UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO GUEDES BRAGA

LEIA RAPIDINHO: UMA APLICAÇÃO PARA EXERCITAR A LEITURA DE PALAVRAS DA LÍNGUA PORTUGUESA

CAMPINA GRANDE - PB

2021

PEDRO GUEDES BRAGA

LEIA RAPIDINHO: UMA APLICAÇÃO PARA EXERCITAR A LEITURA DE PALAVRAS DA LÍNGUA PORTUGUESA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. José Antão Beltrão Moura.

CAMPINA GRANDE - PB

2021

B813l Braga, Pedro Guedes.
Leia rapidinho: uma aplicação para exercitar a leitura de palavras da língua portuguesa. / Pedro Guedes Braga. – 2021.

12 f.

Orientador: Prof. Dr. José Antão Beltrão Moura.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Leitura. 2. Aplicativo de leitura. 3. Aplicação web gamificada. 4. Reconhecimento de voz. 5. Application programming interface – voice. 6. Gamificação. 7. Jogo educativo. 8. Alfabetização. 9. I. Moura, José Antão Beltrão. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

PEDRO GUEDES BRAGA

LEIA RAPIDINHO: UMA APLICAÇÃO PARA EXERCITAR A LEITURA DE PALAVRAS DA LÍNGUA PORTUGUESA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. José Antão Beltrão Moura

Orientador – UASC/CEEI/UFCG

Professor Dr. Adalberto Cajueiro de Farias

Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 20 de OUT de 2021.

CAMPINA GRANDE - PB

RESUMO (ABSTRACT)

Fluent reading is an important characteristic that must be acquired during the literacy process. Different indicators show worrying problems in the literacy level of Brazilian students, including the characteristic of being a fluent reader. This work reports the development of a gamified web application that seeks to improve, through the use of a voice recognition API available in some browsers, the word reading ability of students in the literacy process. At the end of development, a survey was conducted and the results reveal that the application can be useful as a didactic tool at the level of basic education.

Leia Rapidinho: Uma aplicação para exercitar a leitura de palavras da língua portuguesa

Trabalho de Conclusão de Curso

Pedro Guedes Braga
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
pedro.braga@ccc.ufcg.edu.br

José Antão Beltrão Moura
(orientador)
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
antao@computacao.ufcg.edu.br

RESUMO

A leitura fluente é uma característica importante que deve ser adquirida durante o processo de alfabetização. Diferentes indicadores evidenciam problemas preocupantes no nível de alfabetização dos estudantes brasileiros, incluindo a característica de ser um leitor fluente. Este trabalho relata o desenvolvimento de uma aplicação web gamificada que busca aprimorar, a partir do uso de uma API de reconhecimento de voz disponível em alguns *browsers*, a habilidade de leitura de palavras nos estudantes em processo de alfabetização. Ao final do desenvolvimento, foi conduzida uma pesquisa e os resultados revelam que a aplicação pode ser útil como instrumento didático na educação básica.

PALAVRAS-CHAVE

Reconhecimento, voz, alfabetização, Javascript.

LINKS ÚTEIS

https://drive.google.com/file/d/1OXNf8j5o5qn0A-EpjJrlWz4SyB3Azkrs/view?usp=sharing **(Termo de consentimento Livre e Esclarecido)**

https://leia-rapidinho.herokuapp.com/ **(Endereço da aplicação na web)**

https://github.com/PedroGuedesBraga/leia-rapidinho-frontend **(Código fonte - Aplicação *front-end*)**

https://github.com/PedroGuedesBraga/leia-rapidinho-backend **(Código fonte - Aplicação *back-end*)**

Resultados obtidos no questionário de forma detalhada:

https://github.com/PedroGuedesBraga/leia-rapidinho-backend/tree/master/research_results **(Resultados detalhados - Questionário)**

1. INTRODUÇÃO

No processo de alfabetização, a fluência na leitura é uma característica que se refere à capacidade de reproduzir uma sequência de palavras oralmente e com facilidade [1]. Atualmente, ser um leitor fluente é imprescindível, visto que a leitura é uma atividade comum no cotidiano. O fato do nível de alfabetização ter influência na qualidade de vida do indivíduo [2] traz um problema grave para o Brasil, que sofre com avaliações ruins em exames nacionais e internacionais de desempenho escolar.

A Avaliação Nacional de Alfabetização (ANA), realizada em 2016 com mais de dois milhões de estudantes, declarou em seus resultados que 54,73% dos alunos avaliados com idade acima de 8 anos possuem níveis insuficientes de leitura [3].

O PISA (*Programme for International Student Assessment*), realizado em 2018, revelou que apenas cerca de 50% dos estudantes brasileiros atingiram nível 2 ou acima no letramento em leitura e, em contraste, 77,4% dos estudantes dos países da OCDE atingiram esse nível [4], que é considerado o mínimo que todos os estudantes devem adquirir até o final do ensino médio.

Para desenvolver a fluência na leitura, é comum que professores alfabetizadores proponham atividades fazendo o uso de recursos que oferecem pouca ou nenhuma interação com o aluno, sendo portanto pouco atrativos. Muitas vezes são aplicados exercícios em folha de papel para o público infantil (mesmo que a folha seja representada por meios digitais).

Diante da problemática exposta, o presente trabalho relata o desenvolvimento do **Leia Rapidinho**, uma aplicação *web* que busca evoluir a habilidade de fluência na leitura em alunos no processo de alfabetização. Para isso, palavras da língua portuguesa são apresentadas para o usuário e este tem o desafio de reproduzi-las oralmente, utilizando recursos de gamificação como placares e restrição de tempo. É feito o uso da *Web Speech* API [5], uma interface que permite a aplicações que executam em alguns *browsers* realizarem o reconhecimento da voz do usuário. A aplicação pode ser acessada em diferentes dispositivos e é construída com base em uma arquitetura cliente-servidor, utilizando a "pilha" de tecnologias MERN [6].

