UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS LAGO PEREIRA

SIG-EDU: um Sistema Web de Suporte ao Atendimento Educacional Especializado e à Educação Inclusiva

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

Orientador: Prof. Dr. Leandro Krug Wives
Coorientador: Prof. Me. Francisco Dutra dos Santos Jr.

Porto Alegre
2020

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais, Alex e Sue Helen por terem me dado ótimas condições para que eu tivesse uma educação de qualidade e entrasse na UFRGS.

Agradeço ao meu irmão por entender minhas ausências e por ser meu maior companheiro.

Aos meus amigos, aos antigos e aos novos, que compartilharam e ainda vão compartilhar momentos incríveis comigo. Em especial, ao Augusto, que esteve ao meu lado nos momentos mais difíceis da minha graduação.

Por fim, ao meu orientador Leandro e co-orientador Chico, por terem me dado a oportunidade de participar desse projeto e pela ótima orientação.

RESUMO

O serviço de Atendimento Educacional Especializado é uma modalidade para apoio e acompanhamento de alunos com necessidades educacionais especiais. Em Porto Alegre, esse serviço é denominado de “Sala de Integração e Recursos”, e está presente em toda a rede municipal de ensino fundamental. O desafio dos professores que trabalham com esse atendimento tem sido o processo organizativo dos registros e documentações, bem como as possibilidades de visualização de informações sobre as trajetórias dos alunos. O propósito deste trabalho é o desenvolvimento de uma aplicação *web* para dar suporte e qualificar as atividades realizadas por esses profissionais, mantendo a compatibilidade com uma base de dados já operacional para uma aplicação iOS existente. A aplicação resultante foi testada por professores de Salas de Integração e Recursos do município de Porto Alegre, que a avaliaram através do System Usability Scale, atingindo uma pontuação de 89,2 pontos, o que demonstra que o sistema teve sua usabilidade aprovada.

Palavras-chave: Sistema Web. Atendimento Educacional Especializado. Salas de Integração e Recursos. React. Firebase.

SIG-EDU: a Web System to Support Specialized Educational Care and Inclusive Education

ABSTRACT

The Specialized Educational Care is a service to support and accompany students with special educational needs. In Porto Alegre, this service is called "Resource Room" and is present in every municipal elementary school. The challenge of the teachers who work with this service has been the organizational process of records and documentation, as well as the possibility of visualizing information about the students' trajectories. The purpose of this paper is the development of a web application to support and qualify the activities performed by these professionals, while maintaining compatibility with an already operational database for an existing iOS application. The resulting application was tested by Porto Alegre Resource Room teachers, who evaluated it through the System Usability Scale, reaching a score of 89.2 points, which shows that the system had its usability approved.

Keywords: Web System. Specialized Educational Care. Resource Rooms. React. Firebase.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AEE	Atendimento Educacional Especializado
API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
iOS	iPhone Operational System
JS	JavaScript
LDBEN	Lei de Diretrizes e Bases da Educação Nacional
NEE	Necessidade Educacional Especial
NoSQL	Non Structured Query Language
SIGMA	Sistema de Gestão e Acompanhamento Móvel de Alunos Portadores de Necessidades Educacionais Especiais
SUS	System Usability Scale
UFRGS	Universidade Federal do Rio Grande do Sul
UUID	Universally Unique Identifier
UX	User Experience

SUMÁRIO

1 INTRODUÇÃO

A constituição federal garante a todo cidadão brasileiro o direito fundamental à educação. No entanto, é a partir dos princípios da Lei de Diretrizes e Bases da Educação Nacional (LDBEN), dos pressupostos de vários tratados internacionais em que o Brasil é signatário e da Lei de Inclusão, que as políticas e programas governamentais têm sido adotados para atender os estudantes com algum tipo de Necessidade Educacional Especial (NEE) (FREITAS; SCHNECKENBERG, 2014).

O atendimento educacional especializado (AEE) tem como função identificar, elaborar e organizar recursos pedagógicos e de acessibilidade que eliminem as barreiras para a plena participação dos alunos, considerando suas necessidades específicas (Ministério da Educação, 2009). Em Porto Alegre, esse serviço foi criado com a denominação de "Sala de Integração e Recursos", atualmente disponíveis em toda a rede municipal de ensino fundamental, onde professores especializados realizam atendimento e acompanhamento aos alunos com algum tipo de necessidade educacional especial.

Sob essa perspectiva, os professores que trabalham com o AEE têm a responsabilidade de registrar seus atendimentos e manter um histórico atualizado de cada aluno. Atualmente esse controle é feito manualmente, por cada professor, por meio de documentos físicos ou salvos em seus computadores de trabalho.

Uma aplicação já foi desenvolvida com o intuito de facilitar os registros e documentações feitos pelos professores de Salas de Integração e Recursos, porém foi feita somente para iOS.

1.1 Objetivo

Este trabalho tem como objetivo desenvolver uma aplicação *web*, compatível com a solução já desenvolvida para iOS, apoiando os professores de Salas de Integração e Recursos em suas atividades de organização de registros e documentações. A criação de uma interface amigável e uma visualização horizontal dos eventos do aluno são aspectos importantes decididos para essa implementação. Com a criação e integração da aplicação *web*, o acesso ao sistema será universalizado.

1.2 Estrutura do Texto

Esta seção busca dar um panorama de como o texto deste trabalho está estruturado, bem como os principais pontos abordados. O texto está organizado em seis capítulos, o Capítulo 1 faz a introdução do trabalho e os outros capítulos estão organizados da seguinte forma:

- Capítulo 2: apresenta os trabalhos, aplicativos e pesquisas existentes com objetivos similares ao presente trabalho;
- Capítulo 3: descreve a arquitetura do sistema e as tecnologias utilizadas na sua construção;
- Capítulo 4: apresenta as metodologias, ferramentas, padrões utilizados durante o desenvolvimento do trabalho, bem como os componentes e telas criados para a aplicação;
- Capítulo 5: apresenta os testes realizados com usuários e seus resultados;
- Capítulo 6: apresenta as conclusões do trabalho, suas limitações e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Este capítulo busca dar ao leitor um panorama dos trabalhos, aplicativos e pesquisas existentes com objetivos similares aos do presente trabalho.

2.1 Modelagem do processo de atendimento em salas de recursos para alunos com necessidades educacionais especiais

O trabalho (MUNDEL, 2019) mapeou as atividades realizadas nas Salas de Integração e Recursos da rede municipal de ensino de Porto Alegre, organizando-as em uma modelagem de processo. Esse trabalho foi importante para o entendimento do processo de atendimento em Salas de Integração e Recursos e das necessidades dos profissionais que trabalham com esse atendimento.

2.2 SIR-EDU

O Sistema Integrado de Recursos Educacionais para a Gestão do Acompanhamento de Alunos com Necessidades Especiais (SIR-EDU) (FERREIRA, 2017) desenvolveu um protótipo de sistema para a gestão e acompanhamentos dos alunos de Salas de Integração e Recursos.

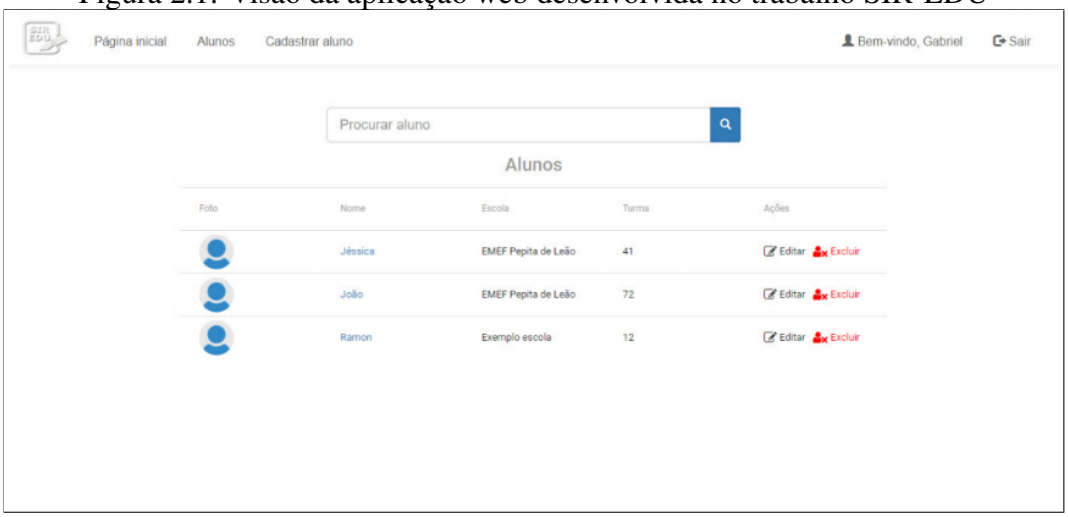
O sistema desenvolvido (Figura 2.1) teve como foco a criação de um *back-end* com uma API para acesso aos dados e um *front-end* de testes foi elaborado para testar o primeiro. Não há uma preocupação específica com a segurança das informações dos alunos e sua usabilidade não foi testada, validada e aprovada por usuários finais.

