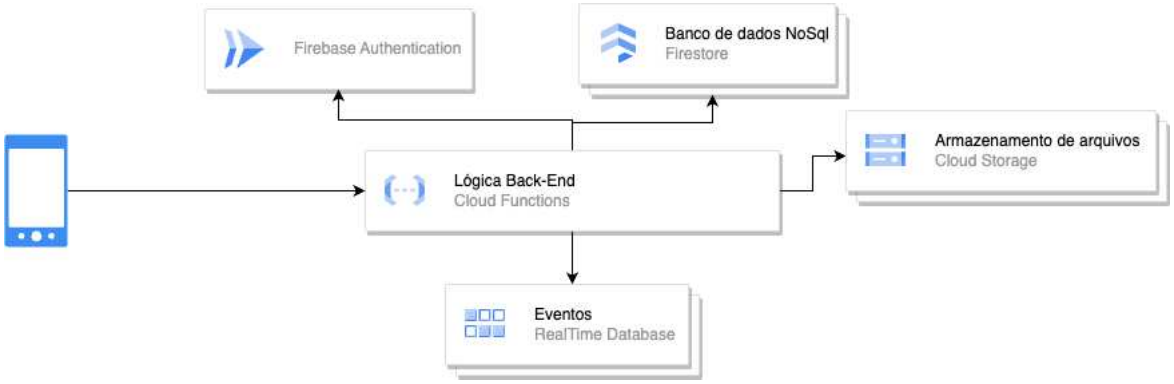
amigável ao desenvolvedor e por seu Software Development Kit (SDK) de fácil compreensão. A interface intuitiva, a documentação detalhada e a forte comunidade de desenvolvedores contribuem para uma experiência suave e sem complicações, permitindo que os desenvolvedores se concentrem no que é mais importante: a criação de funcionalidades de alto valor agregado para os usuários finais. Além disso, o *Firebase* fornece uma ampla gama de serviços gerenciados que minimizam a quantidade de trabalho de infraestrutura necessário, permitindo um tempo de desenvolvimento mais rápido e eficiente.

Quanto à completude, o *Firebase* oferece uma suíte abrangente de serviços em nuvem, cobrindo uma ampla gama de necessidades de desenvolvimento de aplicativos. Desde o gerenciamento de autenticação de usuários, passando pelo armazenamento de dados em tempo real, até a análise de comportamento do usuário e a entrega de notificações, o *Firebase* é uma solução completa para o desenvolvimento e crescimento de aplicativos. Essa completude não só permite a rápida prototipagem e implementação de funcionalidades, mas também facilita a escalabilidade, já que todos os serviços são integrados e projetados para trabalhar juntos de forma eficiente.

Em termos práticos, o projeto torna-se rapidamente escalável devido à grande flexibilidade da computação em nuvem. Com um aumento do número de clientes conectados, os servidores web do *back-end*, de modo transparente, replicam-se para suprir a demanda. Diversas configurações podem ser ajustadas parar dar as diretrizes desse escalonamento, como região preferida, números de *clusters*, número de usuários conectados e eventos de tipo gatilho em uma base de dados.

De modo mais detalhado, toda a arquitetura se baseia em *Cloud functions*, códigos executados no servidor de maneira *serverless*. Observe a figura 8 abaixo.

Figura 8 – Arquitetura principal



Fonte: Figura do autor, (2023).

Em todas as possibilidades nas quais o cliente precisa comunicar-se com o back-end, ele o fará por meio de funções *serverless*. Exemplos de momentos podem ser: ao autenticar-se com o servidor, ao fazer requisições HTTP para atualizar algum dado no front-end, etc. Do lado do servidor, todo tipo de manipulação de dados e *login* de usuários será feita também por funções *serverless*. A figura 8 demonstra bem essa característica.

No quesito banco de dados, a solução proposta foi o *Firestore* também dentro da gama de serviços do *Firebase*. Trata-se de um banco de dados de documentos, onde cada documento é identificado por uma chave única e pode conter vários pares de chave-valor. As razões principais para a escolha deste tipo de banco de dados foram a escalabilidade, modelo de dados flexível e dados em tempo real. No primeiro ponto, a praticidade de ser um serviço na nuvem com gestão do Google garante uma boa escalabilidade de forma natural. No que tange ao segundo ponto, sendo uma estrutura de dados em forma de documentos, há uma grande flexibilidade na modelagem dos dados e também na sua alteração mesmo com a aplicação já em ambiente de produção. Finalmente, o *Firestore* suporta a atualização em tempo real de dados por meio de *WebSockets*, o que é fundamental para a experiência do usuário do "Eduqui". Assim, sempre que um documento é atualizado, todas as instâncias do aplicativo que estão observando esse documento são notificadas da atualização, permitindo uma experiência de usuário dinâmica e reativa.

3.3.1 Definição do banco de dados

Baseado nos requisitos funcionais do sistema, o descritivo do banco de dados foi traçado como sendo:

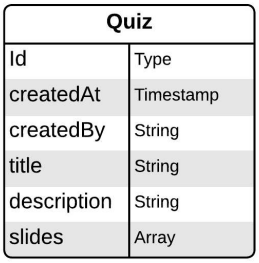
1. Usuários devem poder criar questionários e editá-los. Questionários compõem um conjunto de perguntas.
2. Associado a cada pergunta, existe um conjunto de opções de resposta. Somente uma resposta é correta. Além disso, para cada pergunta, um temporizador pode ser atrelado, junto com um sistema de pontuação específico.
3. Usuários podem criar salas de jogo. Uma sala de jogo compõe um questionário específico com o seu respectivo conjunto de perguntas. Uma sala de jogo incorpora a lógica principal de jogo da aplicação e deve refletir o estado atual da partida, com as perguntas atuais e os jogadores participantes.
4. Durante uma partida, jogadores podem submeter suas respostas para a pergunta atual,

respeitando o temporizador, se existir.

Desse descritivo, foi criado o banco de dados *NoSql*, atendando-se às características específicas desse tipo de banco.

De forma concreta, as coleções criadas são descritas abaixo, com seus respectivos campos e tipo de dados.

Figura 9 – Documento da coleção *Quizzes*



Fonte: Figura do autor (2023).

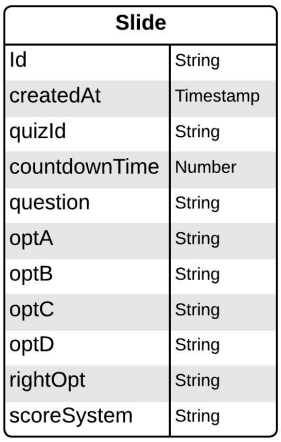
A figura acima 9 exemplifica um documento da coleção *quiz*. Esse documento representa um questionário que pode ser criado e alterado pelo usuário. *createdAt* e *createdBy* representam a data de criação do questionário em *timestamp* e o *id* do criador, respectivamente. Por sua vez, *title* e *description* informam o título e descrição do questionário em questão. Por fim, *slides* é um vetor que contém os *ids* de todas as perguntas do questionário.

Código-fonte 2 – Um documento na coleção Quizzes

```
"quizId": {
        "createdAt": "timestamp",
        "createdBy": "userId",
        "title": "Title",
        "description": "Desc",
        "slides": [
            "slideId1",
            "slideId2",
            "slideId3",
        ]
    }
```

O código-fonte acima exemplifica o modelo real de um documento do tipo *quiz* presente no banco de dados.

Figura 10 – Documento da coleção *Slides*



Fonte: Figura do autor (2023).

A figure 10 acima exemplifica um documento da coleção *Slides*. Além do campo idêntico *createdAt*, este documento atrela-se a um questionário específico baseado no campo *quizId*. Um *slide* contém as informações de uma pergunta, com o título (*question*), opções de resposta (*optA*,*optB*,*optC*, *optD*) e a resposta correta (*rightOpt*). Além disso, o documento também apresenta campos de configuração de cada pergunta, como o temporizador (*contdownTime*) e o sistema de pontuação (*scoreSystem*). Essa lógica atrelada à pergunta garante que cada uma possa ser configurada de um modo específico.

Código-fonte 3 – Um documento na coleção Slides

```
"slideId": {
        "createdAt": "timestamp",
        "quizId": "quizId1",
        "countdownTime": 20,
        "question": "who's there?"
        "optA": "option",
        "optB": "option",
        "optC": "option",
        "optD": "option",
        "rightOpt": "optA",
```

```
        "scoreSystem": "default",
    }
```

O código-fonte acima exemplifica o modelo real de um documento do tipo *slide* presente no banco de dados.

Figura 11 – Documento da coleção *Rooms*

Room	
Id	String
createdAt	Timestamp
createdBy	String
gameState	String
hostId	String
quizId	String
numberOfSlides	Number
currentSlidePos	Number
currentSlideId	String
currentSlide	Map
roomPlayers	Array

Fonte: Figura do autor (2023).