Aplicações que buscam exercitar a fluência na leitura são comuns e facilmente encontradas através de pesquisas nos motores de busca, mas se tornam escassas se filtrarmos pelas que usam reconhecimento de voz. Uma aplicação que faz uso de reconhecimento de voz é o *Read Along* [7], porém, além de estar disponível apenas para dispositivos com sistema operacional *Android*, ela tem foco na leitura de textos e não de palavras de maneira individual. A leitura fluente de textos tem como pré-requisito a leitura fluente de palavras [1].

Um benefício apontado por André Coy, em seu trabalho [8], é que as aplicações que usam reconhecimento de voz automático permitem que as pessoas aprendam no seu próprio ritmo, de acordo com o seu nível de proficiência e sem impactar outras pessoas com um níveis de proficiência diferentes. Um problema comum enfrentado por professores alfabetizadores atualmente é lidar com diferentes níveis de leitura dos alunos "ao mesmo tempo".

Além de relatar o processo de desenvolvimento do Leia Rapidinho, este trabalho expõe os seus resultados a partir de uma

pesquisa que envolveu profissionais com experiência na educação básica. Os resultados evidenciam que a aplicação pode se posicionar como mais um canal de aprendizado para pessoas em processo de alfabetização.

Na seção 2, é exposto uma visão geral das etapas do processo de desenvolvimento da aplicação. As seções 3, 4, 5 e 6 abordam a realização e o resultado dessas etapas. O documento finaliza na seção 7, que faz uma conclusão geral sobre a viabilidade da aplicação, além de considerações sobre desafios, limitações e trabalhos futuros.

2. METODOLOGIA

O processo de concepção da aplicação compreendeu quatro etapas: **coleta de requisitos, desenvolvimento, implantação** e **avaliação** da aplicação.

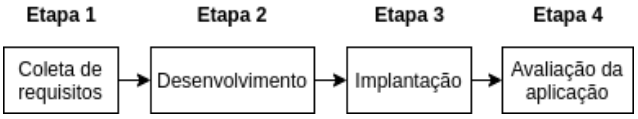

Figura 01: Etapas para realização do trabalho.

Na etapa 1, com a participação de profissionais da educação básica, os requisitos da aplicação foram elencados fazendo o uso de duas práticas comuns: entrevista e *brainstorming*.

A etapa 2 consistiu na idealização da arquitetura e desenvolvimento da aplicação com base nos requisitos levantados e na própria arquitetura. Foi feito o uso da metodologia ágil *Kanban* [9] para dar suporte à organização e priorização das atividades desenvolvidas. O resultado dessa etapa consistiu em uma aplicação funcionando em ambiente de desenvolvimento.

Na etapa 3, uma primeira versão da aplicação foi disponibilizada utilizando uma plataforma que fornece recursos de forma gratuita para execução de aplicações *web.*

Na última etapa, foi aplicado um questionário para um grupo de profissionais da educação básica, com o objetivo de entender se a aplicação tem nível satisfatório de usabilidade e se atende aos objetivos propostos.

3. COLETA DE REQUISITOS

O levantamento de requisitos contou com a participação de duas profissionais da educação básica, que foram informadas previamente sobre o tema e submetidas à assinatura de um Termo de Consentimento Livre e Esclarecido, cujo modelo está na seção de Links Úteis deste documento. Inicialmente foi realizada uma entrevista em que se buscou respostas para duas questões, sendo a primeira (Q1) "Quais exercícios são aplicados atualmente para desenvolver a fluência dos alunos na leitura de palavras?" e a segunda, (Q2) "Quais as dificuldades enfrentadas ao aplicar os referidos exercícios?".

Como resposta para a Q1, foi relatado que é feito o uso de listas de palavras em fichas impressas para os alunos iniciantes e textos curtos para os alunos mais avançados. O grau de complexidade das palavras apresentadas nas fichas varia de acordo com o nível de fluência do aluno. Já para a Q2, foi relatado que há um desafio em atender muitos alunos e coletar o *feedback* do nível de leitura de todos. Ainda como resposta para a Q2, também há o desafio de reter a atenção do aluno na tarefa.

Em seguida, por meio de um *brainstorming*, foi possível imaginar o que a aplicação deveria ter para exercitar e avaliar a fluência na leitura de palavras, preenchendo as lacunas expostas anteriormente. Dessa forma, elencou-se os seguintes requisitos:

REQ 01 - O usuário deve ser capaz de ter uma conta para usar a aplicação. Cada conta de usuário tem um progresso associado.

REQ 02 - A aplicação deve exercitar a fluência na leitura por meio de palavras expostas na tela que serão lidas pelo usuário em voz alta. Sendo assim, deve ser feito o reconhecimento da voz do usuário. A aplicação deve interagir com o usuário a cada leitura, apontando se foi correta ou incorreta.

REQ 03 - O usuário deve conseguir desbloquear conquistas na medida em que obtém progresso na aplicação.

REQ 04 - O desempenho do usuário poderá ser representado por meio de um gráfico, que poderá ser acompanhado pelos responsáveis do usuário (por exemplo, pais ou professores).

REQ 05 - A aplicação deve ter um *design* atrativo ao público infantil, que é o público alvo.

REQ 06 - A aplicação deve ter instruções de uso, dirigida aos responsáveis pelo usuário, que podem assumir o papel de guia.