2.3 SGA-EDU

O Sistema de Gestão e Acompanhamento Educacional (SGA-EDU) (LUCAS, 2018) teve como objetivo refazer e expandir o protótipo descrito anteriormente (SIR-EDU), adicionando funcionalidades e focando-se inicialmente no *front-end*, aproveitando-se da API desenvolvida.

O SGA-EDU (Figura 2.2) não teve como foco o desenvolvimento de uma aplicação responsiva e não estava integrado com outras plataformas. Também não teve como

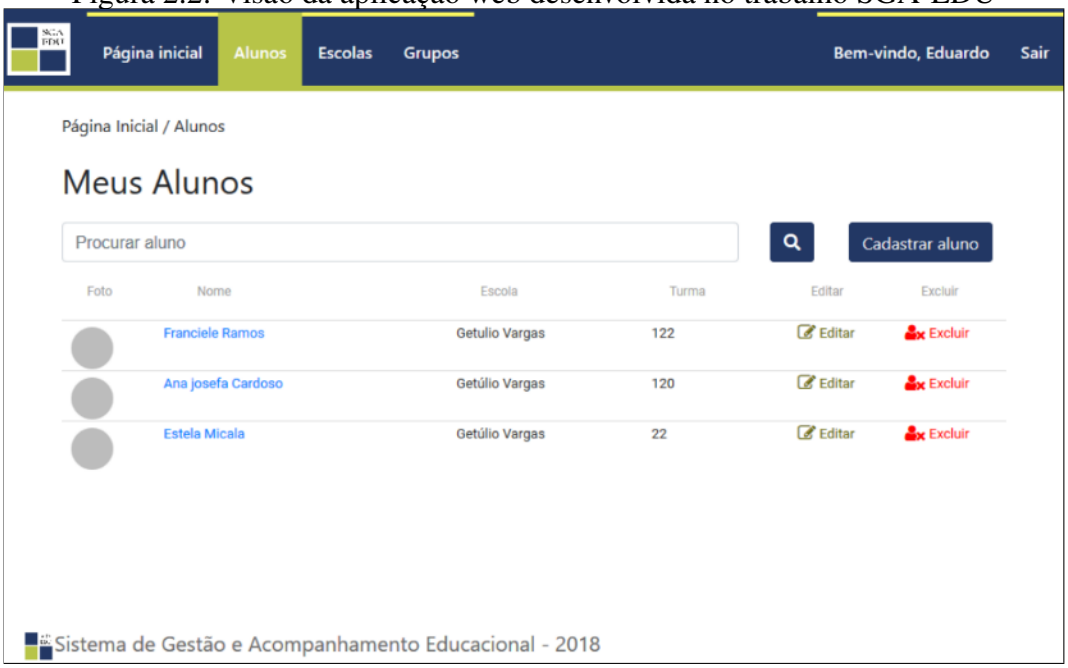
Figura 2.1: Visão da aplicação web desenvolvida no trabalho SIR-EDU



Fonte: (FERREIRA, 2017)

foco a garantia de segurança das informações e não foi validado com usuários finais em termos de usabilidade. Esses pontos são relevantes para a universalização e uso prático do sistema.

Figura 2.2: Visão da aplicação web desenvolvida no trabalho SGA-EDU



Fonte: (LUCAS, 2018)

2.4 SIGMA

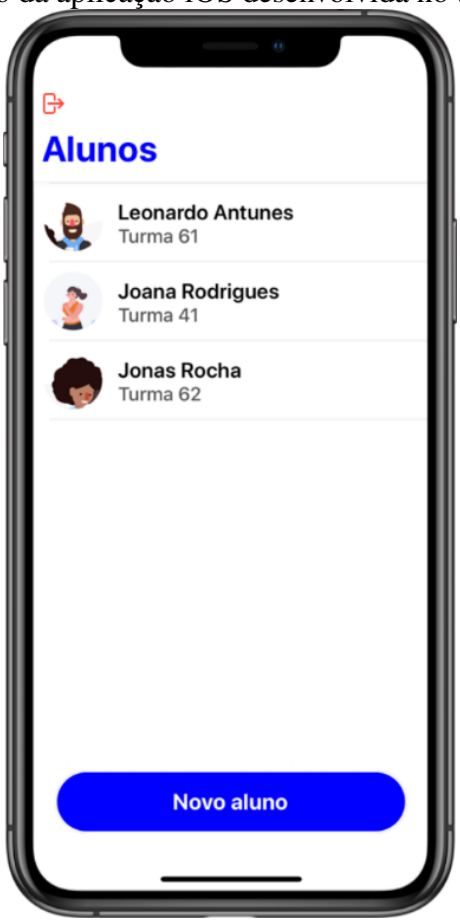
O Sistema de Gestão e Acompanhamento Móvel de Alunos Portadores de Necessidades Educacionais Especiais (SIGMA) (SILVA, 2019) deu início ao sistema SIG-EDU aqui apresentado. Naquele trabalho, foi desenvolvida uma aplicação móvel iOS e também iniciado o desenvolvimento de uma aplicação *web*, porém a segunda não foi completada, pois não era o foco do trabalho.

O resultado do SIGMA foi portanto um aplicativo iOS (Figura 2.3) que solucionou os problemas de segurança dos trabalhos anteriores, configurando regras de acesso ao banco de dados para que somente usuários autenticados e com permissão possam acessar os dados sensíveis de cada aluno. Além disso, foram realizados testes com usuários reais, professores das respectivas Salas de Integração e Recursos, para avaliar a usabilidade do aplicativo iOS e os resultados obtidos foram satisfatórios, estando ele inclusive em uso por uma professora. No entanto, a grande maioria dos professores usa a plataforma Android. Além disso, uma versão *web* seria importante porque alguns formulários são melhor preenchidos e visualizados em telas maiores, o que também ocorre com as mídias e eventos cadastrados pelos usuários.

2.5 Resumo do Capítulo

Os trabalhos descritos neste capítulo comprovam o valor do projeto e evidenciam a oportunidade de qualificação do trabalho dos professores de Sala de Integração e Recursos através do uso da tecnologia. A aceitação da aplicação SIGMA pelos professores abriu espaço para a expansão e universalização do acesso ao sistema, que são grandes motivadores para a realização deste trabalho.

Figura 2.3: Visão da aplicação iOS desenvolvida no trabalho SIGMA



Fonte: (SILVA, 2019)

3 ARQUITETURA DO SISTEMA E TECNOLOGIAS UTILIZADAS

Este capítulo é dedicado a apresentar a arquitetura de *software* e as tecnologias utilizadas na implementação do sistema desenvolvido neste trabalho.

Em um sistema computacional que envolva comunicação entre dois ou mais módulos, esses módulos são constantemente rotulados entre duas categorias, incentivando a separação de conceitos da camada de apresentação e a camada de acesso aos dados (LAMIM, 2014). Essas categorias são *front-end* e *back-end*. Enquanto o *front-end* é responsável por apresentar dados ao usuário e possibilitar sua interação com o sistema, o *back-end* é a parte do sistema que não é visível ao usuário.

A arquitetura do sistema é apresentada na primeira seção do capítulo e as tecnologias utilizadas são apresentadas nas outras duas seções de acordo com os conceitos introduzidos acima: *Front-End* e *Back-End*.

3.1 Arquitetura do Sistema

Conforme a complexidade de um sistema computacional aumenta, os algoritmos e estruturas de dados utilizados no seu desenvolvimento deixam de ser a única preocupação do desenvolvedor. Surge a necessidade de se criar definições a respeito da estrutura geral do sistema e da forma como seus componentes interagem entre si (GARLAN; SHAW, 1994). Essa organização geral é feita com o uso de uma arquitetura de *software*. É importante notar que um sistema pode possuir mais de uma arquitetura, uma vez que cada camada da aplicação pode conter suas próprias regras e restrições.

A definição das arquiteturas a serem utilizadas em um sistema é um dos passos iniciais de um projeto de *software*, visto que esta escolha irá guiar todo o desenvolvimento, e alterá-la pode ser bastante custoso. A seguir, está descrita em maiores detalhes a arquitetura de *software* utilizada na implementação deste trabalho.

3.1.1 Cliente-servidor

A arquitetura Cliente-Servidor consiste em uma ou mais aplicações clientes que fazem requisições para uma aplicação servidora (Encyclopaedia Britannica, 2019). Os clientes fazem a interface com o usuário e se comunicam com o servidor, que acessa os

dados da aplicação, normalmente em um Banco de Dados.

Exemplos de aplicações clientes são aplicativos móveis, *websites* e aplicações para a obtenção e envio de *e-mail*. Já as aplicações servidoras não fazem interação direta com os usuários, apenas recebem requisições dos clientes, as processam e respondem. Então, nesses exemplos, os servidores seriam os componentes do sistema responsáveis por processar e armazenar os dados enviados pelos clientes, bem como fornecem dados para eles.

Neste trabalho, a arquitetura foi utilizada da seguinte forma: o cliente apresenta-se como a página *web* com a qual os professores usuários do sistema interagem para cadastrar alunos e eventos; o servidor é a parte do sistema que fornece os dados para a página *web* e armazena no banco de dados os dados por ela enviados.

3.2 *Front-End*

O termo *front-end*, em um sistema de informação, é utilizado para descrever a parte do *software* responsável por apresentar informações ao usuário e lidar com suas interações, geralmente através do uso de uma interface gráfica.