A figura 11 acima descreve um documento da coleção de *rooms*, que representa uma sala de jogo. Este documento encapsula todos os campos de configuração necessários para o desenrolar de uma partida do jogo. Fora os campos supracitados, o campo *hostId* contém o *id* do anfitrião da partida, que controla o jogo. Baseado nos dados enviados pelo anfitrião, o campo *gameState* modifica o seu valor para refletir o estado atual do jogo (começar jogo, pergunta em andamento, mostrar resultados, terminar jogo). *gameState* aceita os seguintes campos: *"CREATED"*, *"STARTED"*, *"NEXT"* e *"RESULTS"*. Baseado no valor desse campo, eventos são iniciados por meio de *cloud functions* para a execução da lógica do jogo. Para cada pergunta em andamento, *currentSlide*, *currentSlideId* e *currentSlidePos* são atualizados, significando, respectivamente, o objeto da pergunta em questão, o *id* da pergunta e a posição da pergunta no questionário. For fim, *roomPlayers* trata-se de um vetor contendo todos os *ids* dos usuários conectados na sala de jogo. O campo *roomPlayers* é atualizado em tempo real baseado nos jogadores conectados.

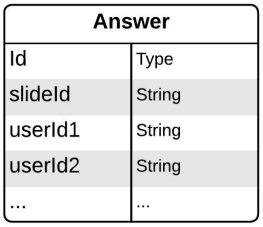
Código-fonte 4 – Um documento na coleção rooms

```
"roomId1": {
```

```
    "createdAt": "timestamp",
    "createdBy": "userId1",
    "gameState": "START",
    "hostId": "userId1",
    "quizId": "quizId1",
    "numberOfSlides": 3,
    "currentSlidePos": 0,
    "currentSlideId": "slideId1",
    "currentSlide": slide,
    "roomPlayers": [
        "userId2",
        "userId3",
        "userId4"
    ]
}
```

O código-fonte acima exemplifica o modelo real de um documento do tipo *room* presente no banco de dados.

Figura 12 – Documento da coleção *Answers*



Fonte: Figura do autor (2023).

Por fim, a figura 12 representa o modelo de um documento de resposta do tipo *answers*. Cada documento deste tipo está atrelado a um pergunta. Para cada usuário, um campo é criado contendo a resposta fornecida.

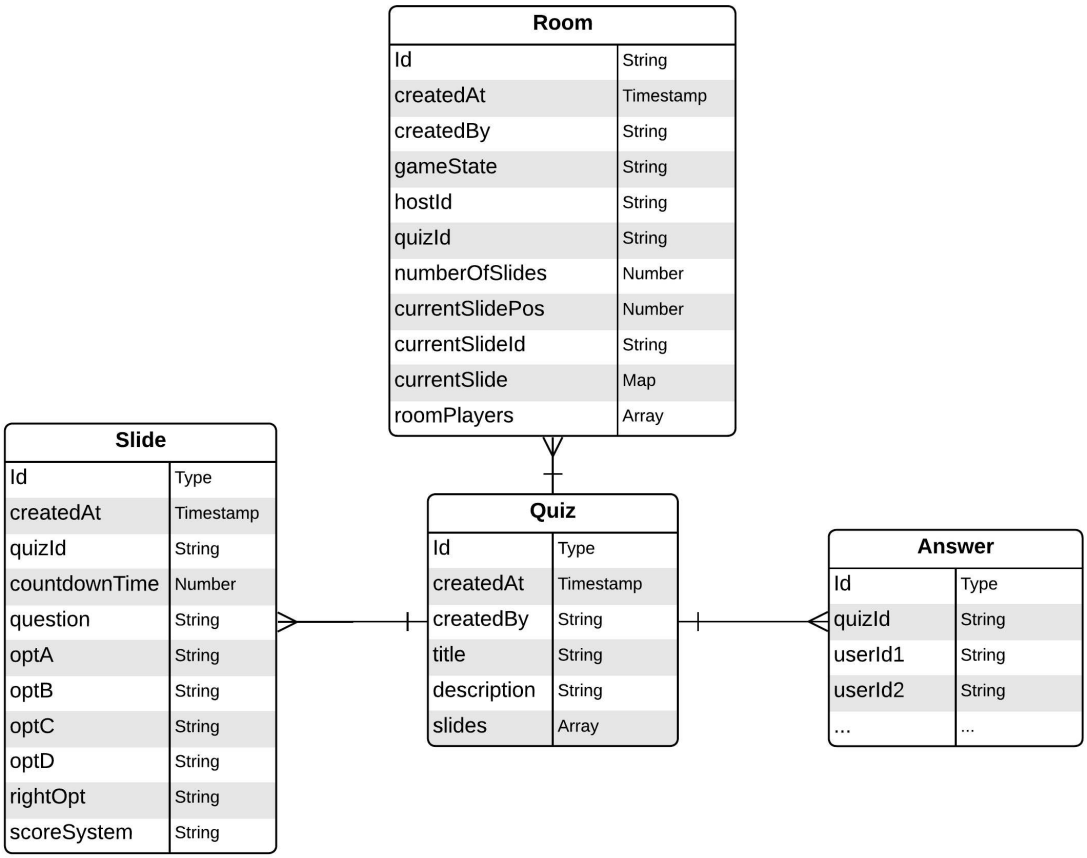
Código-fonte 5 – Um documento na coleção Answers

```
"answerId1": { "createdAt": "timestamp",
    "userId1": "option",
```

```
    "userId2": "option",
    "userId3": "option",}
```

A descrição acima desconsidera toda a infraestrutura de *WebSockets* associada, que torna possível a atualização em tempo real de todos os clientes. Toda a lógica envolvendo o controle de usuários logados e online, e os mecânicos de sincronização de contadores são feitos utilizando o banco de dados em tempo real, propício para tal. Observe a figura 13 abaixo que apresenta o esquemático completo do banco de dados.

Figura 13 – Esquemático completo do banco de dados *NoSql*



Fonte: Figura do autor (2023).
Nota: Dado que trata-se de um banco de dados não-relacional, o diagrama ER apresentado representa uma simplificação das coleções usadas.

4 RESULTADOS E DISCUSSOÕES

Este capítulo concentra-se sobre a análise e apresentação dos resultados alcançados após a fase de desenvolvimento inicial do “Eduquiz”. Esta análise fornece uma visão detalhada do processo de construção tanto do ponto de vista técnico quanto da experiência do usuário.

Será discutido inicialmente todas as jornadas de usuário relevantes que foram implementadas na aplicação, juntamente com as escolhas técnicas de computação em nuvem. Além disso, toda a experiência de usuário e design da interface será levada em consideração e como essas características contribuíram para a eficácia geral da plataforma.

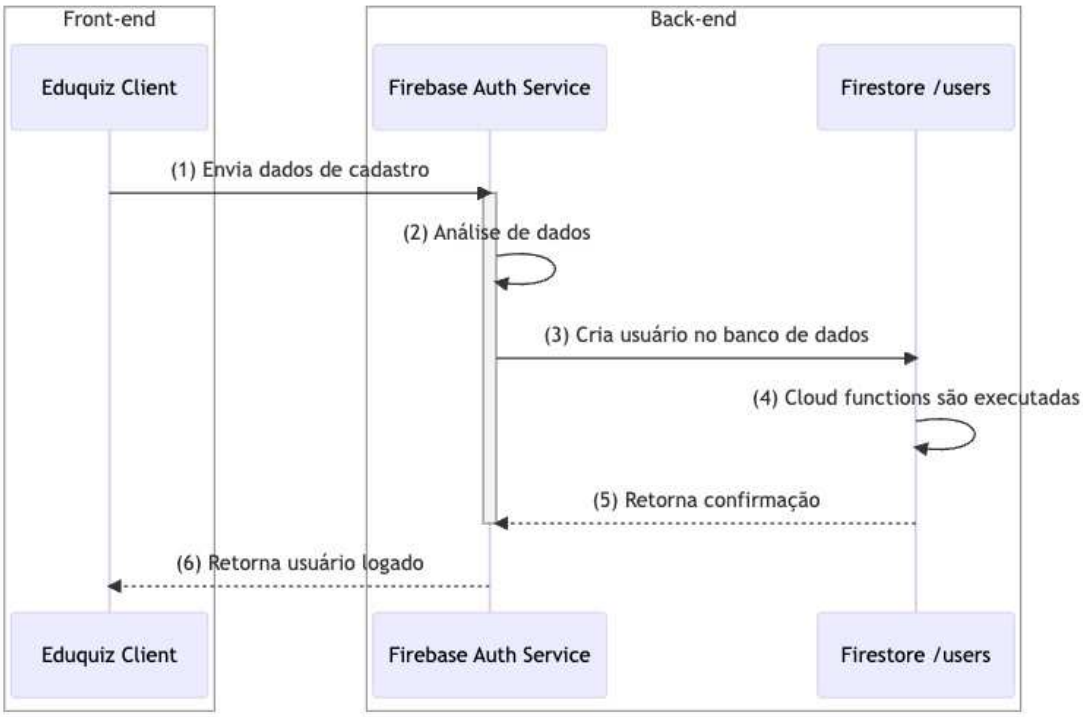
4.1 Jornadas de usuário

O desenvolvimento do “Eduquiz” foi separado em diferentes jornadas de usuários que, eventualmente, deram origem às primeiras estórias de usuário. Dessas estórias, todas as funcionalidades foram derivadas. Abaixo segue uma descrição detalha de todas as jornadas de usuário relevante que foram implementadas utilizando os serviços do *Firebase*.