4. MÉTODO DE DESENVOLVIMENTO E ARQUITETURA DA SOLUÇÃO

4.1 Método de desenvolvimento

Na etapa de desenvolvimento da aplicação, optou-se pelo uso da metodologia ágil Kanban, pois tem uma baixa complexidade para ser praticada e é compatível com o cenário no qual há apenas uma pessoa desenvolvendo a solução. Foi utilizada a ferramenta *Trello* [10], que disponibiliza um quadro Kanban de forma gratuita e permite customizações em relação à criação de colunas e tarefas.

No que se refere à composição do quadro Kanban, as tarefas (representadas por cartões) foram geradas a partir dos requisitos coletados (cada requisito poderia gerar uma ou mais tarefas) e ordenadas verticalmente de acordo com o grau de prioridade (quanto mais elevada era a tarefa, maior era sua prioridade). Foram definidas as colunas **a desenvolver**, **em refino**, **em desenvolvimento** (com limite máximo de 2 tarefas para essa coluna) e **em revisão**, que nessa ordem, mapeiam no quadro o fluxo de trabalho. Para concluir todas as tarefas do quadro foi necessário um período de 64 dias.

4.2 Arquitetura da solução

O Leia Rapidinho é composto por três elementos principais: uma aplicação cliente (aplicação *front-end*), uma aplicação que executa em um servidor (aplicação *back-end*) e um banco de dados não relacional. Além desses, existem dois serviços de terceiros que se comunicam com a aplicação: um serviço de envio de e-mail e um serviço de reconhecimento de voz. As próximas subseções irão abordar em detalhes os componentes principais da aplicação.

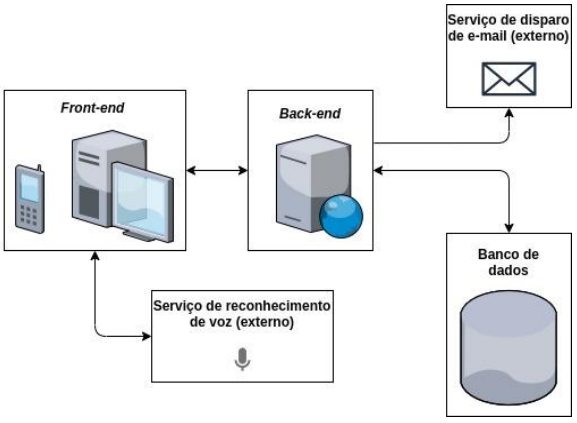

Figura 02: Componentes da aplicação e seus relacionamentos

4.2.1 Aplicação *front-end*

A aplicação *front-end* executa no *browser* e é responsável por fornecer uma interface de interação para o usuário fazer o uso do sistema. Além disso, se comunica com a aplicação *back-end* por meio de requisições HTTP e com um serviço de reconhecimento de voz.

4.2.1.1 Tecnologias utilizadas no *front-end*

O *front-end* da aplicação foi construído utilizando Javascript como linguagem de programação, com um suporte essencial das bibliotecas *React* [11], *Redux* [12], *Semantic UI React* [13] e *react-speech-recognition* [14].

O *React* é uma popular biblioteca Javascript que utiliza o conceito de componentes: partes independentes e reutilizáveis que, se organizando de forma hierárquica, compõem a interface de usuário (um formulário com botões e caixas de texto pode ser um exemplo de componente). Componentes podem ter um estado próprio (conjunto de dados) e seu comportamento pode variar de acordo com a disposição desse estado. O uso dessa biblioteca permite o desenvolvimento rápido (dado a característica de reuso) de uma aplicação modularizada e escalável.

O *Redux* é uma biblioteca que tem inspiração na arquitetura Flux [15] e permite gerenciar um estado global e centralizado da aplicação, que pode ser disponibilizado para qualquer componente que compõe a interface de usuário. O uso dessa biblioteca tem o objetivo diminuir a complexidade da aplicação, tornando mais simples o compartilhamento de dados entre vários componentes, especialmente quando estão pouco relacionados na hierarquia de componentes. A figura 3 apresenta um diagrama que detalha o fluxo de dados com o uso do Redux: os componentes que compõem a interface de usuário podem, por meio de uma função conhecida como *dispatch*, disparar *actions* (ações) para uma entidade denominada *store*. A *store* faz o uso de funções puras, denominadas *reducers*, que processam as *actions* e retornam um novo o estado global da aplicação. Por fim, a interface de usuário é atualizada com o novo estado global da aplicação.

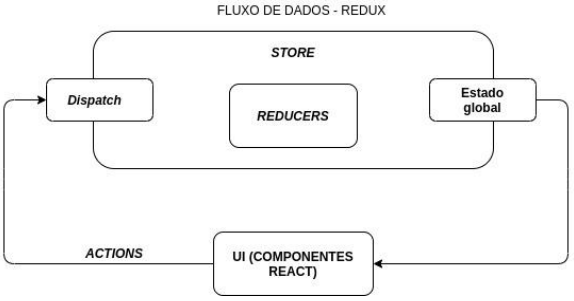

Figura 03: Fluxo de atualização do estado global da aplicação com Redux

O *Semantic UI React* é uma biblioteca que fornece componentes já estilizados e de fácil importação na aplicação, como botões e campos de formulários. O uso dessa biblioteca permitiu uma maior agilidade, diminuindo o esforço no desenvolvimento de um design atrativo.

A biblioteca *react-speech-recognition* permite realizar a conversão de fala para texto em componentes React e seu uso foi motivado por sua boa documentação. Internamente essa biblioteca faz o uso da *Web Speech* API, uma interface implementada por alguns *browsers* que provê a conversão de fala em texto. A estratégia usada para converter voz em texto pode variar de acordo com o *browser* que executa a aplicação, pois cada um pode implementar a interface de maneira singular. Por exemplo, se a aplicação está executando no *Google Chrome* será necessário uma conexão à internet, pois o processamento é feito de forma remota, a partir do envio do áudio capturado para um serviço externo do Google.