Fazendo um paralelo com a arquitetura de *software* citada anteriormente, Cliente-Servidor, o *front-end* comporta-se como o cliente da aplicação, fazendo requisições para o *back-end* (detalhado mais para frente), que atua como servidor. As subseções a seguir descrevem as tecnologias utilizadas para a implementação do *front-end* no sistema *web* desenvolvido neste trabalho.

3.2.1 HTML

Linguagens de marcação são usadas para a criação de documentos com elementos semânticos que destaquem a estrutura de seu conteúdo. Nesse contexto, HTML (*Hyper-text Markup Language* é a linguagem de marcação padrão da *web*. Documentos HTML são transmitidos de um servidor *web* para um navegador na máquina cliente, que então irá renderizar visualmente a estrutura que o criador do documento codificou (MOZILLA, 2019b).

As marcações são feitas através de *tags*: palavras ou caracteres reservados da linguagem, de acordo com a sua sintaxe. Em HTML, as *tags* são denotadas por palavras

entre caracteres ‘<’ (menor que) e ‘>’ (maior que). Alguns exemplos de *tags* HTML são:

- **`<img>`**: representa uma imagem;
- **`<p>`**: representa um parágrafo;
- **`<ol>`**: representa uma lista ordenada (*ordered list*);
- **`<li>`**: representa um item de uma lista (*list item*);
- **`<table>`**: representa uma tabela;

3.2.2 CSS

Para customizar a visualização de documentos HTML, usa-se a linguagem de *stylesheet* CSS, sigla para *Cascading Style Sheets* (MOZILLA, 2019a). CSS pode ser usada para realizar tanto tarefas simples como adicionar cores na página e alterar a fonte do texto quanto para tarefas mais complexas como animações de elementos da página e configurar *layouts* adaptáveis para diferentes tipos e tamanhos de tela.

A primeira especificação W3C[1] referente ao CSS deu-se em 1996. Na época, a separação entre o conteúdo e a apresentação de um documento *web* começava a se fundir, e o CSS forneceu uma maneira de separar estas responsabilidades (WELLS, 2018).

3.2.3 Javascript

Javascript (também referido como js) é uma linguagem de programação multi-paradigma criada em 1995 em colaboração entre a Netscape Communications[2] e a Sun Microsystems[3] (Netscape, 1995). Inicialmente foi chamada de LiveScript, e, alguns meses após seu anúncio, recebeu o nome que possui atualmente.

Junto com HTML e CSS, Javascript é um dos pilares do desenvolvimento *web*. É usado no *front-end* como a linguagem de *script* responsável por adicionar comportamentos que o HTML por si só não é capaz de apresentar ao usuário, como a adição de animações complexas, carregamento de informações e integração com os serviços do *back-end*. Existem inúmeros *frameworks* para o desenvolvimento *web* construídos com

[1] W3C é a sigla para *World Wide Web Consortium*, i.e., um consórcio de diversas empresas com o objetivo de estabelecer padrões para a Internet

[2] Empresa de tecnologia californiana que fez muito sucesso nos anos 1990.

[3] Empresa norte-americana criadora do Java. Foi comprada pela Oracle em 2009.

base no Javascript, dentre os quais se destacam jQuery[4], React[5] (utilizado neste trabalho) e Angular[6] (Stack Overflow, 2019).

A linguagem também pode ser utilizada no *back-end* para a criação de servidores *web*, por exemplo. Isso é possível com o uso do ambiente Node.js, abordado na seção que trata do *back-end*.

3.2.4 React

React é uma biblioteca JavaScript criada pelo Facebook e utilizada por várias empresas líderes de mercado como Uber, Netflix, Whatsapp, Instagram e pelo próprio Facebook (DEVATHON, 2019). Segundo pesquisa realizada em 2019 pelo StackOverflow, é o segundo *framework web* mais popular, perdendo apenas para o jQuery (Stack Overflow, 2019).

A biblioteca visa simplificar o desenvolvimento de interfaces visuais, através de uma sintaxe declarativa e do uso de componentes reutilizáveis. React introduz alguns novos conceitos, que serão explorados ao longo das subseções desse capítulo.

3.2.4.1 JSX

JavaScript XML (JSX) é uma extensão de sintaxe para JavaScript que permite a escrita de *markup* (linguagem de marcação) e lógica de programação em um mesmo arquivo. Embora React não exija a utilização de JSX, a maioria das pessoas considera-a útil como auxílio visual ao trabalhar com a interface de usuário dentro do código JavaScript. Ela também permite que o React mostre mensagens de erro e aviso mais úteis (Facebook Inc., 2019c). A Figura 3.1 contém um exemplo de elemento criado utilizando JSX.

Figura 3.1: Exemplo de um elemento criado com JSX

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

Fonte: (Facebook Inc., 2019c)

[4]<https://jquery.com> Acesso em novembro de 2019
[5]<https://reactjs.org> Acesso em novembro de 2019
[6]<https://angularjs.org> Acesso em novembro de 2019

3.2.4.2 Componentes

Uma aplicação desenvolvida em React é composta de vários componentes, cada um responsável por renderizar uma parte pequena e reutilizável de HTML. Os componentes podem ser aninhados em outros componentes para permitir que aplicativos complexos sejam construídos a partir de blocos simples (GeeksforGeeks, 2019).

Existem duas formas principais de manipular dados entre componentes em React, *Props* e *State*, esses conceitos são explicados nas subseções seguintes.

3.2.4.3 Props

Props, uma abreviação de *properties* (propriedades), são os parâmetros passados para um componente. Esses parâmetros são imutáveis, não podem ser alterados pelo componente.

A Figura 3.2 apresenta um exemplo de utilização de *props*, disponibilizado na documentação oficial do React (Facebook Inc., 2019a). No exemplo, o elemento `Welcome` recebe a *prop* `name` com o valor "Sara".

Figura 3.2: Exemplo de utilização de *props*

```
const element = <Welcome name="Sara" />;
```

Fonte: (Facebook Inc., 2019a)

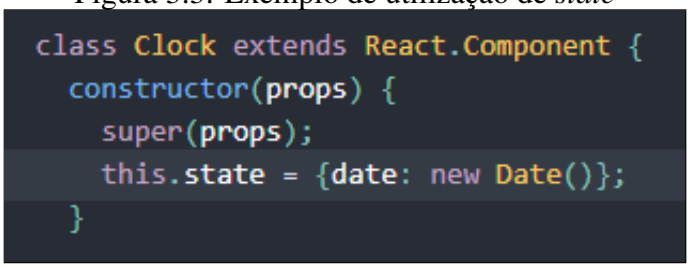
3.2.4.4 State

O *State* do componente é similar as *props*, com a diferença de ser privado e totalmente controlado pelo componente (Facebook Inc., 2019e). O *state* de um componente é criado em sua inicialização e pode utilizar as *props* recebidas para definir seus valores.

Alterações no *state* de um componente devem ser feitas através de chamadas do método `setState` e sempre que uma alteração é feita, o componente é re-renderizado pelo React.

A Figura 3.3 apresenta um exemplo disponibilizado na documentação oficial do React (Facebook Inc., 2019e) de definição do *state* de um componente. No exemplo, o componente `Clock` inicializa seu *state* com o atributo `date`, que recebe uma nova instância da classe `Date`.

Figura 3.3: Exemplo de utilização de *state*



Fonte: (Facebook Inc., 2019e)