4.1.1 Jornada de cadastro e login

A primeira dessas jornadas de usuário trata-se de uma da mais essenciais em toda aplicação: a jornada de cadastro e *login* de contas. Observe a figura 14 abaixo.

Figura 14 – Jornada de usuário feliz de cadastro e *login*



Fonte: Figura do autor (2023).

O envio e troca de mensagens dá-se da seguinte maneira em um cenário feliz:

(1) a aplicação cliente envia os dados de cadastro do usuário por meio do *SDK* proprietário do *Firebase*.

(2) O serviço de back-end de autorização do *Firebase* analise os dados enviados e certifica-se de que estão conformes.

(3) Uma vez aprovados, o serviço de autenticação, por meio de *Cloud functions* cria um novo documento na coleção *users*.

(4) Mais *Cloud functions* são executadas para garantir que todas as coleções estão em sincronia.

(5),(6) Uma vez toda a configuração feita, os dados do novo usuário retornam de volta para o cliente da aplicação.

Dada as circunstâncias, se algum problema ocorrer durante a sequência de troca de mensagens, *Firebase* retornará uma lista de erros que pode ser interpretada pelo cliente. Os

Tabela 2 – Principais código de erros gerados por *Firebase Authentication*.

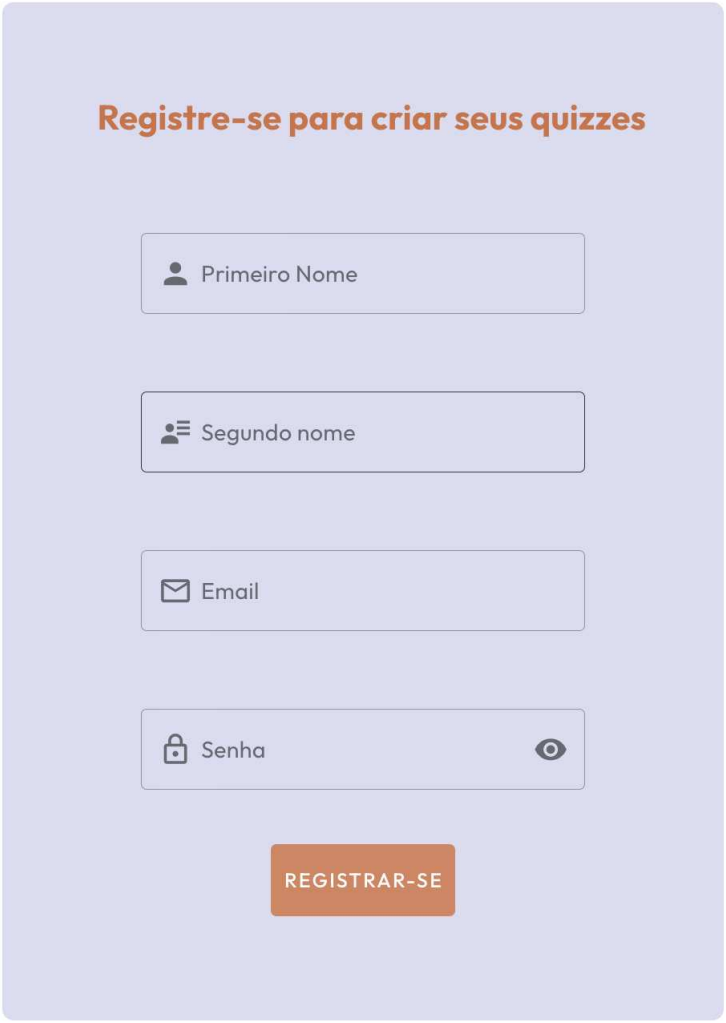
Código de erro	Descrição
auth/invalid-email	O valor fornecido para a propriedade do usuário email é inválido.
auth/user-already-found	Usuário já cadastrado.
auth/invalid-password	O valor fornecido para a propriedade do usuário senha é inválido.

Fonte: elaborado pelo autor (2023).

códigos de erro da etapa de *login* seguem o mesmo padrão, com erros adicionais para e-mails e senhas ja cadastradas. Para uma visão completa dos códigos de erro do *Firebase Authentication*, consultar apêndice.

Em termos de UI, as sessões de cadastro e *login* tiveram como visual abaixo, já com as lógicas de erro implementadas.

Figura 15 – Tela de cadastro *login*



Fonte: Figura do autor (2023).

Figura 16 – Tela de login *login*

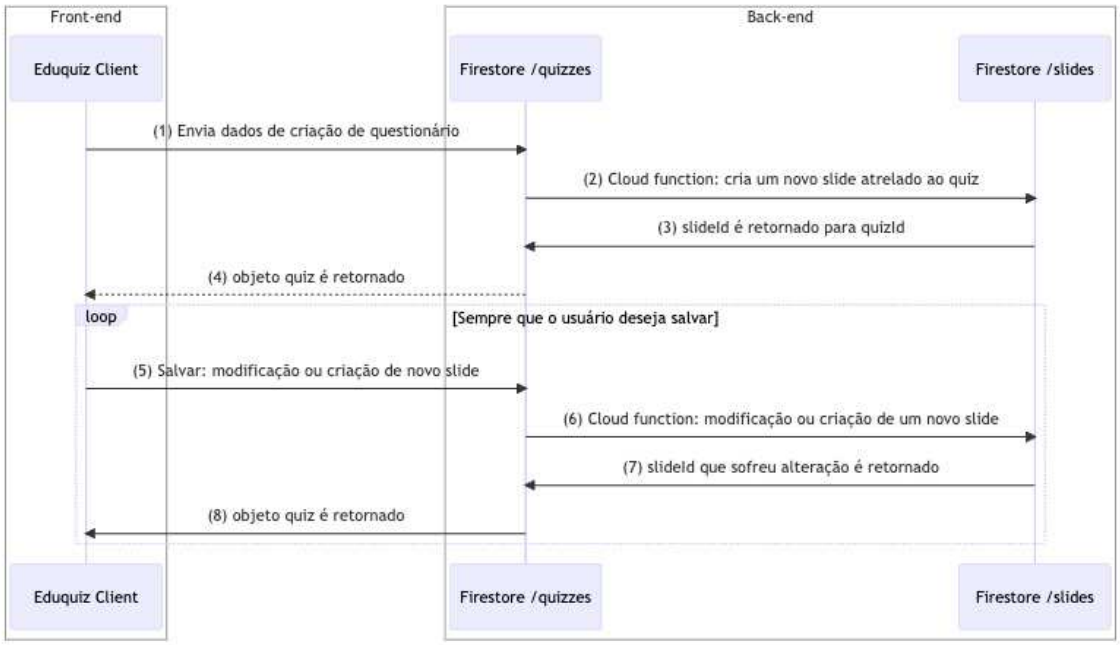
Fonte: Figura do autor (2023).

O processo de desenvolvimento foi bastante simplificado e organizado, devido a utilização do *Firebase authentication*. Com essa ferramenta, não só toda a parte de segurança foi abstraída e garantida pelos servidores do Google, mas também a garantia de escalabilidade eficiente e de tratamento de erros.

4.1.2 Jornada de criação/edição de questionários

No que se refere à criação de um questionário, o usuário logado realiza a jornada descrita na figura 17 abaixo.

Figura 17 – Jornada de usuário feliz de criação de questionário



Fonte: Figura do autor (2023).

O envio e troca de mensagens dá-se da seguinte maneira em um cenário feliz:

(1) Uma vez logado, o usuário clica no botão de criação de questionário, onde preenche o título e descrição. Os dados são enviados para o servidor e um novo questionário é criado. Como campos padrões, se tem *createdBy*, que identifica o usuário que criou o questionário; e *quizId*, que representa o identificador do questionário.

(2), (3) Por meio de *cloud functions*, um novo documento do tipo slide, que representa o layout de uma pergunta de um questionário, é criado na coleção de slides e é atrelado por meio de seu *slideId* ao conjunto de slides de um questionário.