4.2.1.2 Estrutura do *front-end*

A aplicação *front-end* foi estruturada com inspiração no popular conceito de componentes de apresentação e *containers* [16]. A figura 4 expõe a estrutura de diretórios utilizada. O código fonte da aplicação está presente no diretório *src* e os subdiretórios estão organizados da seguinte maneira:

- ***Actions***: Os arquivos deste diretório contém funções que retornam outra função responsável por despachar uma ação para a *store*, podendo então alterar o estado global da aplicação.
- ***Api***: Disponibiliza uma interface para a conexão aos *endpoints* do *backend* da aplicação.
- ***Components***: Reúne os componentes reutilizáveis da aplicação (como por exemplo um cabeçalho de menu). Esses componentes não estão diretamente conectados ao estado global da aplicação.
- ***Containers***: Armazena os componentes que estão diretamente conectados ao estado global da aplicação e podem gerenciá-lo. Cada um desses componentes representa uma página que pode ser acessada por um usuário da aplicação e são construídos a partir de componentes reutilizáveis.
- ***Reducers***: Os arquivos desse diretório disponibilizam funções que constroem um novo estado global da aplicação a partir de uma determinada *action* disparada.
- ***Utils***: Os arquivos desse diretório armazenam funções utilitárias.

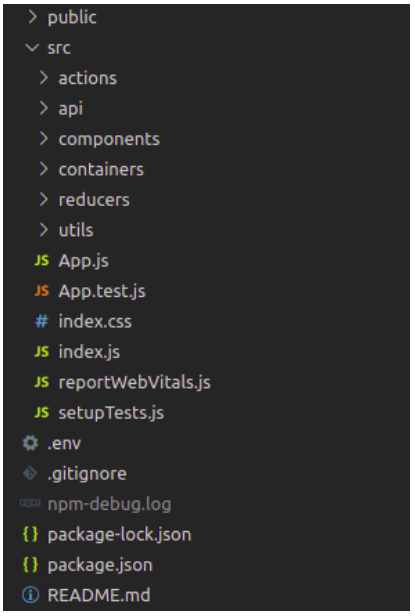

Figura 04: Estrutura de diretórios da aplicação *front-end*

4.2.2 Aplicação *back-end*

A aplicação *back-end* é executada em um servidor e além de armazenar as regras de negócio do sistema, tem a responsabilidade de processar e responder requisições HTTP oriundas da aplicação *front-end*, realizar conexões ao banco de dados e interagir com o serviço externo de envio de e-mail.

4.2.2.1 Tecnologias utilizadas no *back-end*

Assim como no *front-end*, o código fonte da aplicação *back-end* foi construído utilizando a linguagem *Javascript*, sendo isso possível por mérito do *Node.js* [17], uma plataforma que permite

interpretar código *Javascript* sem a necessidade de um *browser*. Os principais *frameworks* e bibliotecas que dão suporte à aplicação *back-end* são: *express* [18], *mongoose* [19], *jsonwebtoken* [20] e *bcrypt* [22].

O *express* é um *framework* popular que permite desenvolver e configurar aplicações *web* de forma minimalista (o *framework* em si possui apenas o essencial) e não opinativa (o *framework* não restringe a aplicação à uma estrutura ideal). Seu uso se deu devido à experiência prévia do desenvolvedor. Além disso, possui uma documentação completa.

O *mongoose* [19] é uma biblioteca ODM (*Object-Document Model*) e é responsável por mapear entidades presentes no banco de dados em objetos da aplicação (no caso, mapeia registros do banco de dados em objetos *Javascript*). Seu uso se deu pela facilidade de configuração e por prover uma interface para manipular os dados armazenados de maneira simples (permite realizar consultas, gravações, atualizações, dentre outras operações).

A biblioteca *jsonwebtoken* [20] é capaz de dar suporte à criação de *tokens JWT* [21]. Esses tokens fazem uso de assinaturas digitais e são utilizados para garantir autenticação ao transmitir informação por meio de requisições entre as aplicações *back-end* e *front-end*.

O *bcrypt* [22] é uma biblioteca que permite fazer o uso de algoritmos de *hash* para aumentar o nível de segurança ao manipular dados sensíveis. Na aplicação em questão, senhas de usuário são aplicadas a algoritmos de *hash* (que são determinísticos e unidirecionais) e o resultado retornado é salvo no banco de dados. Assim, caso alguma entidade adquira acesso indevido aos dados armazenados, haverá uma segunda linha de defesa com o objetivo de preservar dados sensíveis do usuário.

4.2.2.2 Estrutura do *back-end*

A aplicação *back-end* tem inspiração no princípio da separação de preocupações. Foram definidas camadas, cada uma com responsabilidade bem definida para processamento da requisição oriunda da aplicação *front-end*. Os diretórios mais importantes da aplicação são listados a seguir e podem ser visualizados na figura 05.

- ***Routes*:** O conteúdo desse diretório define todos os *endpoints* que uma aplicação cliente pode fazer uso para interagir com a aplicação *backend*, por meio de requisições HTTP.
- ***Controllers:*** Arquivos desse diretório possuem métodos que são associados a algum *endpoint* específico da aplicação. Esses métodos são responsáveis por lidar com as requisições HTTP realizando validações, repassando informações importantes para a camada de serviço e respondendo às próprias requisições (definindo atributos como código de status e conteúdo a ser retornado no corpo da resposta).
- ***Middlewares:*** Diretório que possui funções auxiliares que podem ser executadas antes do fluxo atingir os métodos dos *Controllers* ou após a execução destes.
- ***Services*:** Responsável por possuir métodos que processam as informações enviadas pela camada de *controllers* com base nas regras de negócio da aplicação. Além disso, faz integrações com serviços externos e interações com o banco de dados.
- ***Models:*** Diretório que reúne o mapeamento de entidades do banco de dados em objetos *Javascript*, fornecendo uma interface para manipulação dessas entidades.
- ***Logs:*** Reúne arquivos que registram eventos importantes que ocorrem na execução da aplicação.