3.2.4.5 Virtual DOM

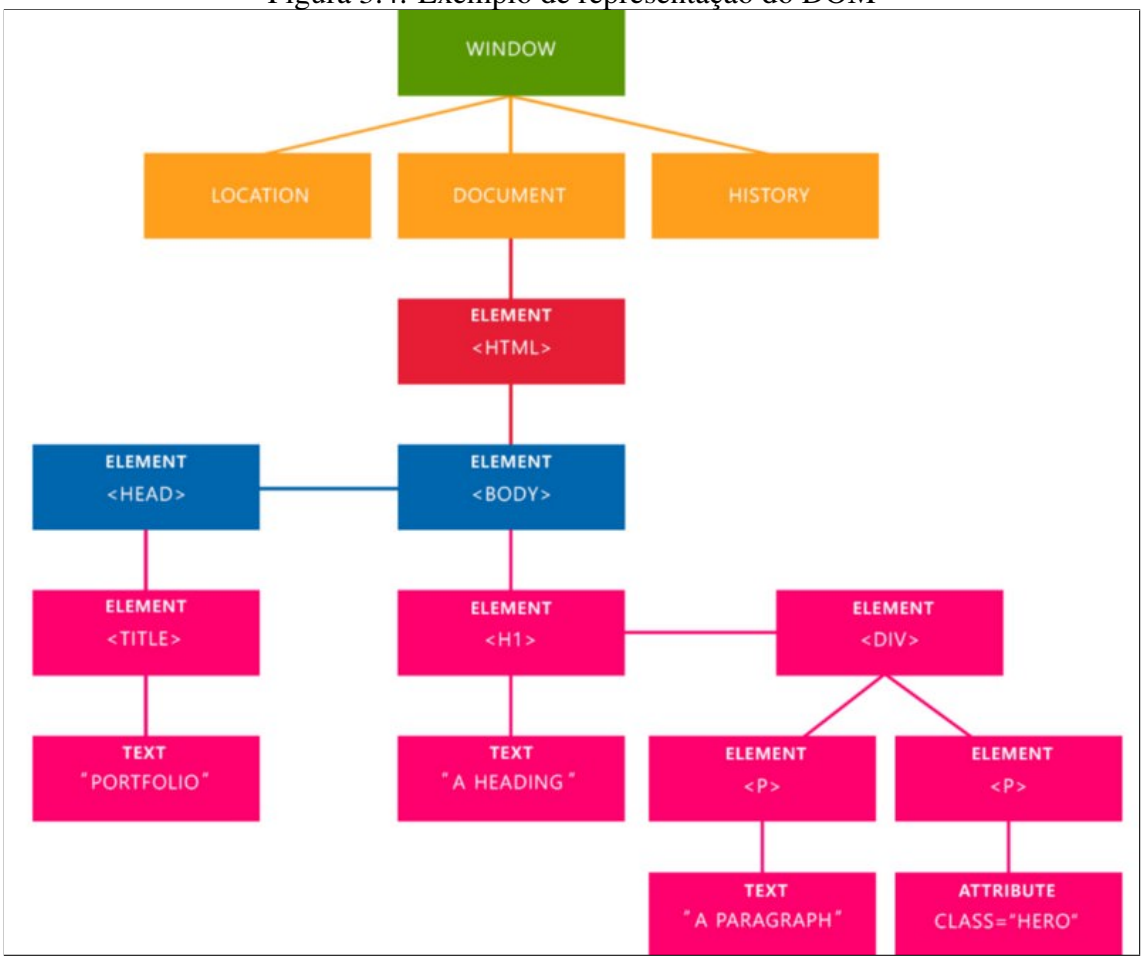
DOM significa *Document Object Model*, ou seja, Modelo de Objeto de Documento. O DOM é mantido na memória do navegador e representa o documento usando uma estrutura de árvore, com nodos e objetos, servindo de interface de programação para documentos HTML, permitindo a criação, modificação e consulta de elementos HTML de uma maneira orientada a objetos, direto da linguagem de programação sendo utilizada. Com isso, linguagens como JavaScript podem se conectar à página para alterar a estrutura, o estilo e conteúdo do documento (MOZILLA, 2019c). A Figura 3.4 contém um exemplo dessa estrutura.

Sempre que o DOM é alterado, o navegador precisa realizar operações para redesenhar a página, recalculando o CSS e refazendo o layout. Em páginas complexas, essas operações podem se tornar ineficientes. Para resolver este problema, React faz o uso do Virtual DOM com o objetivo de diminuir o uso de recursos do navegador, tornando essas operações mais rápidas (COPES, 2018).

O Virtual DOM é um conceito de programação onde uma representação ideal, ou "virtual", da interface do usuário é mantida em memória e sincronizada com o DOM por uma biblioteca como a ReactDOM[7]. Esse processo é chamado de reconciliação (Facebook Inc., 2019f). Quando algum componente é alterado, o React não descarta o que já foi renderizado. Em vez disso, ele compara o Virtual DOM com o DOM "real", computa as diferenças, descobre quais sub-árvores precisam ser atualizadas e então atualiza apenas essas sub-árvores, tornando o processo mais eficiente (BUNA, 2019).

[7] <https://www.npmjs.com/package/react-dom> Acesso em novembro de 2019

Figura 3.4: Exemplo de representação do DOM



Fonte: (MALDONADO, 2018)

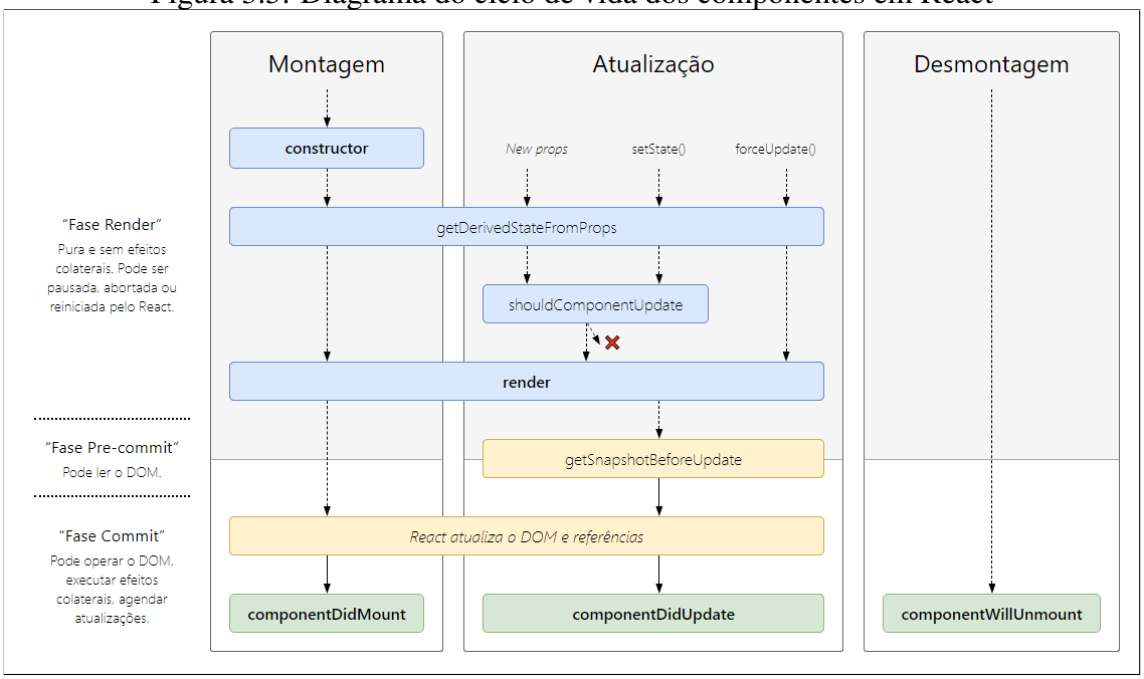
3.2.4.6 Ciclo de Vida

Um conceito muito importante para o desenvolvimento em React é o Ciclo de Vida (Figura 3.5) dos componentes. Cada componente tem seu próprio ciclo de vida, que pode ser definido como a série de métodos invocados em diferentes fases da existência do componente. Em cada etapa são executados diversos métodos, chamados de métodos de ciclo de vida (*lifecycle methods*, em inglês) que podem ser sobrescritos pelo desenvolvedor para adicionar comportamentos a um momento específico da vida do componente.

As três fases do ciclo de vida segundo (Facebook Inc., 2019d) são:

- **Montagem**: ocorre quando uma instância de um componente está sendo criada e inserida no DOM;
- **Atualização**: a atualização pode ser causada por alterações em *props* ou no *state* e seus métodos são chamados quando um componente está sendo re-renderizado;
- **Desmontagem**: ocorre quando um componente está sendo removido do DOM.

Figura 3.5: Diagrama do ciclo de vida dos componentes em React



Fonte: (Facebook Inc., 2019d)

A seguir serão detalhados os métodos do ciclo de vida (Figura 3.5), de acordo com a documentação oficial do React (Facebook Inc., 2019d). É importante observar que, conforme são lançadas novas versões do React, o ciclo de vida pode ser alterado. Por isso, os métodos descritos abaixo são referentes à versão 16.4 do React, utilizada neste trabalho.

- **`constructor`**: é chamado antes que o componente seja montado e normalmente é utilizado apenas para dois propósitos: inicializar o *state* do componente e ligação (*binding*) de *event handlers* à uma instância;
- **`getDerivedStateFromProps`**: invocado imediatamente antes da chamada do método **`render`**, tanto na fase de **montagem** quanto nas **atualizações** subsequentes. É utilizado para atualizar o *state* do componente a partir da mudança nas *props*;
- **`render`**: é o único método obrigatório em um componente, retorna o JSX a ser renderizado no DOM. O método deve ser puro, ou seja, não deve alterar o *state* ou as *props* do componente;
- **`componentDidMount`**: invocado imediatamente após um componente ser montado (inserido na árvore);
- **`shouldComponentUpdate`**: é invocado antes da renderização do componente para decidir se ele deve ou não ser re-renderizado. Este método é utilizado para

alterar o comportamento padrão do React, que re-renderiza o componente sempre que há uma alteração em seu *state* é alterado;

- **`getSnapshotBeforeUpdate`**: é invocado imediatamente antes que o retorno da renderização mais recente seja escrito no DOM. Isto permite que o componente capture alguma informação do DOM, como a posição do *scroll*, antes que ela seja alterada;
- **`componentDidUpdate`**: é invocado imediatamente após alguma atualização ocorrer;
- **`componentWillUnmount`**: é invocado imediatamente antes da desmontagem e destruição do componente.

3.3 *Back-End*

O *back-end* é responsável por atender a solicitações das aplicações de *front-end* de um sistema. Também cabe ao *back-end*, em geral, fazer o intermédio entre a aplicação utilizada pelo usuário final (como aplicativos móveis, ou sites *web*) e o banco de dados que contém as informações do sistema. Além disso, o *back-end* é, geralmente, a parte responsável por realizar computações mais pesadas, diminuindo a carga de operações no dispositivo do usuário final.