(4) O objeto quiz com todas as informações do questionário são retornados para o cliente, que pode agora mostrar os dados na tela.

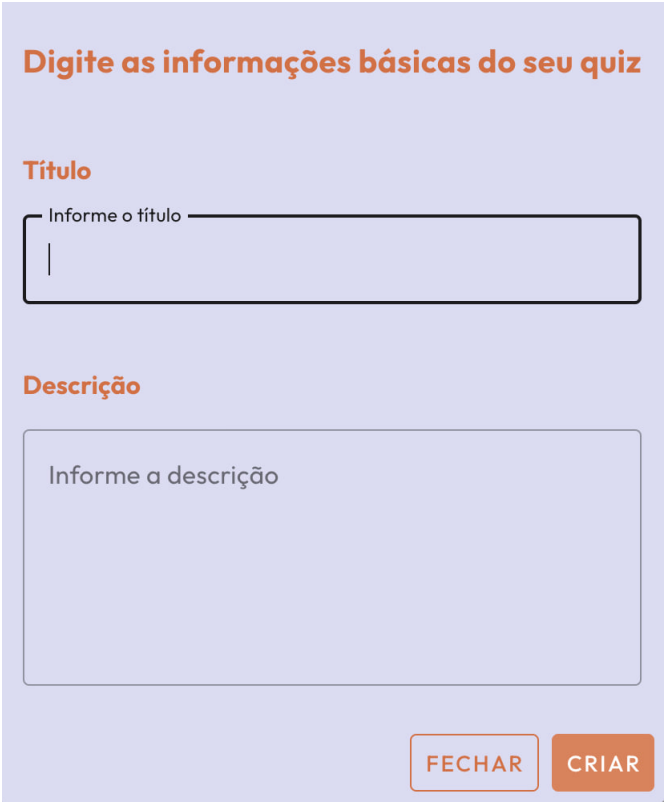
(5),(6),(7),(8) Sempre que o usuário realizar alguma modificação em um slide, ele tem a possibilidade de salvar. Uma vez clicado no botão de salvar, as modifições são reprodizadas do mesmo modo descrito acima, no final, retornando o objeto de quiz atualizado. Nesse

procesos, slides atuais podem ser modificados e/ou excluídos e novos também podem ser criados.

A característica mais importante de um questionário são os slides que ele contém. Assim, convém explicar com detalhes do que se trata um objeto do tipo slide. Um slide é a unidade básica de um questionário. Semanticamente, esse objeto representa uma pergunta, com o título da pergunta, uma imagem de referência, um conjunto de opções selecionáveis e uma resposta correta. Além disso, existem outros parâmetros como o tipo de pergunta, o tempo de resposta máximo para a pergunta e o tipo de pontuação associada ao slide. O objetivo do "Eduqui" é ser altamente personalizável, para tal, a possibilidiade de modificação de slides é essencial.

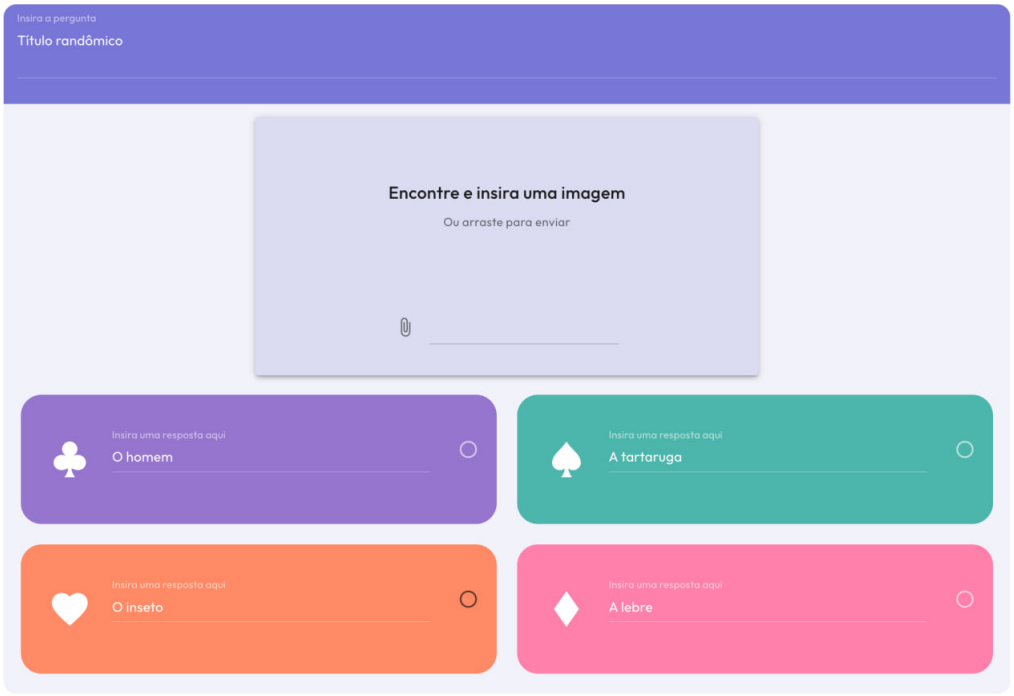
Em termos de UI, as etapas de criação e edição de quiz tiveram como visual o apresentado abaixo, já com as lógicas de erro implementadas.

Figura 18 – *pop-up* de criação de questionário



Fonte: Figura do autor (2023).

Figura 19 – Tela de edição de questionário



Fonte: Figura do autor (2023).

A figura 19 ilustra como de fato um slide se parece, mostrando as secções de pergunta, respostas e imagens atreladas.

Outro ponto interessante é a reatividade que a interface tem em relação às ações do usuário. Este pode modificar diversos aspectos dos slides e poderá ver em tempo real as modificações. Um exemplo disso é a seleção da resposta correta.

A figura 20 abaixo ilustra como a UI se adapta à resposta escolhida pelo usuário no canto esquerdo da tela de criação de questionário. A resposta correta foi a de tipo *coração*, por isso a cor diferenciada.

Figura 20 – Menu auxiliar na lateral esquerda da tela de edição.

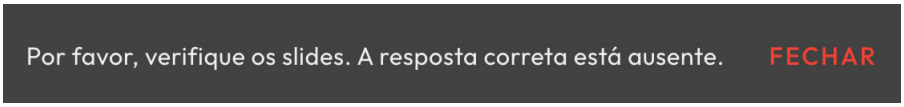
Fonte: Figura do autor (2023).

Outro exemplo de reação da UI se encontra na barra de navegação principal, na parte de cima da aplicação. Na presença de itens não salvos, um alerta com animação é mostrado e, caso o padrão de slide estiver incorreto, uma mensagem de erro é mostrada ao usuário. Este comportamento é descrito nas figuras 21 e 22 abaixo.

Figura 21 – Barra de navegação superior que se modica baseado nas ações do usuário.

Fonte: Figura do autor (2023).

Figura 22 – Quadrado de erro de slide.