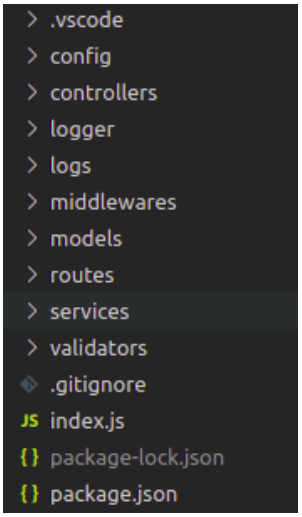

Figura 05: Estrutura de diretórios da aplicação *back-end*

4.2.3 Banco de dados

Para a persistência de dados da aplicação, optou-se pelo uso do *MongoDB* [23], um software *open-source* de banco de dados orientado a documentos. Seu uso se deu devido a três fatores principais: baixa complexidade de instalação e uso, experiência prévia do desenvolvedor e o fato de possuir boa documentação.

O *MongoDB* armazena cada registro de dados em estrutura denominada documento, que é armazenada internamente no formato BSON (uma representação binária do popular formato JSON, estendida para suportar tipos de dados mais avançados, como datas). Documentos são reunidos em estruturas denominadas coleções e uma ou mais coleções podem estar presentes em um banco de dados.

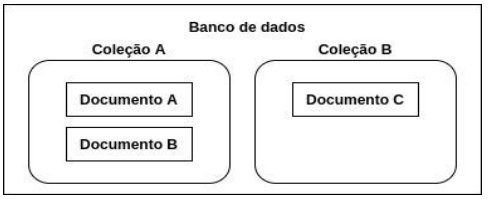

Figura 06: Estruturação dos dados no MongoDB

5. IMPLANTAÇÃO E FUNCIONAMENTO DA APLICAÇÃO

5.1 Implantação

Para disponibilizar uma versão inicial do Leia Rapidinho ao público alvo, foi feito o uso do *Heroku* [24], um serviço de computação em nuvem que usa o modelo *PaaS* (*Plataform as a Service*), disponibilizando recursos para a execução da aplicação.

Além de ser gratuito, o uso do Heroku permitiu maior agilidade no processo de implantação, pois a preocupação em relação à aspectos de instalação e manutenção de componentes de hardware e software necessários para a execução da aplicação são de propriedade do provedor do serviço.

5.2 Funcionamento da aplicação

Nas subseções seguintes, serão expostas as funcionalidades do Leia Rapidinho, que pode ser acessada a partir do endereço na seção de Links Úteis deste documento. Com o objetivo de reter a atenção do usuário, é possível notar a presença

de mecânicas de jogos na aplicação, como por exemplo a visualização de progresso e restrições de tempo.

5.2.1 *Login* e Registro

O Leia Rapidinho impõe que seus usuários usem credenciais registradas previamente para acessar as funcionalidades do sistema. Assim, ao acessar a página inicial, o usuário se depara com um formulário de *login* e nele é necessário introduzir um endereço de e-mail e senha como credenciais.

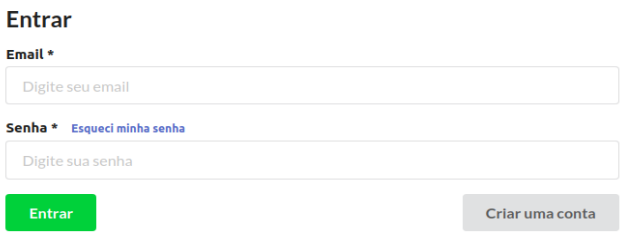

Figura 07: Formulário presente na página de *login*

Ao introduzir credenciais válidas no processo de *login*, um token de acesso JWT é retornado pela aplicação *back-end* para a aplicação *front-end*. Essa última armazena o token em memória e o envia quando precisa requisitar recursos da aplicação *back-end* que demandam autenticação.

Caso o usuário não tenha uma conta registrada no sistema, é fornecido a opção de criação de conta por meio de uma pagina de registro. Nela, é necessário preencher um formulário com nome, sobrenome, endereço de email, senha e confirmação de senha. Após submeter o formulário, um e-mail é disparado para o endereço informado por meio de um serviço externo, a fim de confirmar que o usuário tem a posse do mesmo.

Figura 08: Formulário presente na página de registro

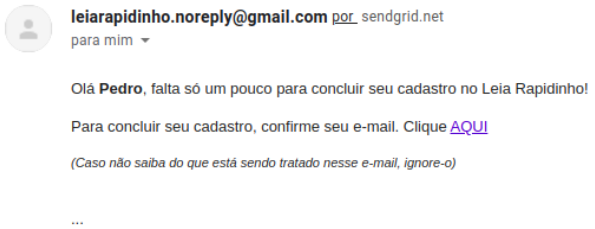

Figura 09: E-mail de confirmação enviado na etapa de registro

5.2.2 Menu principal

Logo após inserir credenciais válidas, o usuário é redirecionado para uma página que exibe um menu principal. Nela, é possível acessar as instruções direcionadas aos responsáveis do usuário e iniciar uma nova rodada. Com o objetivo de facilitar o entendimento para usuários que não são fluentes na leitura, todos os botões tem ícones no seu conteúdo.