As subseções a seguir apresentam as tecnologias utilizadas para na implementação do *back-end* do sistema desenvolvido neste trabalho.

3.3.1 Node.js

Node.js[8] é um ambiente de execução para JavaScript *server-side*, isto é, na aplicação servidora do sistema.

A principal característica que diferencia o Node.js de outras linguagens como PHP[9], Java[10] e C#[11], é o fato de sua execução ser *single-thread*. Ou seja, apenas uma *thread* é responsável por executar o código Javascript da aplicação. Apesar disso, o Node.js é capaz de tratar requisições de forma concorrente através de chamadas de E/S (entrada

[8] <https://nodejs.dev> Acesso em novembro de 2019
[9] <https://www.php.net/> Acesso em novembro de 2019
[10] <https://www.java.com/pt_BR/> Acesso em novembro de 2019
[11] <https://docs.microsoft.com/pt-br/dotnet/csharp/> Acesso em novembro de 2019

e saída) não-bloqueantes, o que evita o bloqueio da *thread* enquanto se aguarda o retorno de uma operação de leitura ou escrita (Opus Software, 2018). Assim sendo, um servidor Node.js é executado em um único processo, sem criar uma nova *thread* para cada requisição recebida.

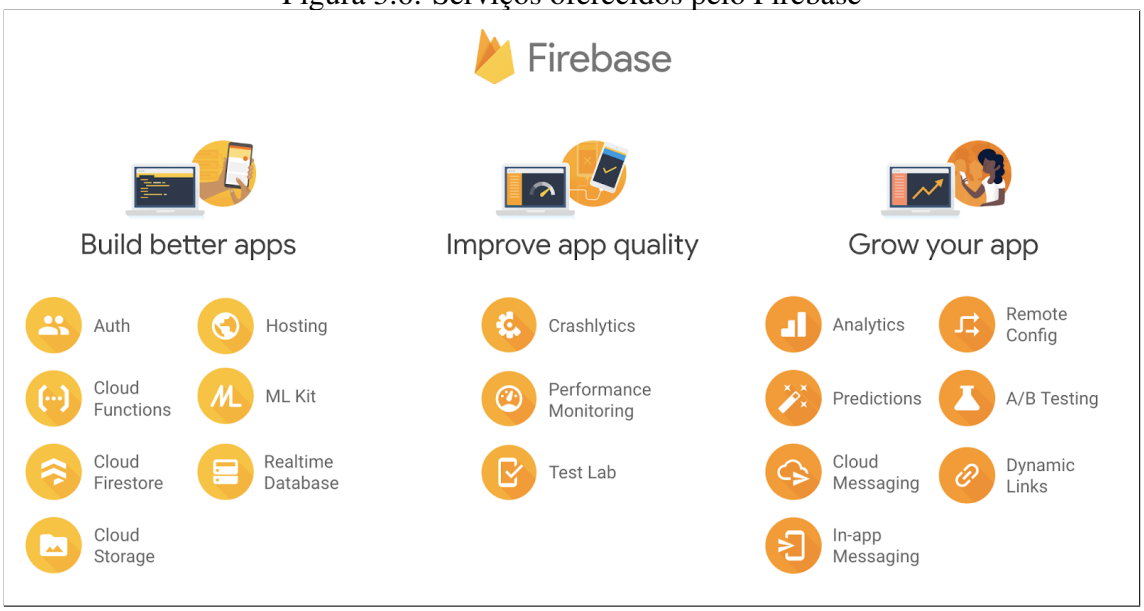
A lista de produtos que utilizam Node.js na implementação de seus serviços inclui: Netflix, Trello, PayPal, LinkedIn, Walmart e Uber, segundo (CHRZANOWSKA, 2017).

3.3.2 Firebase

Firebase é uma plataforma de desenvolvimento da Google que oferece diversos serviços (Figura 3.6) para a construção de aplicações móveis e *web*.

Neste projeto, foram utilizados três serviços providos pelo Firebase: *Firebase Auth*, para o cadastro e autenticação dos usuários, *Cloud Firestore*, para o banco de dados e *Cloud Storage* para o armazenamento de mídias (imagens, áudios e vídeos). Esses serviços são detalhados nas subseções seguintes.

Figura 3.6: Serviços oferecidos pelo Firebase



Fonte: (FIREBASE, 2019b)

3.3.2.1 Firebase Auth

FirebaseAuth é um serviço de autenticação que oferece suporte a diversas opções de credenciamento, como e-mail, número de telefone, Facebook e Twitter (FIREBASE,

2019c). O serviço também provê funcionalidades de confirmação de e-mail, recuperação de senha e verificação por SMS.

3.3.2.2 Cloud Firestore

O *Cloud Firestore* é um banco de dados NoSQL (não-relacional) hospedado na nuvem que pode ser acessado por aplicações *Web*, iOS e *Android* através de SDKs[12] e bibliotecas nativas. Seguindo o modelo não-relacional, os dados são armazenados no *Cloud Firestore* em documentos que contém mapeamentos chave-valor. Esses documentos são armazenados em coleções, que podem ser usadas para organizar os dados e criar consultas (FIREBASE, 2019a).

O *Cloud Firestore* é completamente integrado com o *Firebase Auth*, por isso permite que o desenvolvedor crie de regras de segurança para restringir o acesso aos dados.

3.3.2.3 Cloud Storage

O *Cloud Storage* é um serviço de armazenamento de objetos em nuvem. Ele pode ser utilizado através de SDKs e bibliotecas que possibilitam o *download* e *upload* de mídias diretamente dos clientes (STORAGE, 2019).

Assim como o *Cloud Firestore*, o *Cloud Storage* também permite a criação de regras de segurança para controlar o acesso à arquivos individuais ou grupos de arquivos.

[12] Software Development Kits, conjuntos de ferramentas e bibliotecas para desenvolvimento em uma plataforma

4 PROJETO E DESENVOLVIMENTO

Este capítulo busca apresentar ao leitor as principais metodologias, ferramentas e padrões utilizados durante o desenvolvimento deste trabalho.

4.1 Métodos Ágeis

O conceito de Métodos Ágeis consiste em uma abordagem de gestão de projetos que oferece maior flexibilidade e adaptabilidade ao projeto, uma vez que ajudam a encarar as imprevisibilidades através de entregas incrementais e ciclos iterativos (BERNARDO, 2015). Por isto, é considerado uma alternativa mais moderna aos métodos tradicionais de gestão.

4.2 Scrum

Dentro do contexto de métodos ágeis, *Scrum* é uma metodologia que define um formato de time, eventos e artefatos para o desenvolvimento de um produto. O *Scrum* emprega um abordagem iterativa e incremental para otimizar a previsibilidade e controlar os riscos (SCHWABER; SUTHERLAND, 2017).

O formato de time sugerido no Guia do Scrum (SCHWABER; SUTHERLAND, 2017), por seus criadores é:

- **Product Owner (dono do produto)**: responsável por garantir a qualidade do produto final desenvolvido pelo time, a partir do gerenciamento e priorização do *backlog* do produto;
- **Time de desenvolvimento**: conjunto de profissionais que desenvolvedorem as funcionalidades priorizadas pelo *Product Owner*;
- **Scrum master (mestre do *Scrum*)**: responsável por garantir que os eventos e práticas do *Scrum* sejam adotados pelo time, organizando as reuniões e organizando a equipe.

Nos parágrafos seguintes, o Scrum é explicado segundo (ÁGIL, 2014):

No *Scrum*, os projetos são divididos em ciclos chamados de *Sprints*. Uma *Sprint* representa um período de tempo dentro do qual um conjunto de atividades deve ser exe-

cutado.

As funcionalidades a serem implementadas em um projeto são mantidas em uma lista conhecida como *Product Backlog*.

No início de cada *Sprint*, é realizada a cerimônia de *Sprint Planning*, onde é feito o planejamento do ciclo. O *Product Owner* prioriza as histórias de usuário a serem implementadas, a partir do *Product Backlog*.

As histórias escolhidas são então transferidas do *Product Backlog* para o *Sprint Backlog*. Este último conceito representa a lista de funcionalidades que devem ser implementadas durante a *Sprint*.

No início de cada dia da *Sprint*, é feita uma reunião chamada de *Daily*, com o objetivo de compartilhar conhecimento sobre o que foi feito pela equipe no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.

Ao final de uma *Sprint*, as funcionalidades implementadas são apresentadas pela equipe em uma reunião conhecida como *Sprint Review*.

Por fim, é feita uma retrospectiva para que a equipe discuta sobre os problemas enfrentados durante a *Sprint* e crie propostas para resolvê-los. Após a retrospectiva, a equipe parte para o planejamento do próximo *Sprint*, reiniciando o ciclo.

4.2.1 *ScrumBut*

A utilização do *Scrum* com ressalvas, adaptando algumas práticas definidas no Guia do Scrum em decorrência de características específicas do projeto ou limitações do time, é chamada de *ScrumBut*, ou "Scrum mas".

Segundo o site Scrum.org (2019), a sintaxe utilizada no *ScrumBut* é: “Nós usamos *Scrum*, mas `<motivo>`, `<solução alternativa>`”.