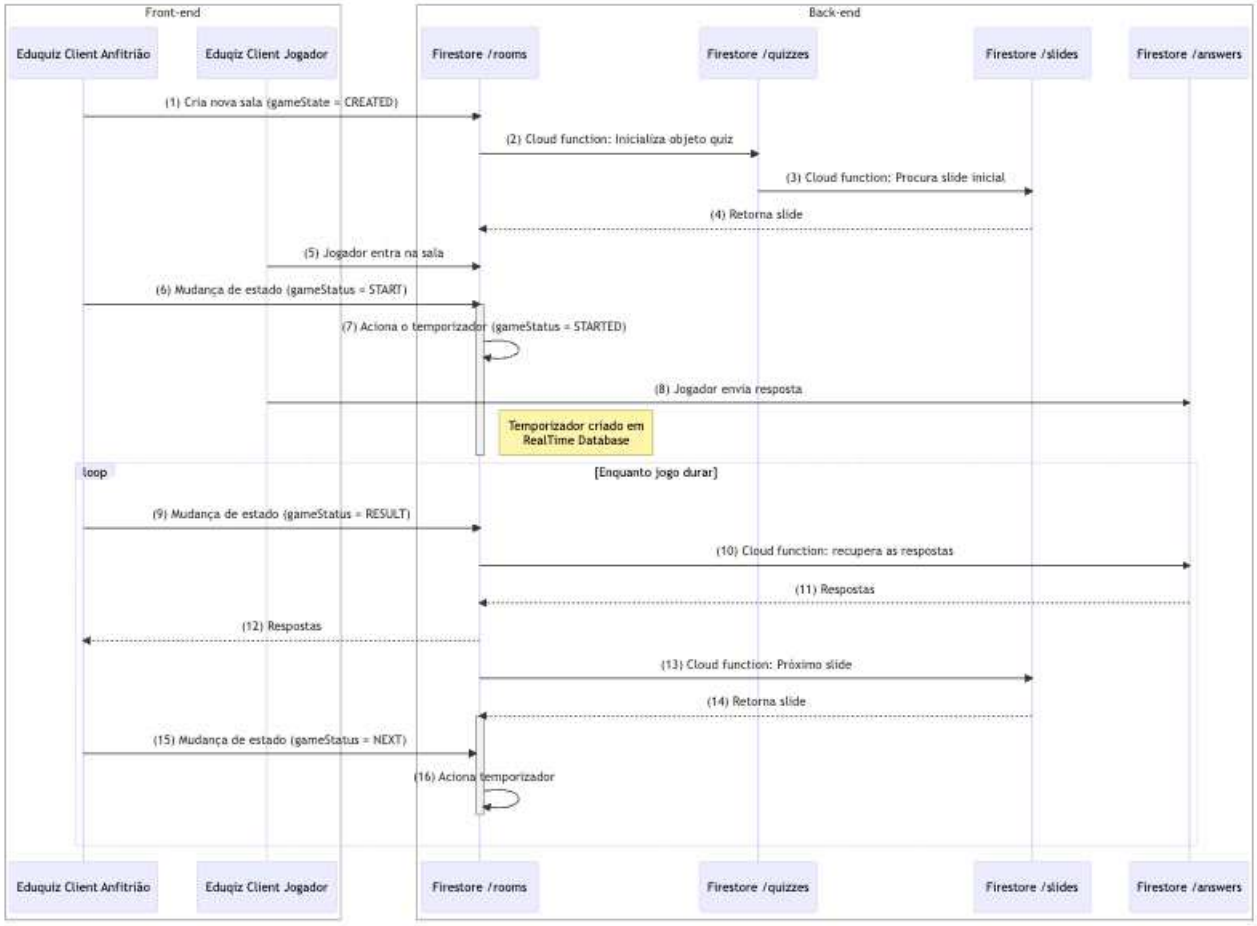


Fonte: Figura do autor (2023).

4.1.3 Jornada de jogo

No que se refere à partida em si, o primeiro usuário a clicar em jogar em um questionário, torna-se o anfitrião da sala e responsável por dar continuidade ao jogo. Os usuários que entram após a criação são jogadores e podem participar do jogo de forma anônima se desejarem. A partir desse ponto, toda a comunicação entre clientes e servidores é feita por meio de *WebSockets* em tempo real. A figura abaixo representa um diagrama de sequência que mostra como o jogo se desenrola com ações do cliente e do back-end da aplicação.

Figura 23 – Diagrama de sequência de uma partida.



Fonte: Figura do autor (2023).

O envio e troca de mensagens dão-se da seguinte maneira:

- (1) Ao clicar em jogar, o cliente do Eduquiz na condição de anfitrião envia uma requisição de criação de sala na coleção de salas, estabelecendo o estado do jogo para “CREATED”.
- (2) O *Firestore*, em resposta à criação da sala, aciona uma função para inicializar o objeto de quiz em na coleção *quizzes*.
- (3)-(4) A função acionada na etapa anterior busca o slide inicial na coleção de slides, por meio da referência encontrada no quiz. Uma vez encontrado, o slide é atualizado no documento da sala. Uma vez que toda a lógica de uma sala é implementada utilizando *WebSockets*, a informação é refletida de volta ao cliente.
- (5) Um segundo cliente do Eduquiz, um jogador, entra na sala criada por meio de uma *URL* de acesso. De forma anônima ou não, o usuário é adicionado ao conjunto de jogadores da sala. O jogo continua em modo de espera aguardando uma chamada de inicialização.
- (6) O cliente anfitrião solicita uma mudança de estado por meio da atualização do campo

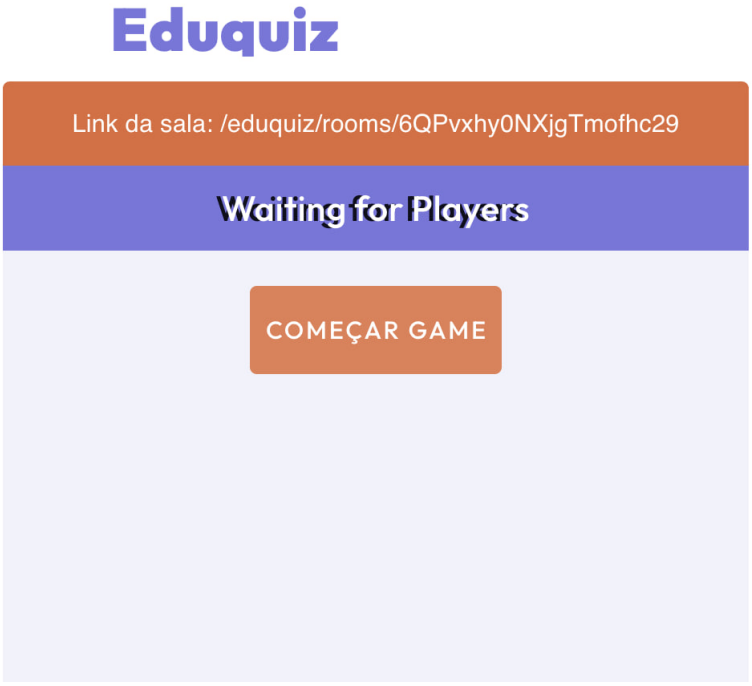
gameState para “START”, que dá início ao jogo.

- (7) Com o jogo iniciado, um evento de modificação do campo *gameState* aciona um temporizador associado ao slide em questão. Esse temporizador é iniciado por meio de documentos na *RealTime Database*.
- (8) Com uma lógica interna, os clientes sincronizam seus temporizadores com o do servidor e, durante esse tempo, os jogadores enviam suas respostas para coleção de respostas.
- (9) Após o fim do tempo estabelecido, o cliente anfitrião pode solicitar os resultados, alterando mais uma vez o campo *gameState* para “RESULT”.
- (10)-(14) Devido a atualização no campo *gameState*, um evento aciona uma função na nuvem para recuperar as respostas dos jogadores na coleção de respostas, já calculando as pontuações baseadas no slide em questão. Todas os campos são atualizados e um novo slide é recuperado.
- (15) O cliente anfitrião solicita uma mudança de estado novamente, alterando o estado do jogo para “NEXT”.
- (16) O *Firestore* aciona um temporizador ao receber a solicitação para avançar para o próximo slide, re-iniciando o ciclo até que o jogo acabe.

A grande vantagem dessa arquitetura é a reatividade. Utilizando o *SDK* do *Firebase* para gerenciar os documentos e campos que são designados para a utilização de *WebSockets* torna a aplicação mais dinâmica e resiliente a falhas. O acesso aos documentos são controlados: somente o anfitrião realiza mudanças no documento de uma sala, e o back-end, justamente com as *cloud functions*, reage aos eventos gerados.

Em termos de UI, a experiência de usuário para as salas e jogos do “Eduquiz” são as mais interessantes. Uma vez uma sala criada, a *UI* mudará o visual a depender do tipo de usuário. Para o anfitrião, botões de controle serão mostrados, pelos quais o usuário pode controlar o andamento da partida, como iniciar o jogo, mostrar resultados e avançar. Do lado do jogador, este pode definir o seu apelido, uma vez que este tipo de usuário não precisa estar registrado no sistema; aguardar o início da partida e responder às questões. Nas figuras abaixo serão mostrados esses componentes em detalhes.

Figura 24 – Layout de espera para anfitrião.



Fonte: Figura do autor (2023).

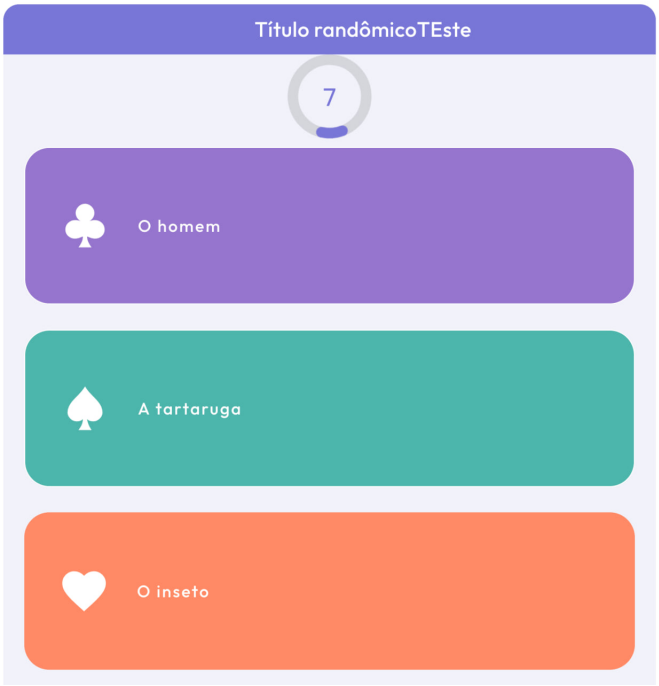
A figura 24 mostra o *layout* para um tela *mobile* da secção de espera de um usuário anfitrião. O *URL* pode ser usado para entrar na sala como jogador.