Figura 10: Página de menu principal.

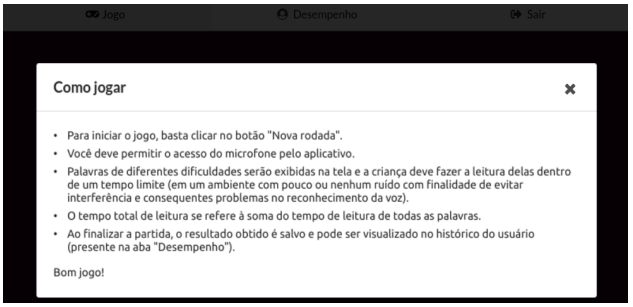

Figura 11: Instruções para uso da aplicação

5.2.3 Execução de uma rodada

Ao iniciar uma nova rodada, é consultado na base de dados uma lista de palavras aleatórias e em seguida, uma contagem regressiva de três segundos é exibida na tela para que o usuário se prepare. Ao término da contagem, a primeira palavra da lista é exibida e o usuário deve realizar a sua leitura (nesse momento o *browser* pode solicitar a ativação do microfone do dispositivo).

Ao ler uma palavra de forma incorreta, ela é preenchida na cor vermelha por um curto período de tempo. Em contrapartida, ao ler uma palavra de forma correta, uma animação com confetes é exibida na tela e a palavra seguinte da lista é exibida para ser lida.

Figura 11: Palavra lida incorretamente

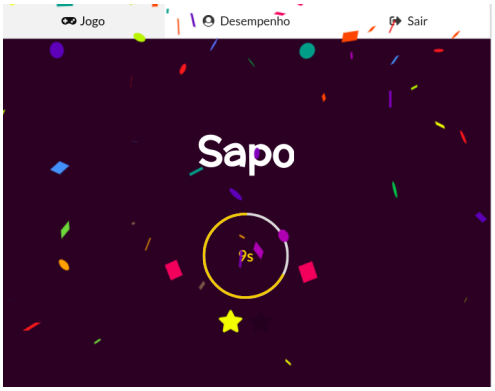

Figura 12: Palavra lida corretamente

Cada palavra foi inserida previamente no banco de dados da aplicação com uma estimativa de dificuldade e tempo de leitura. O tempo total de uma rodada é calculado a partir da soma do tempo de leitura de todas as palavras da lista recuperada e é exibido em forma de contagem regressiva. Uma rodada acaba em dois cenários: caso o tempo total acabe ou caso o usuário leia todas as palavras. Quando uma rodada atinge seu término, os dados são salvos no banco de dados da aplicação e ícones em forma de estrela são exibidos, sendo um para cada palavra lida.

Figura 13: Tela exibida ao término de uma rodada.

5.2.4 Visualização de desempenho do usuário

Ao acessar uma página de desempenho, o usuário é capaz de visualizar o total de palavras lidas em cada nível de dificuldade. Emblemas podem ser desbloqueados pelo usuário na medida em que se obtém progresso na aplicação. Além disso, é possível visualizar um gráfico que expõe a quantidade de palavras lidas nas últimas 15 partidas.

As informações na página de desempenho são geradas a partir dos dados salvos ao término de cada rodada e permitem que os responsáveis pelo usuário verifiquem se está havendo algum progresso em relação ao nível de leitura.

Figura 14: Página de desempenho

6. AVALIAÇÃO

6.1 Metodologia utilizada

Com o intuito de descobrir aspectos que podem ser melhorados ou acrescentados e entender o grau de satisfação dos usuários em relação a usabilidade e objetivos do sistema, foi conduzida uma pesquisa com a participação de 4 professores que possuíam entre 2 e 24 anos de experiência no nível básico de educação da rede pública de ensino. Como instrumento de coleta de informações, foi utilizado um questionário disponibilizado de forma online. Ao todo, 13 itens (detalhados na tabela 1) fizeram parte do questionário, que teve o uso da aplicação como pré-requisito para ser respondido. Nos 12 primeiros itens, o respondente foi encarregado de emitir um grau de concordância para uma afirmativa. Já o último item era opcional e poderia ser respondido de forma aberta.

Tabela 01: Itens do questionário de avaliação da aplicação

Nº	ITEM
1	Eu indicaria o Leia Rapidinho com frequência para alunos em processo de alfabetização.
2	Eu acho o Leia Rapidinho desnecessariamente complexo.
3	Eu acho fácil de usar o Leia Rapidinho.
4	Eu acho que é necessário uma pessoa com conhecimentos técnicos para que seja feito o uso do Leia Rapidinho.
5	Eu acho que as diferentes funções do Leia Rapidinho estão muito bem integradas.
6	Eu acho que o Leia Rapidinho apresenta muita inconsistência no contexto de usabilidade (Padrões de cores, fontes usadas nos textos ou comportamento de botões mudando drasticamente de acordo com a funcionalidade em uso são exemplos de inconsistência).
7	Eu acho que os alunos junto com seus responsáveis aprenderão a usar o Leia Rapidinho rapidamente.
8	Eu acho o Leia Rapidinho confuso de usar.

9	Eu me senti confiante usando o Leia Rapidinho.
10	Eu precisei aprender várias coisas novas antes de usar o Leia Rapidinho.
11	O Leia Rapidinho tem a capacidade de exercitar a leitura de palavras nos alunos em processo de alfabetização.
12	O Leia Rapidinho tem a capacidade de atrair crianças em processo de alfabetização.
13	Sugestões para melhoria do Leia Rapidinho (opcional).