Neste projeto, por ser desenvolvido por apenas uma pessoa, algumas práticas e eventos do *Scrum* acabaram sendo flexibilizadas ou descartadas:

- Nós usamos *Scrum*, mas por contarmos com apenas um desenvolvedor, não fazemos reuniões diárias;
- Nós usamos *Scrum*, mas por contarmos com apenas um desenvolvedor, ele também faz o papel de *Scrum Master*;
- Nós usamos *Scrum*, mas por contarmos com apenas um desenvolvedor, não fazemos retrospectiva.

4.3 Histórias de Usuário

Em um ambiente ágil, uma história de usuário é um instrumento utilizado no processo de levantamento de requisitos para descrever a especificação de uma funcionalidade do software (LONGO; SILVA, 2014). Ela fornece uma maneira simples para que os desenvolvedores e clientes consigam detalhar o que o sistema deve fazer, facilitando o desenvolvimento, que ocorre em etapas (BECK, 2000).

Uma das premissas das histórias de usuário é que elas devem ser facilmente compreendidas. Por esse motivo, foi utilizado o modelo sugerido por (COHN, 2008), que segue o seguinte formato: "Como `<tipo de usuário>`, quero `<objetivo>` para que `<razão>`".

As histórias de usuário utilizadas neste trabalho estão listadas abaixo. Foram reutilizadas as principais histórias definidas em (SILVA, 2019) e as histórias 1, 6 e 8 foram definidas especificamente para o sistema *web* desenvolvido neste trabalho.

1. Como professor, quero um aplicativo web para que eu possa acessar do meu computador;
2. Como professor, quero me registar usando minhas credenciais (*e-mail* e senha) para que eu possa manter minhas informações seguras (SILVA, 2019);
3. Como professor, quero fazer o acesso usando minhas credenciais (*e-mail* e senha) para que eu possa visualizar minhas informações seguras (SILVA, 2019);
4. Como professor, quero sair da minha sessão para que outros professores possam usar o mesmo dispositivo (SILVA, 2019);
5. Como professor, quero adicionar, remover e editar alunos para poder adicionar eventos aos meus alunos (SILVA, 2019);
6. Como professor, quero ver todos os meus alunos;
7. Como professor, quero excluir alunos e eventos para que eu possa organizar o meu trabalho (SILVA, 2019);
8. Como professor, quero ver os eventos dos alunos em uma linha do tempo horizontal para que eu possa visualizar a ordem de ocorrência dos eventos;
9. Como professor, quero ver os conteúdos dos eventos para que eu possa ter uma melhor compreensão do evento (SILVA, 2019);
10. Como professor, quero fazer o envio de áudio, vídeo e imagens das realizações de um aluno para que eu possa registrar suas realizações (SILVA, 2019);

11. Como professor, quero ser capaz de criar um parecer para que não precise fazer isso fora do aplicativo (SILVA, 2019);
12. Como professor, quero que a interface do aplicativo seja perfeita e rápida de usar, para que eu possa registrar eventos de alunos com eficiência (SILVA, 2019).

4.4 Sprints

O desenvolvimento do projeto teve uma duração de cinco *sprints*, cada uma com duas semanas. Os principais objetivos de cada uma delas estão descritos a seguir:

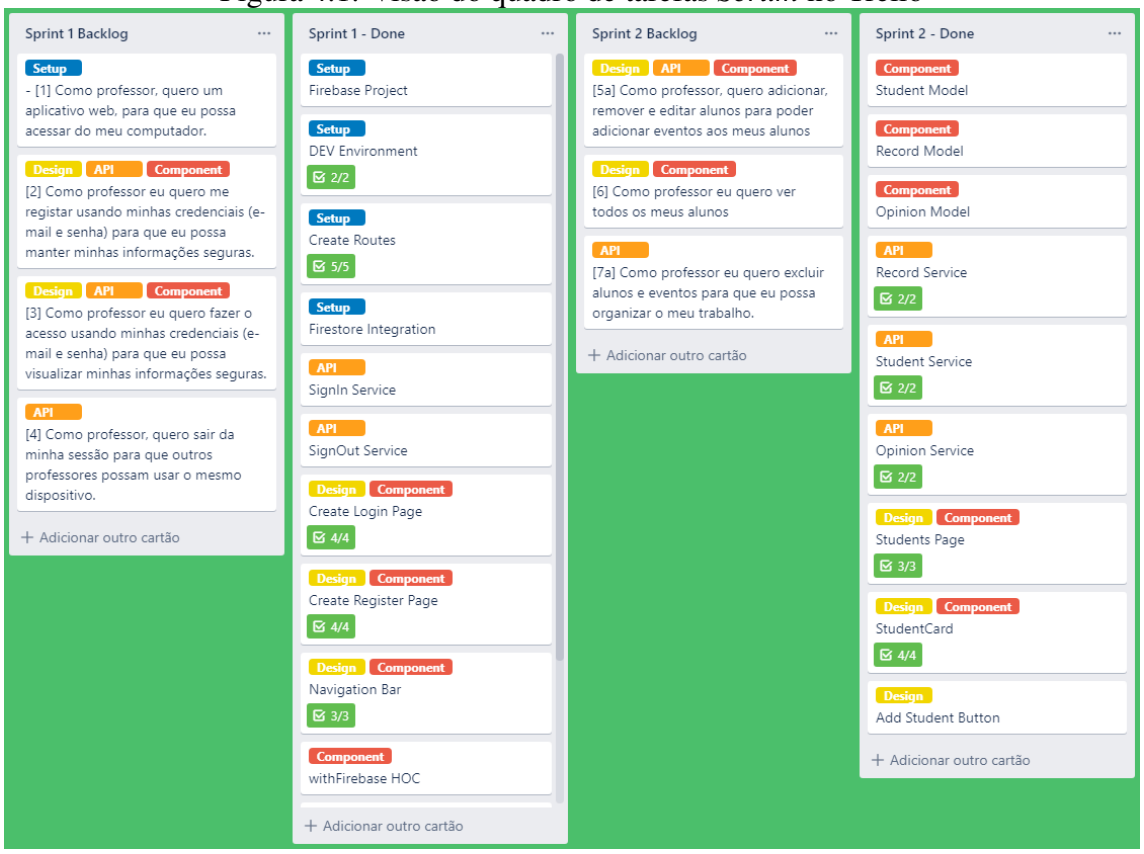
- ***Sprint* 1**: Configuração do projeto do *Firebase*, criação do ambiente de desenvolvimento, definição das principais rotas e criação do serviço de cadastro e autenticação;
- ***Sprint* 2**: Modelagem das entidades, desenvolvimento dos componentes da página Alunos e criação dos serviços de consulta, inserção, atualização e remoção no banco de dados;
- ***Sprint* 3**: Finalização dos componentes da página Alunos, início do desenvolvimento da página de Aluno e criação do serviço de *Storage* para consulta, inserção, atualização e remoção de mídias;
- ***Sprint* 4**: Desenvolvimento dos componentes para exibição de mídias e dos componentes relacionados a eventos e pareceres;
- ***Sprint* 5**: Correção de erros, ajustes na interface e preparação dos testes com usuários.

4.5 Quadro de *Scrum*

Para a organização e controle do andamento das *sprints*, foi criado um quadro de *Scrum*. Para isso foi utilizado o Trello[1], uma ferramenta de colaboração para organizar o projeto em quadros, que fornecem uma visualização gráfica dos itens que estão sendo trabalhados, quais pessoas estão envolvidas em cada item e qual o *status* de uma tarefa no projeto (TRELLO, 2017). A Figura 4.1 mostra uma visão das duas primeiras *sprints* no quadro.

[1] <https://trello.com> Acesso em novembro de 2019

Figura 4.1: Visão do quadro de tarefas *Scrum* no Trello



Fonte: Provido pelo autor.

4.6 Entidades Modeladas

Esta seção é dedicada a apresentar as entidades utilizadas no sistema. Foram modeladas as entidades definidas no trabalho (SILVA, 2019), realizando o mínimo de alterações possível, visando facilitar a integração no futuro.

4.6.1 Teacher

A entidade *Teacher* é utilizada para representar um professor. Um documento do tipo *Teacher* é criado no *Firebase* quando o usuário se cadastra no sistema. A seguir estão descritos os campos implementados para essa entidade:

- **`id`** - *String* no formato *UUID*, um identificador único universal utilizado para identificar o professor e para associar os alunos ao professor
- **`email`** - *E-mail* cadastrado para o professor, armazenado com o tipo *String*;
- **`name`** - Nome do professor, armazenado com o tipo *String*;

- **`phone`** - Telefone do professor, armazenado com o tipo *String*;
- **`timestamp`** - Data de criação do professor, armazenado com o tipo *Number*;
- **`lastModified`** - Data da última modificação no professor, armazenado com o tipo *String*.

4.6.2 Student

A entidade *Student* é utilizada para representar um aluno. Um documento do tipo *Student* é criado no *Firebase* quando o usuário cria um novo aluno.