Figura 25 – Layout de espera para jogador

Fonte: Figura do autor (2023).

Como mostrado na figura 25, o usuário do tipo jogador que entrou pelo *URL* da partida, pode escolher um apelido e esperar pelo início do jogo.

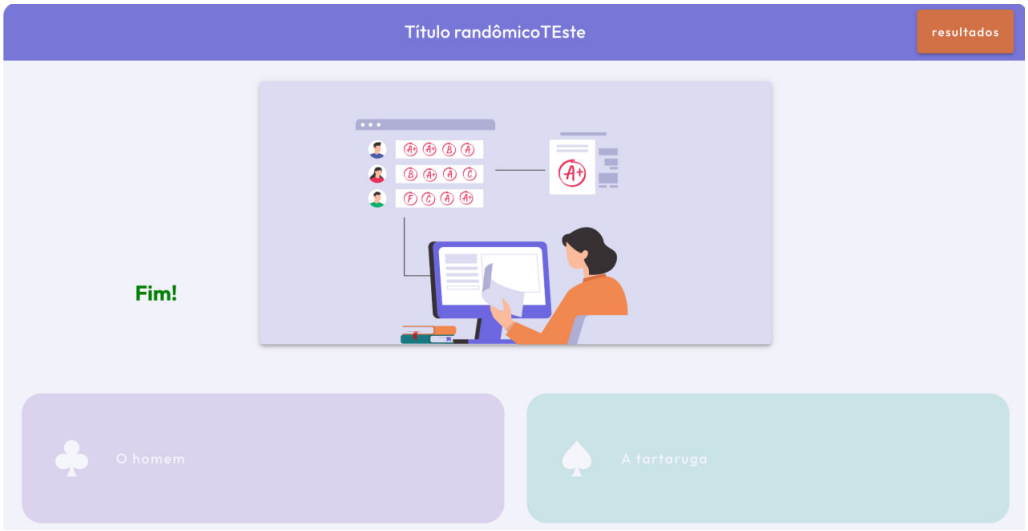
Figura 26 – Layout de jogo para jogador



Fonte: Figura do autor (2023).

Como mostrado na figura 26 uma vez dentro do jogo, os jogadores podem acessar os botões de resposta e escolher a correta, baseado em seus conhecimentos. É mostrado também o tempo restante para a resposta e uma vez terminado, os botões são desativados. Uma parte interessante deste design é a responsividade, a depender do tamanho da tela, a aplicação se ajusta para fornecer o máximo de detalhe possível. Por exemplo, como visto na figura 27, em telas maiores, é possível vizualizar a imagem de referência do slide.

Figura 27 – Layout de tela cheia.



Fonte: Figura do autor (2023).

4.2 Disposição dos componentes e fluxo de dados do *Front-end*

Como mencionado no capítulo de metodologia, a arquitetura de *front-end* do "Eduquiz" foi projetada totalmente sobre os conceitos de *Options API* do *Vue.js*. Nesse sentido, será apresentado nesta secção a estruturação efetiva das rotas da aplicação, bem como o fluxo de dados global e os componentes principais.

Código-fonte 6 – Rotas principais da aplicação

```
const router = createRouter({
  history: createWebHistory(),
  routes: [
    { path: "/eduquiz/", component: LandingPage },
    { path: "/eduquiz/register", component: Register },
    { path: "/eduquiz/login", component: Login },
    { path: "/eduquiz/create-quiz/:quizId", component:
       CreateQuiz },
    { path: "/eduquiz/my-quizzes", component: MyQuizzes },
    { path: "/eduquiz/public-rooms", component: PublicRooms
        },
    { path: "/eduquiz/rooms/:roomId", component: TheRoom },
  ],
});
```

- ***"/eduquiz/"***: página inicial da aplicação, contendo as informações básicas de aplicação, junto com alguns depoimentos e formas de utilizar. O componente principal desta aplicação é o *LandingPage.vue*, que implementa os sub-componentes padrões como botões, *cards*, cabeçalho e roda-pé.
- ***"/eduquiz/register"***: página de criação de uma nova conta. O componente *Register.vue* inicializa um formulário de cadastro padrão.
- ***"/eduquiz/login"***: página de entrada para um usuário já cadastrado. O componente *Login.vue* se assemelha bastante ao de cadastro. Nesta página é onde as proteções contra usuários sem cadastro são instanciadas.
- ***"/eduquiz/create-quiz/:quizId"***: página de criação de um novo questionário. O usuário

cadastrado cria um novo questionário e pode editá-lo por meio do componente principal *CreateQuiz*. Ao editar um questionário já existente, o usuário também é redirecionado para o mesmo *endpoint*.

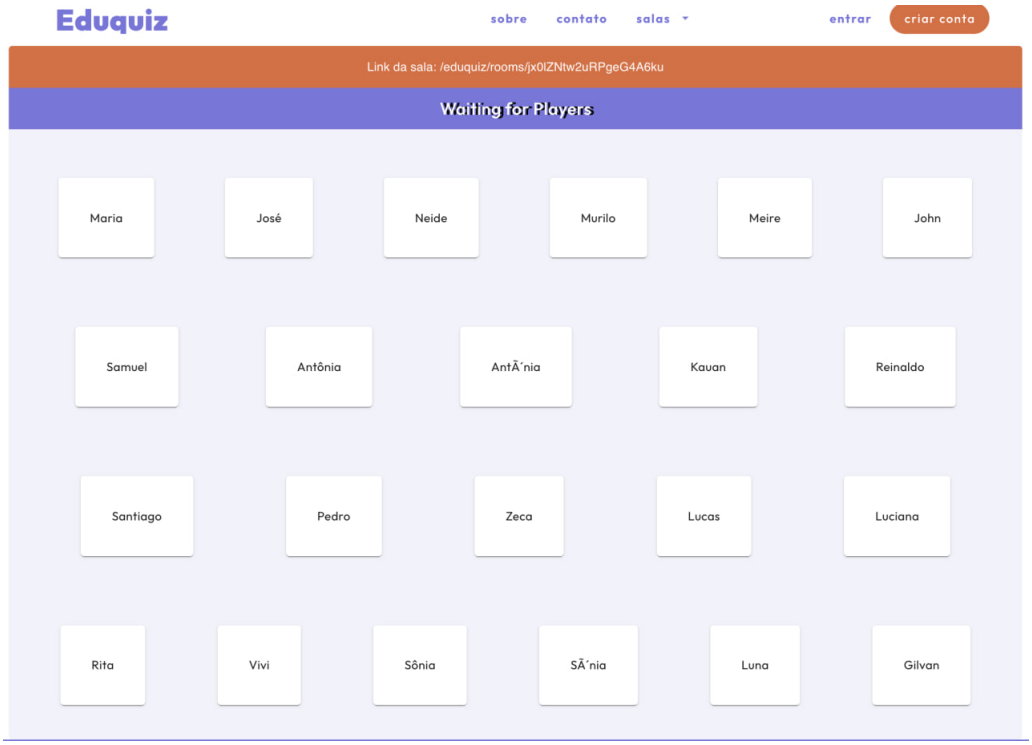
- ***"/eduquiz/my-quizzes"***: página na qual o usuário tem acesso a todos seus questionários já criados e pode iniciar uma sessão de jogo.
- ***"/eduquiz/public-rooms"***: página na qual um usuário tem acesso aos questionários públicos criados por outros usuários e tem a opção de iniciar uma nova sessão de jogo. O usuário não precisa estar cadastrado para ter acesso a essa página.
- ***"/eduquiz/rooms/:roomId"***: página principal de uma sessão de jogo. O componente principal *TheRoom* inicia uma conexão *WebSocket* com o servidor e inicia uma sessão de jogo com todos os usuários. Todas as telas principais do jogo são carregadas no componente principal, com toda a lógica de usuário anfitrião e jogadores embutida.

4.3 Testes back-end e performance do sistema

Nesta secção, será analisada as métricas do sistema e como ele responde às demandas de múltiplos usuários conectados ao mesmo tempo. Também será analisada o tempo de resposta das requisições e latências.

Uma sessão de jogo foi criada e 50 usuários foram adicionados à sala, representando um caso real de utilização. Além disso, esse cenário foi replicado para diferentes salas, ao todo, 50 salas foram criadas com as mesmas configurações. Finalmente, uma parcela desses usuários gerados foram conectados a partir de navegadores distintos. Essa configuração de teste totalizou 2500 usuários conectados.

Figura 28 – Sala teste com 50 usuários.



Fonte: Figura do autor (2023).