Frequentemente é necessário comparar aplicações, mesmo que pertençam a contextos diferentes, tomando a usabilidade como referência. Dessa forma, os 10 primeiros itens do questionário foram construídos a partir de uma adaptação dos itens que compõem um questionário base, usado para medir a escala de usabilidade SUS (*System Usability Scale*) [25] de um sistema. Tanto a escala SUS como o questionário base que a mede foram idealizados em 1986 por John Brooke e os itens a serem respondidos tem um caráter generalista, o que permite o uso em aplicações de diferentes contextos. A medição da escala SUS é um valor entre 0 e 100 e pode ser encontrada após uma manipulação matemática envolvendo as respostas obtidas. Os itens são compostos por afirmações e os respondentes devem escolher um valor entre 1 (discordância total) e 5 (concordância total).

Com o objetivo de entender se a aplicação atende aos objetivos propostos, foram idealizados os itens de número 11 e 12 do questionário, que como nos 10 primeiros itens, foram respondidos com um valor entre 1 e 5.

O último item do questionário é de caráter opcional e permitiu que os respondentes, com suas próprias palavras, propusessem melhorias para o sistema.

6.2 Análise de resultados

Para a análise dos resultados obtidos, os itens respondidos no questionário foram **separados em 3 grupos**. Utilizou-se a linguagem de programação R [26] em conjunto com a ferramenta RStudio [27] para fazer a análise estatística dos resultados. Todas as respostas obtidas podem ser encontradas na seção de Links Úteis deste documento.

O **primeiro grupo** analisado compreende os 10 primeiros itens do questionário, que avaliam a usabilidade do sistema. O valor médio das respostas de cada um desses itens foram calculados e são expostos no gráfico de barras da figura 14.

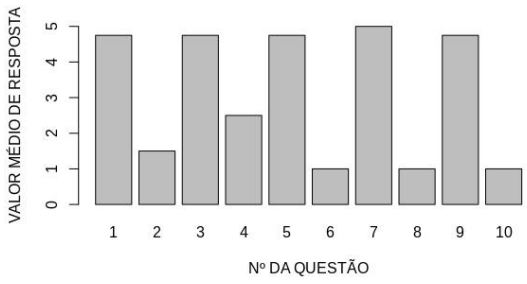

Figura 14: Média de resposta dos 10 primeiros itens do questionário.

De maneira geral, todos os 10 itens que avaliam a usabilidade do sistema trazem um resultado positivo. É possível observar no gráfico que o item 4 teve a avaliação menos satisfatória. O item afirma que é necessário alguém com conhecimento técnico para realizar o uso da aplicação. Observando as respostas deste item de forma individual, podemos evidenciar que apenas um respondente concorda totalmente, enquanto os outros discordam parcialmente ou totalmente. A análise que pode ser feita é que um dos usuários precisou de auxílio para ativar o microfone do seu dispositivo, o que o fez concordar integralmente com a afirmativa.

Prosseguindo com a análise das respostas do primeiro grupo de itens, foi calculado o valor na escala de usabilidade do sistema (SUS) com base no método proposto por *Brooke* [25], usando as médias calculadas para cada item. O resultado obtido foi 92.5, em um intervalo entre 0 e 100. Uma pesquisa realizada por Bangor et al. [28], correlacionou alguns valores na escala com adjetivos. Em especial, valores como 85.5 ou 90.9 são caracterizados como "excelentes" e "o melhor que se poderia imaginar", respectivamente. Sendo assim, com base no resultado obtido, podemos concluir que o Leia Rapidinho obteve sucesso em oferecer um bom nível de usabilidade para os usuários.

O **segundo grupo** analisado é composto pelos itens 11 e 12 do questionário, que pretendem avaliar se a aplicação atende aos seus objetivos. Os valores médios das respostas desses itens (representados no gráfico de barras da figura 15) revelam que, do ponto de vista dos professores, a aplicação é capaz de reter a atenção de um aluno e exercitar a leitura de palavras.

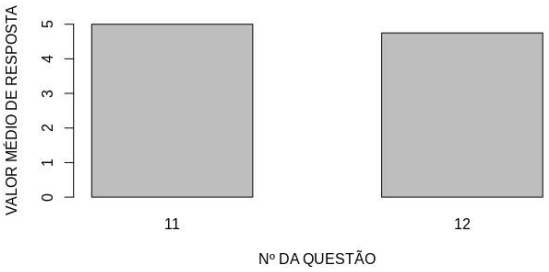

Figura 15: Valor médio das respostas dos itens 11 e 12

O **terceiro grupo** é composto apenas pelo item 13. Por meio desse item, os respondentes propuseram, com suas próprias palavras, duas melhorias para a aplicação que são expostas e comentadas a seguir:

- Sugestão de melhoria 01: "Percebi que algumas palavras se repetiram nas diferentes rodadas, sugiro menos repetição das palavras."
- Sugestão de melhoria 02: "Representar a criança com um personagem da escolha dela, antes do início do jogo, e liberar acessórios, roupas e novidades para o personagem, a cada determinado número de acertos."

A sugestão de melhoria 01 é importante, pois a repetição de palavras em diferentes rodadas de leitura, especialmente em rodadas que se sucedem, pode desencorajar o usuário da aplicação a continuar fazer seu uso, já que não é interessante ler a mesma palavra diversas vezes.

A sugestão de melhoria 02 propõe um novo mecanismo de gamificação que é capaz de reter ainda mais a atenção do usuário para exercitar a leitura, que é um objetivo da aplicação.

7. CONCLUSÃO

Os resultados obtidos na fase de avaliação desse trabalho revelam o Leia Rapidinho como uma ferramenta promissora ao propor o exercício da leitura de palavras. Nas próximas subseções são apresentados os desafios, as limitações e melhorias que podem ser abordadas em trabalhos futuros.