- **`id`** - *UUID* (identificador único universal) utilizado para identificar unicamente o aluno e para associar os eventos ao aluno, armazenado como *String*;
- **`teacherId`** - *UUID* referenciando o professor que criou o aluno, *String*;
- **`name`** - Nome do aluno, armazenado como *String*;
- **`generalRegistry`** - RG do aluno, armazenado com o tipo *Number*;
- **`fatherName`** - Nome do pai do aluno, armazenado com o tipo *String*;
- **`motherName`** - Nome da mãe do aluno, armazenado com o tipo *String*;
- **`birthDate`** - Data de nascimento do aluno, armazenada com o tipo *String*;
- **`schoolId`** - Nome da escola do aluno, armazenado com o tipo *String*;
- **`classNumber`** - Turma do aluno, armazenada com o tipo *String*;
- **`series`** - Ano do aluno, armazenado com o tipo *String*;
- **`shift`** - Turno das aulas do aluno (Manhã, Tarde ou Noite), armazenado com o tipo *String*;
- **`responsibleName`** - Nome do responsável pelo aluno, armazenado com o tipo *String*;
- **`relationship`** - Parentesco do responsável com o aluno, armazenado com o tipo *String*;
- **`phone`** - Telefone do responsável pelo aluno, armazenado com o tipo *String*;
- **`address`** - Endereço do responsável pelo aluno, armazenado com o tipo *String*;
- **`cid`** - *String* que representa o código da doença conforme a Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde - CID
- **`specialNeed`** - Necessidades especiais do aluno, armazenadas com o tipo *String*;
- **`history`** - Histórico do aluno, armazenado com o tipo *String*;

- **`termsOfUse`** - Aceitação do aluno para o aplicativo, armazenado com o tipo *Boolean*;
- **`timestamp`** - Data de criação do aluno, armazenado com o tipo *Number*;
- **`lastModified`** - Data da última modificação no aluno, armazenado com o tipo *Number*,

4.6.3 Record

A entidade *Record* é utilizada para representar um evento. Um documento do tipo *Record* é criado no *Firebase* quando o usuário adiciona um evento a um aluno.

- **`id`** - *UUID* utilizado para identificar unicamente o evento, armazenado com o tipo *String*;
- **`studentId`** - *UUID* referenciando o aluno ao qual o evento está associado, armazenado com o tipo *String*;
- **`teacherId`** - Identificador referenciando o professor ao qual o evento está associado, armazenado com o tipo *String*;
- **`mediaType`** - Tipo de mídia do evento (texto, áudio, vídeo ou imagem), armazenado com o tipo *String*;
- **`comment`** - Comentário do evento, armazenado com o tipo *String*;
- **`timestamp`** - Data de criação do evento, armazenado com o tipo *Number*;
- **`lastModified`** - Data da última modificação no evento, armazenado com o tipo *Number*.

Durante a implementação foi identificado um problema com as *timestamps* que dificultou a integração da aplicação com o banco de dados de produção. As *timestamps* utilizadas pela aplicação SIGMA são geradas em um formato específico do iOS. Por isso, surgiu a necessidade de alguns ajustes para a integração com o banco de dados de produção no futuro, evitando soluções temporárias.

4.6.4 Opinion

A entidade *Opinion* é utilizada para representar um parecer. Um documento do tipo *Opinion* é criado no *Firebase* quando o usuário adiciona um parecer a um aluno.

- **`id`** - *UUID* utilizado para identificar unicamente o parecer, armazenado com o tipo *String*;
- **`studentId`** - *UUID* referenciando o aluno ao qual o evento está associado, armazenado com o tipo *String*;
- **`teacherId`** - Identificador referenciando o professor ao qual o evento está associado, armazenado com o tipo *String*;
- **`content`** - Conteúdo do parecer, no formato chave-valor, armazenado com o tipo *Map*;
- **`timestamp`** - Data de criação do parecer, armazenado com o tipo *Number*;
- **`lastModified`** - Data da última modificação no parecer, armazenado com o tipo *Number*.

4.7 Ambiente de Desenvolvimento

Visando possibilitar o teste do sistema, foi criado um ambiente de desenvolvimento contendo cópias de todos os documentos e mídias da aplicação SIGMA. Isto foi feito para que os testes e alterações feitos durante o desenvolvimento do sistema descrito neste trabalho não afetassem o sistema já desenvolvido.

4.8 Git

Em desenvolvimento de software, o versionamento de código é essencial para manter o controle de versões funcionais. Para isso são utilizados sistemas de controle de versão que ajudam os desenvolvedores a gerenciar alterações no código-fonte ao longo do tempo. O *software* de controle de versão registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo, para que o desenvolvedor possa recuperar versões específicas mais tarde (CHACON; STRAUB, 2014).

Para este projeto, a ferramenta escolhida foi o Git, uma ferramenta *open source* para controle de versionamento criada por Linus Torvalds[2] em 2005 (CHACON; STRAUB, 2014). Git é hoje a ferramenta mais popular para controle de versão e é utilizada pelas maiores empresas do mundo. A ferramenta foi utilizada através do GitHub, uma plataforma de hospedagem de código para controle de versão e colaboração que permite que

[2]Engenheiro de software finlandês, criador do Kernel do Linux

várias pessoas trabalhem juntas em projetos de qualquer lugar (GITHUB, 2016).

4.9 Yarn

Anunciado pelo Facebook em 2016 com a proposta de ser mais rápido, seguro e confiável que o NPM[3] (MCKENZIE; NAKAZAWA; KYLE, 2016), Yarn é um gerenciador de pacotes para JavaScript que foi utilizado para controlar as dependências do projeto.

O arquivo `package.json`, utilizado pelo Yarn, contém os *scripts* para inicialização do ambiente de execução e as dependências do projeto. As principais dependências utilizadas foram: React, Firebase, CoreJS, ReactDOM, Webpack e Prop-Types.

O Yarn facilita o compartilhamento e execução do projeto por outros desenvolvedores ou em outras máquinas, pois garante que serão utilizadas as mesmas versões dos pacotes instalados inicialmente. Para instalar as dependências deve-se executar o comando `yarn install` no diretório raiz do projeto e para iniciar o ambiente de desenvolvimento basta executar o comando `yarn start`.

4.10 *Polyfills*

CoreJS[4] é uma biblioteca para JavaScript que inclui *polyfills* para ECMAScript[5]. Um *polyfill* é um pedaço de código usado para fornecer funcionalidades modernas em navegadores mais antigos que não o suportam nativamente (MOZILLA, 2019d).

Polyfills são importantes pois garantem que usuários com navegadores desatualizados tenham a mesma experiência que teriam acessando a aplicação através de navegadores modernos. Por este motivo, a biblioteca CoreJS foi utilizada no projeto.

4.11 *High Order Components*

High Order Component (HOC) é uma técnica avançada do React para reutilizar a lógica de um componente. *HOCs* não são parte do React, e sim um padrão que surgiu da própria natureza de composição do React (Facebook Inc., 2019b).

[3] Gerenciador de pacotes do Node

[4] <https://www.npmjs.com/package/core-js> Acesso em novembro de 2019

[5] linguagem de script que forma a base do JavaScript

Um HOC, como definido por Dan Abramov[6] em (ABRAMOV, 2015), consiste em uma função que recebe como parâmetro um componente já existente e retorna outro componente, envolvendo-o e atribuindo-lhe novas funcionalidades ou dados (MARUTA, 2018). Os principais HOCs criados neste trabalho são descritos a seguir:

- **`withAuthentication`**: criado para prover aos componentes os dados do usuário autenticado;
- **`withAuthorization`**: criado para adicionar uma lógica de autorização às páginas, redirecionando o usuário para a tela de *login* caso ele ainda não esteja autenticado;
- **`withFirebase`**: criado para prover aos componentes as funções de acesso ao banco de dados.

4.12 Heroku

Para a aplicação dos testes com usuários, foi necessário disponibilizar a aplicação na nuvem. Isso foi feito através do Heroku, uma solução de Plataforma como Serviço que permite ao desenvolvedor criar e subir rapidamente suas aplicações para a nuvem (STORI, 2013).

O *deploy*[7] da aplicação pode ser feito automaticamente, através da execução do comando `git push heroku master`. O Heroku se encarrega de instalar as dependências do projeto, necessárias para sua execução.

4.13 Componentes Principais

Nesta seção, são apresentados os principais componentes criados ao longo do desenvolvimento deste trabalho. Os componentes aqui descritos foram utilizados na criação das páginas que serão detalhadas na seção seguinte.

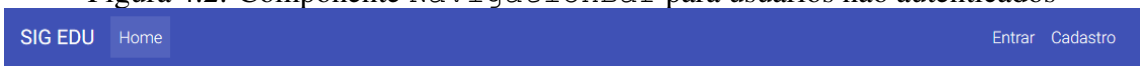
[6] Desenvolvedor do React e co-autor do Redux

[7] Disponibilizar um sistema para uso, seja num ambiente de desenvolvimento, para testes ou em produção

4.13.1 NavigationBar

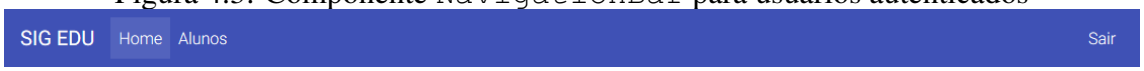
A barra de navegação é representada pelo componente `NavigationBar` e dispõe de duas formas de visualização: a primeira para usuários autenticados (Figura 4.2) e a segunda para usuários não autenticados (Figura 4.3).