Sobre o tempo de conexão à sala e carregamento total da página, temos as seguintes características:

Tabela 3 – Comparações entre tempo de carregamento a depender do navegador.

Navegador	1º Exibição	1ºExibição Conteúdo	domContentLoadedEventEnd*	domInteractive**
Chrome 113.0	167 ms	167 ms	125 ms	18 ms
Firefox 114.0	161 ms	160 ms	100 ms	20 ms
Safari 16.4.1	186 ms	180 ms	123 ms	18 ms

Fonte: elaborado pelo autor (2023).

**domContentLoadedEventEnd*: uma métrica que mede o tempo entre o momento em que o usuário navega até uma página e o momento em que o documento HTML inicial é completamente carregado e analisado.

***domInteractive*: uma métrica que mede o tempo entre o momento em que o usuário navega até uma página e o momento em que ela é considerada interativa para ele.

A tabela 3 mostra pouca variação entre o tipo de navegador utilizado. Na média, todos os navegadores performaram bem, em condições ideias de conexão com a internet. Isso se deve, provavelmente, ao *framework* utilizado, Vue.js em sua versão 3 garante um pequeno *bundle* de arquivos finais que tornam rápida a execução em navegadores modernos.

No que se refere às funções em nuvem, foram analisadas as seguintes quanto ao tempo de execução e resposta:

Tabela 4 – *Cloud functions* principais do sistema

Nome	Ambiente	Região	tempo de resposta
newUserUpdate	Node.js 18	us-central1	67 ms
newRoomCreated	Node.js 18	us-central1	84 ms
eventReceivedByHost	Node.js 18	us-central1	156 ms
eventReceivedSetTimeout	Node.js 18	us-central1	267 ms

Fonte: elaborado pelo autor (2023).

Sobre a funcionalidade de cada uma:

- **newUserUpdate**: função que verifica se um usuário está online ou offline por meio de análises no banco de dados de tempo real
- **newRoomCreated**: função configura uma nova sala baseada nas informações do usuário.
- **eventReceivedByHost**: função que gerencia um jogo em andamanto. Para cada atualização um novo evento é gerado.
- **eventReceivedSetTimeout**: função que inicia um temporizador no *RealTime Database* e propaga essa informação para os clientes.

O tempo de resposta obtido não distingue se o tipo de inicialização da função foi um *cold start* ou *hot start*, tratando-se apenas da média em um caso geral para os casos de testes.

O dado mais importante dessa análise trata-se do tempo de resposta de **eventReceivedSetTimeout**, visto que é um gargalo do sistema. Basicamente, após um evento de geração de temporizador é gerado, somente após, em média, ***267 ms*** esse dado é propagado para os clientes. Em outras palavras, no início de uma pergunta com temporizador, os clientes executam, em média, um tempo próprio temporização reduzido de 267 ms.

4.4 Avaliação de requisitos e comparativo com soluções existentes

Uma vez a aplicação construída, é importante verificar se de fato os requisitos de planejamento foram atendidos. Para isso, em comparativo com o que foi mostrado na secção de requisitos funcionais e não-funcionais 3.1.3 e do que foi exposto neste capítulo de resultados, tem-se:

Tabela 5 – Requisitos funcionais e não-funcionais atendidos.

Requisito	Eduquiz
Criação de questionários	Sim
Gerenciamento de questionários	Sim
Participação em questionários	Sim
Feedback em tempo real	Sim
Suporte a vários jogadores	Sim
Criação de cartões de memorização	**Não**
Registro e login do usuário	Sim
Tabela de classificação	Sim

Fonte: elaborado pelo autor (2023).

De acordo com a tabela 5 observa-se que a maioria dos requisitos propostos foi adequadamente atendida, o que é representado pela resposta "Sim" na maioria das linhas. As funcionalidades atendidas evidenciam um bom nível de eficiência na execução do projeto.

Entretanto, a tabela também expõe uma área onde o projeto não conseguiu satisfazer os requisitos iniciais: a "Criação de cartões de memorização". A principal razão para essa discrepância é atribuída à limitação de tempo disponível para o desenvolvimento. A complexidade envolvida na implementação desse recurso em específico exigia um investimento além do escopo de tempo alocado ao projeto, o que culminou na sua não realização.

Dado a discussão acima, cabe pontuar, finalmente, que as funcionalidades implementadas tratam-se apenas de uma versão de protótipo do projeto. Em eventuais incrementos de desenvolvimento, espera-se uma nova avaliação de requisitos e funcionalidades frente aos usuários do "Eduquiz".

Outro ponto a se considerar, ademais, é o posicionamento do "Eduquiz" frente às outras soluções disponíveis no mercado. Fazendo referência ao quadro 1 exposto no capítulo de metodologia, se pode posicionar o "Eduquiz" da seguinte maneira:

Tabela 6 – Comparações entre plataformas existentes e Eduquiz.

Característica	Kahoot!	Squizzy	Quizizz	Classquiz	Eduquiz
Gratuito	Não	Sim	Sim	Sim	**Sim**
De código-aberto	Não	Somente *front-end*	Não	Sim	**Sim**
Apoio da comunidade	Não	Não	Não	Não	**Não**
Funcionaliades extras	Sim	Sim	Não	Não	**Não**
Conceitos de UI/UX	Sim	Sim	Não	Não	**Sim**

Fonte: elaborado pelo autor (2023).

A tabela 6 apresenta um panorama comparativo entre a aplicação "Eduquiz" e algumas das principais soluções similares disponíveis no mercado, utilizando-se de critérios supra-definidos.

A gratuidade do “Eduquiz” é um ponto importante, principalmente no campo educacional, onde muitas vezes os recursos são limitados. Em comparação com o *Kahoot!*, que é uma plataforma paga, o “Eduquiz” apresenta-se como uma opção mais acessível e totalmente sem fins lucrativos. Quando observamos o critério do código-aberto, “Eduquiz” destaca-se ao ser completamente transparente, permitindo que qualquer indivíduo possa contribuir e personalizar a plataforma de acordo com suas necessidades.

No que diz respeito ao Apoio da comunidade, o “Eduquiz” não apresenta apoio no sentido tradicional, mas, sendo um projeto universitário, tem o potencial de envolver a comunidade acadêmica da Universidade Federal do Ceará. Isto poderia se traduzir em contribuições significativas para o projeto a partir do corpo discente e docente.

Quanto às funcionalidades extras, embora o “Eduquiz” não as apresente neste momento, o código foi projetado para ser facilmente expandido. Isto indica a preocupação em construir uma base sólida que permita a implementação de novas funcionalidades no futuro.

Além disso, No quesito Conceitos de UI/UX, “Eduquiz” destaca-se ao lado do *Kahoot!* com um esforço considerável em apresentar uma interface reativa e responsiva aos usuários.

Assim, com base nas análises apresentadas nesta seção, podemos concluir que o “Eduquiz” foi capaz de atender à maioria dos requisitos funcionais e não funcionais estabelecidos inicialmente. Além disso, quando comparado a outras soluções disponíveis no mercado, “Eduquiz” apresenta-se como uma alternativa competitiva. Sua gratuidade, código aberto e preocupação com a experiência do usuário são atributos que o destacam perante outros produtos do mercado. Ainda, o potencial de envolver a comunidade acadêmica da Universidade Federal do Ceará no projeto, traz a possibilidade de expansão e melhoramento da plataforma.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho de conclusão de curso apresentou o desenvolvimento de “Eduquiz”, uma aplicação web de código aberto que serve como um Sistema de Resposta de Estudantes baseado em jogos. Com a educação digital se tornando cada vez mais predominante, o “Eduquiz” surge como uma solução de código-aberto para manter o engajamento em sala de aula.

Durante o desenvolvimento deste projeto, foram abordados diferentes aspectos essenciais para a construção de uma aplicação eficiente e eficaz. Em primeiro lugar, a definição clara dos requisitos funcionais e não-funcionais permitiu que a aplicação atendesse às necessidades específicas de seus usuários. Em segundo lugar, uma experiência de usuário pensada em detalhes em *Figma* assegurou que o sistema fosse não só funcional, mas também intuitivo e esteticamente agradável.

A aplicação da metodologia ágil e das melhores práticas de desenvolvimento de software possibilitou uma implementação eficiente da arquitetura back-end e front-end, fundamentada na utilização dos serviços do *Firebase*. Além disso, o uso de funções *serverless* e um banco de dados *NoSQL* do *Firestore* permitiu a construção de uma aplicação escalável.

A contribuição para a comunidade de código aberto foi também um elemento-chave deste projeto. Ao disponibilizar o código-fonte da aplicação, espera-se que outros desenvolvedores possam aprender e continuar a expandir e melhorar o “Eduquiz”.