7.1 Desafios e limitações

Realizar o reconhecimento de voz a partir de uma interface provida pelos *browsers* limita o uso da aplicação. Cada solução de *browser* pode fornecer a interface ou não e, caso forneça, ainda tem a autonomia de utilizar a sua própria estratégia de implementação (alguns fazem o reconhecimento apenas de palavras em inglês, por exemplo). A partir dessa limitação, houve o desafio de manter um bom nível de usabilidade do sistema, de forma a identificar quais *browsers* não são compatíveis com a aplicação para que seja exibido uma mensagem amigável de erro para o usuário.

7.2 Trabalhos Futuros

As limitações expostas na subseção anterior e as sugestões de melhoria coletadas na fase de avaliação revelam que há espaço para trabalhos futuros que podem evoluir o Leia Rapidinho.

Um possível trabalho futuro é utilizar um único serviço de reconhecimento de voz para que os usuários façam a leitura das palavras. Dessa forma, a aplicação será independente da interface que realiza reconhecimento de voz provida pelos *browsers*.

Outro trabalho futuro consiste em atender as duas sugestões de melhoria coletadas na fase de avaliação desse trabalho. Sendo assim, é proposto uma evolução na página de desempenho, para que o usuário visualize seu progresso de forma mais elegante (por exemplo: o usuário pode ser representado na aplicação, sendo capaz de desbloquear itens para si ao obter progresso). Também se propõe o estabelecimento de alguma regra para que palavras lidas pelo usuário em rodadas anteriores não sejam exibidas novamente para leitura.

A avaliação do Leia Rapidinho pelo público infantil (público alvo) não foi realizada neste trabalho, porém é de extrema importância e deve ser realizada em trabalhos futuros, com a finalidade de avaliar a percepção desses usuários acerca do sistema.

8. REFERÊNCIAS

[1] Você sabe o que é fluência de leitura? Alfaebeto, 2018. Disponível em: <https://www.alfaebeto.org.br/2018/09/27/o-que-e-fluencia-de-leitura/>

[2] Ganho social da alfabetização se reflete diretamente sobre qualidade de vida. Revista Educação, 2017. Disponível em: <https://revistaeducacao.com.br/2017/09/19/ganho-social-da-alfabetizacao-se-reflete-diretamente-sobre-qualidade-de-vida/>

[3] MEC anuncia política nacional de Alfabetização para reverter estagnação na aprendizagem. Ministério da Educação, 2017. Disponível em: <http://portal.mec.gov.br/ultimas-noticias/211-218175739/56321-mec-anuncia-politica-nacional-de-alfabetizacao-para-reverter-estagnacao-na-aprendizagem>

[4] Relatório Brasil no PISA 2018 Versão Preliminar. Diretoria de avaliação da educação básica DAEB, 2019, p. 69. Disponível em: <https://download.inep.gov.br/acoes_internacionais/pisa/documentos/2019/relatorio_PISA_2018_preliminar.pdf>

[5] Web Speech API. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API>

[6] MERN. Disponível em: <https://www.mongodb.com/mern-stack>

[7] Read Along. Disponível em: <https://readalong.google/>

[8] Coy, Andre. (2013). On the Use of Automatic Speech Recognition to Facilitate Increased Literacy Rates in Jamaica. International Journal of Cross-Disciplinary Subjects in Education. Special Issue Volume 3. 1379-1386. 10.20533/ijcdse.2042.6364.2013.0192.

[9] Kanban. Disponível em: <https://pt.wikipedia.org/wiki/Kanban>

[10] Trello. Disponível em: <https://trello.com/pt-BR>

[11] React. Uma biblioteca JavaScript para criar interfaces de usuário. Disponível em: <https://pt-br.reactjs.org/>

[12] Redux. A Predictable State Container for JS Apps. Disponível em: <https://redux.js.org/>

[13] Semantic UI React. Disponível em: <https://react.semantic-ui.com/>

[14] React Speech Recognition. Disponível em: <https://github.com/JamesBrill/react-speech-recognition#readme>

[15] Flux. Application architecture for building user interfaces. Disponível em: <https://facebook.github.io/flux/>

[16] Containers vs presentational components. Disponível em: <https://medium.com/@yassimortensen/container-vs-presentational-components-in-react-8eea956e1cea>

[17] NodeJS Javascript Runtime. Disponível em: <https://nodejs.org/en/>

[18] Express.js. Framework web rápido, flexível e minimalista para Node.js. Disponível em: <https://expressjs.com/pt-br/>

[19] Mongoose. elegant mongodb object modeling for node.js. Disponível em: <https://mongoosejs.com/>

[20] JsonWebToken implementation for node.js. Disponível em: <https://github.com/auth0/node-jsonwebtoken>

[21] JWT. Disponível em: <https://pt.wikipedia.org/wiki/JSON_Web_Token>

[22] BCrypt. Disponível em: <https://github.com/kelektiv/node.bcrypt.js>

[23] MongoDB. Disponível em: <https://pt.wikipedia.org/wiki/MongoDB>

[24] Heroku: Cloud Application Platform. Disponível em: <https://en.wikipedia.org/wiki/Heroku>

[25] Brooke, John. (1995). SUS: A quick and dirty usability scale. Usability Eval. Ind.. 189.

[26] R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Disponível em: <https://www.R-project.org/>

[27] RStudio Team (2020). RStudio: Integrated Development Environment for R. RStudio, PBC, Boston, MA. Disponível em: <http://www.rstudio.com/>

[28] Bangor, Aaron & Kortum, Phil & Miller, James. (2009). Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. J. Usability Stud.. 4. 114-123.