No entanto, há uma gama de possibilidades para trabalhos futuros relacionados ao “Eduquiz”. Alguns desses possíveis desenvolvimentos incluem:

- **Personalização Avançada:** Embora o “Eduquiz” já ofereça alguma personalização, poderia haver uma funcionalidade expandida para permitir que os usuários personalizem ainda mais a experiência de aprendizado. Isto poderia incluir a personalização da interface do usuário, a possibilidade de criar caminhos de aprendizado personalizados, entre outros.
- **Integração com Outras Plataformas:** Para aumentar a acessibilidade e a utilidade do “Eduquiz”, poderia ser interessante explorar a integração com outras plataformas educacionais existentes. Isso poderia facilitar o uso do “Eduquiz” em uma variedade de contextos educacionais.
- **Análise de Dados e Aprendizado de Máquina:** Com o uso crescente de tecnologias de análise de dados e aprendizado de máquina na educação, o “Eduquiz” poderia se beneficiar do uso dessas tecnologias para fornecer ideias sobre o progresso dos alunos, personalizar a experiência de aprendizado e muito mais.

- **Recursos de Acessibilidade:** Fazendo jus ao compromisso com a inclusão, seria importante explorar ainda mais os recursos de acessibilidade. Isso poderia incluir a adaptação do design da interface do usuário para ser mais amigável aos deficientes visuais, incluir opções de legendas para deficientes auditivos, entre outros.

Adicionalmente, existem várias áreas de pesquisa que poderiam ser exploradas em trabalhos futuros. Por exemplo, o impacto do uso de aplicações baseadas em jogos no desempenho dos alunos, a eficácia do "Eduquiz" em diferentes ambientes de aprendizado e a relação entre o design da interface do usuário e a experiência do usuário.

REFERÊNCIAS

BAO, W. Covid-19 and online teaching in higher education: A case study of peking university. **Human Behavior and Emerging Technologies**, v. 2, n. 2, p. 113–115, 2020. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/hbe2.191>.

BLINOWSKI, G.; OJDOWSKA, A.; PRZYBYŁEK, A. Monolithic vs. microservice architecture: A performance and scalability evaluation. **IEEE Access**, v. 10, p. 20357–20374, 2022.

BOGOST, I. **Gamification Is Bullshit**. [S.l.]: Ian Bogost Blog, 2011.

CARINE GEORGE D. KUH, S. P. K. R. Student engagement and student learning: Testing the linkages. Res High Educ, 2006.

DENNY, P. The effect of virtual achievements on student engagement. **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**, ACM, p. 763–772, 2013.

DETERDING, S. Gamification: Toward a definition. 2011.

DRAGONI, N.; GIALLORENZO, S.; LLUCH-LAFUENTE, A.; MAZZARA, M.; MONTESI, F.; MUSTAFIN, R.; SAFINA, L. Microservices: yesterday, today, and tomorrow. In: ____. [S.l.: s.n.], 2017.

FETTE, I.; MELNIKOV, A. The websocket protocol. **Internet proposed standard RFC**, v. 6455, 2011.

FIELDING, R. T. Architectural styles and the design of network-based software architectures. In: . [S.l.: s.n.], 2000.

GEE, J. P. **What video games have to teach us about learning and literacy**. [S.l.]: Palgrave Macmillan, 2003.

GONçALVES, M. M. **Arquitetura de microsserviços**. 2020. Disponível em: <https://medium.com/@marcelomg21/arquitetura-de-microsservi%C3%A7os-bc38d03fbf64>. Acesso em: 30 jun. 2023.

HAMARI, J.; KOIVISTO, J.; SARSA, H. Does gamification work? – a literature review of empirical studies on gamification. **HICSS '14: Proceedings of the 47th Hawaii International Conference on System Sciences**, 2014.

HANUS, M. D.; FOX, J. Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. **Computers Education**, v. 80, p. 152 – 161, 2015. ISSN 0360-1315. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360131514002000>.

HUNG, H.-C.; HUANG, I.; HWANG, G.-J. Effects of digital game-based learning on students' self-efficacy, motivation, anxiety, and achievements in learning mathematics. **Journal of Computers in Education**, Springer, v. 1, n. 2-3, p. 151–166, 2014.

KAPP, K. M. **The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education**. [S.l.]: Pfeiffer, 2012.

LAWSON, B.; SHARP, R. **Introducing HTML5**. [S.l.]: New Riders, 2011.

MOLONEY, K. Taking the game out of gamification. In: **11th International Conference on Knowledge, Culture and Change in Organisations**. [S.l.: s.n.], 2011.

NEWMAN, S. Building microservices: designing fine-grained systems. O'Reilly Media, Inc., 2015.

NICHOLSON, S. **A RECIPE for Meaningful Gamification**. [S.l.]: Learning, Education and Games, 2015.

NORMAN, D. A. **The design of everyday things**. [New York]: Basic Books, 2002. ISBN 0465067107 9780465067107. Disponível em: <http://www.amazon.de/The-Design-Everyday-Things-Norman/dp/0465067107/ref=wl_it_dp_o_pC_S_nC?ie=UTF8&colid=151193SNGKJT9&coliid=I262V9ZRW8HR2C>.

OWEN, H.; LICORISH, S. Game-based student response system: The effectiveness of kahoot! on junior and senior information science students' learning. **Journal of Information Technology Education: Research**, v. 19, p. 511–553, 01 2020.

RANIERI, M.; RAFFAGHELLI, J. E.; BRUNI, I. Game-based student response system: Revisiting its potentials and criticalities in large-size classes. **Active Learning in Higher Education**, v. 22, n. 2, p. 129–142, 2021. Disponível em: <https://doi.org/10.1177/1469787418812667>.

ROBERTS, R. The game is getting old: Keeping video game collections. In: **Proceedings of the ACM 2015 Conference on Computer Supported Cooperative Work and Social Computing**. [S.l.: s.n.], 2015. p. 1200–1208.

SHULL, F.; SINGER, J.; SJØBERG, D. I. **Software engineering process: principles and goals**. [S.l.]: John Wiley & Sons, Inc., 2002.

TULLIS, T.; ALBERT, B. **Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics (Interactive Technologies): Collecting, Analyzing, and Presenting ... Kaufmann Series in Interactive Technologies)**. Morgan Kaufmann, 2008. ISBN 0123735580. Disponível em: <http://www.amazon.de/Measuring-User-Experience-Interactive-Technologies/dp/0123735580%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0123735580>.

WANG, A. I.; TAHIR, R. The effect of using kahoot! for learning – a literature review. **Comput. Educ.**, Elsevier Science Ltd., GBR, v. 149, n. C, may 2020. ISSN 0360-1315. Disponível em: <https://doi.org/10.1016/j.compedu.2020.103818>.

YOU, E. **Vue.js: The Progressive JavaScript Framework**. 2020. Acessado em: 1 Julho 2023. Disponível em: <https://vuejs.org/>.

ZICHERMANN, G.; CUNNINGHAM, C. **Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps**. [s.n.], 2011. Disponível em: <http://shop.oreilly.com/product/0636920014614.do>.

APÊNDICE A – QUESTIONÁRIO DE LEVANTAMENTO DE REQUISITOS

Questões do tipo **"seria útil"**, com respostas variando de **1 - Não faz diferença** a **5 - Imprescindível** :

Questão 1. Criar modelos de questões personalizados que possam ser reutilizados?

Questão 2. Atribuir questionários e avaliações a grupos específicos de alunos ou turmas? Ex: questionários aplicados somente para conjunto de usuários.

Questão 3. Mostrar resultado parcial durante a partida?

Questão 4. Possuir recursos que permitam acompanhar o progresso dos alunos e identificar áreas em que eles estão com dificuldades ou se destacando?

Questão 5. Incluir conteúdo multimídia, como imagens ou vídeos, nos seus quizzes e avaliações?

Questão 6. Colaborar com outros professores para criar quizzes?

Questão 7. Ter distintivos ou recompensas por completar determinadas tarefas ou atingir certas pontuações?

Questão 8. Ter a possibilidade de configurar o tempo de reposta de cada pergunta?

Questão 9. Ter uma aplicação responsiva adaptada para telas menores como celular e tablet?

Questão 10. Incluir desafios ou exercícios de programação nos seus quizzes, permitindo que os alunos pratiquem e apliquem conceitos de programação em tempo real?

Questão 11. Avaliar automaticamente as tarefas e projetos de programação, usando ferramentas de análise de código ou frameworks de testes automatizados, como Sonarcloud, eslint, etc.?

Questão 12. Criar quizzes com diferentes tipos de perguntas de programação, como perguntas de múltipla escolha, resposta curta ou preencher lacunas?

Questão 13. Personalizar a linguagem ou o framework de programação usado nos seus quizzes, de acordo com o seu currículo ou objetivos de aprendizagem específicos?

Questão 14. Incluir trechos de código ou programas de exemplo nos seus quizzes, permitindo que os alunos analisem e modifiquem o código existente?