Figura 4.2: Componente `NavigationBar` para usuários não autenticados



Fonte: Provido pelo autor.

Figura 4.3: Componente `NavigationBar` para usuários autenticados

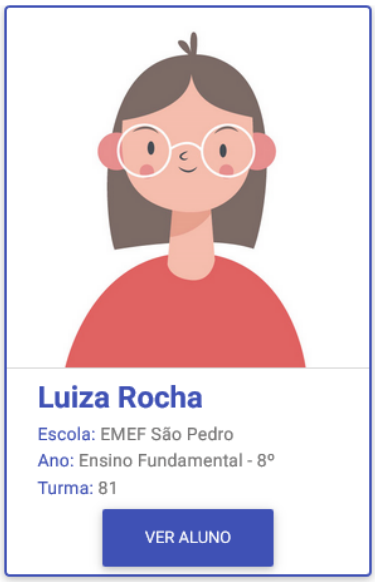


Fonte: Provido pelo autor.

4.13.2 StudentCard

O componente `StudentCard` (Figura 4.4) apresenta as informações principais de um aluno e dispõe de um botão que redireciona o usuário para a página do aluno.

Figura 4.4: Componente `StudentCard`

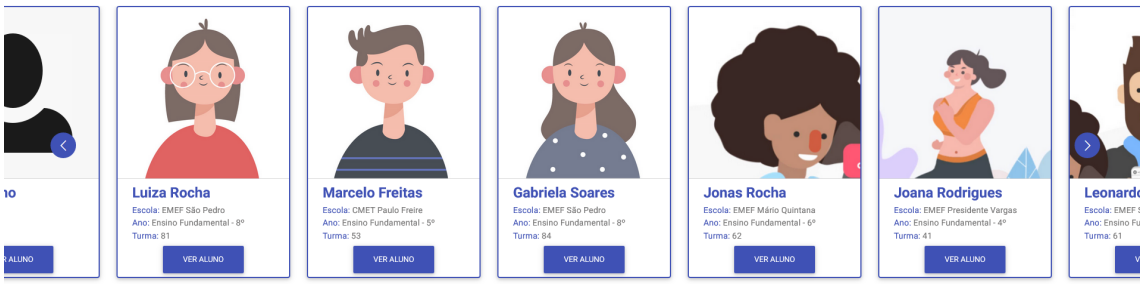


Fonte: Provido pelo autor.

4.13.3 StudentCarousel

O componente `StudentCarousel` (Figura 4.5) recebe uma lista de alunos e renderiza um carrossel de `StudentCards`, um para cada aluno.

Figura 4.5: Componente `StudentsCarousel`



Fonte: Provido pelo autor.

4.13.4 StudentSection

O componente `StudentSection` (Figura 4.6) é responsável por exibir as informações de um aluno.

Figura 4.6: Componente `StudentSection`

Fonte: Provido pelo autor.

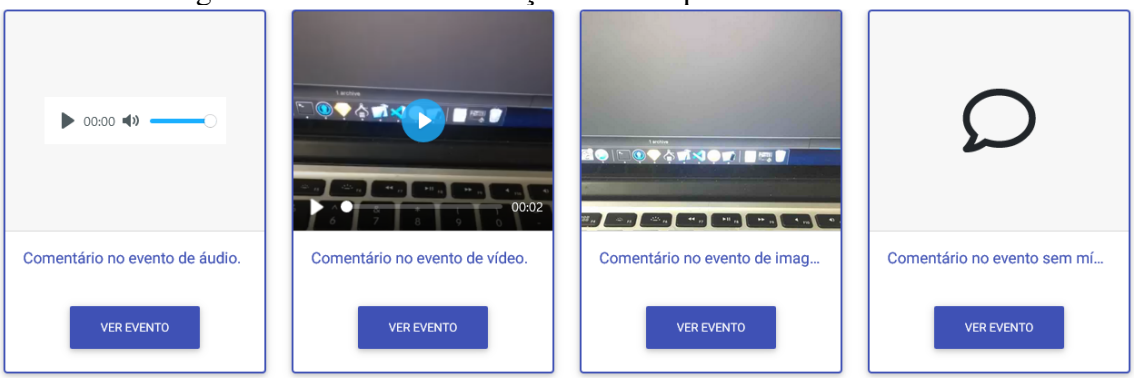
4.13.5 EventCard

O componente `EventCard` é utilizado para exibir uma pré-visualização de um evento do aluno. A Figura 4.7 apresenta quatro `EventCards`, um para cada tipo de evento: evento de áudio, evento de vídeo, evento de imagem e evento sem mídia (apenas

texto). O botão de um `EventCard` abre a visualização completa do evento.

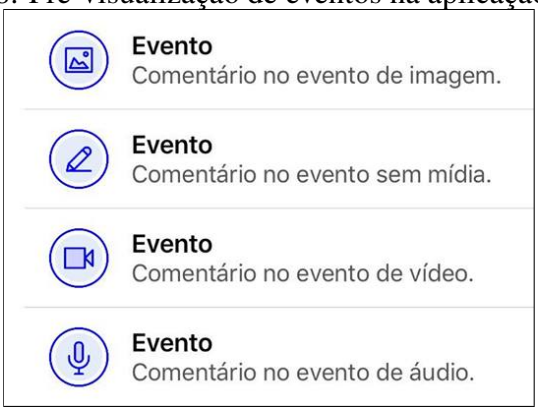
A pré-visualização das mídias dos eventos, desenvolvida para a aplicação *web*, é um diferencial em relação a aplicação SIGMA (Figura 4.8) que não oferece ao usuário uma forma de visualizar as fotos ou reproduzir áudios e vídeos sem acessar o conteúdo de cada evento.

Figura 4.7: Diferentes exibições do componente `EventCard`



Fonte: Provido pelo autor.

Figura 4.8: Pré-visualização de eventos na aplicação SIGMA

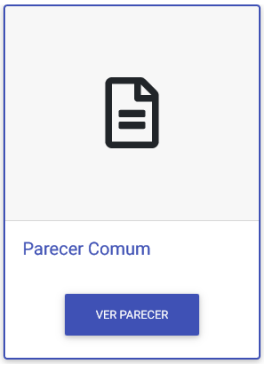


Fonte: (SILVA, 2019).

4.13.6 OpinionCard

Similar ao `EventCard`, o componente `OpinionCard` (Figura 4.9) é utilizado para exibir a pré-visualização de um parecer do aluno. O botão de um `OpinionCard` abre a visualização completa do parecer.

Figura 4.9: Componente `OpinionCard`

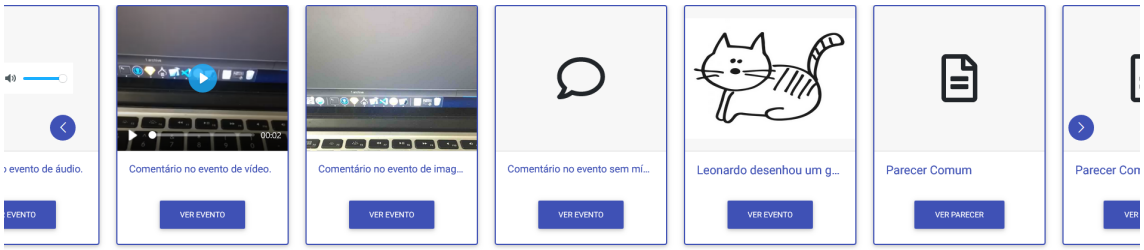


Fonte: Provido pelo autor.

4.13.7 RecordsCarousel

`RecordsCarousel` (Figura 4.10) é o componente responsável por exibir o histórico de eventos e pareceres do aluno. O componente recebe uma lista de eventos e uma lista de pareceres, renderizando-os, por ordem de criação, em uma linha do tempo horizontal.

Figura 4.10: Componente `RecordsCarousel`



Fonte: Provido pelo autor.

4.13.8 EventModal

O componente `EventModal` (Figura 4.11) apresenta a visualização completa de um evento do aluno e também é utilizado para a criação de um novo evento.

Este componente dispõe de uma funcionalidade que ainda não foi implementada na aplicação SIGMA: a edição de eventos. Na aplicação móvel, para alterar o comentário ou a mídia de um evento, deve-se excluir o evento e criar outro com a modificação desejada.

Figura 4.11: Componente `EventModal`

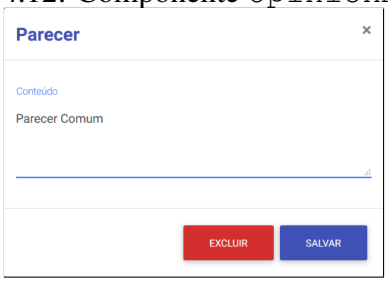
Fonte: Provido pelo autor.

4.13.9 OpinionModal

Similar ao `EventModal`, o componente `OpinionModal` (Figura 4.12) apresenta a visualização completa de um parecer do aluno e também é utilizado para a criação de um novo parecer.

O componente `OpinionModal`, assim como o `EventModal` também permite edição, o que é um diferencial em relação a aplicação SIGMA.

Figura 4.12: Componente `OpinionModal`



Fonte: Provido pelo autor.

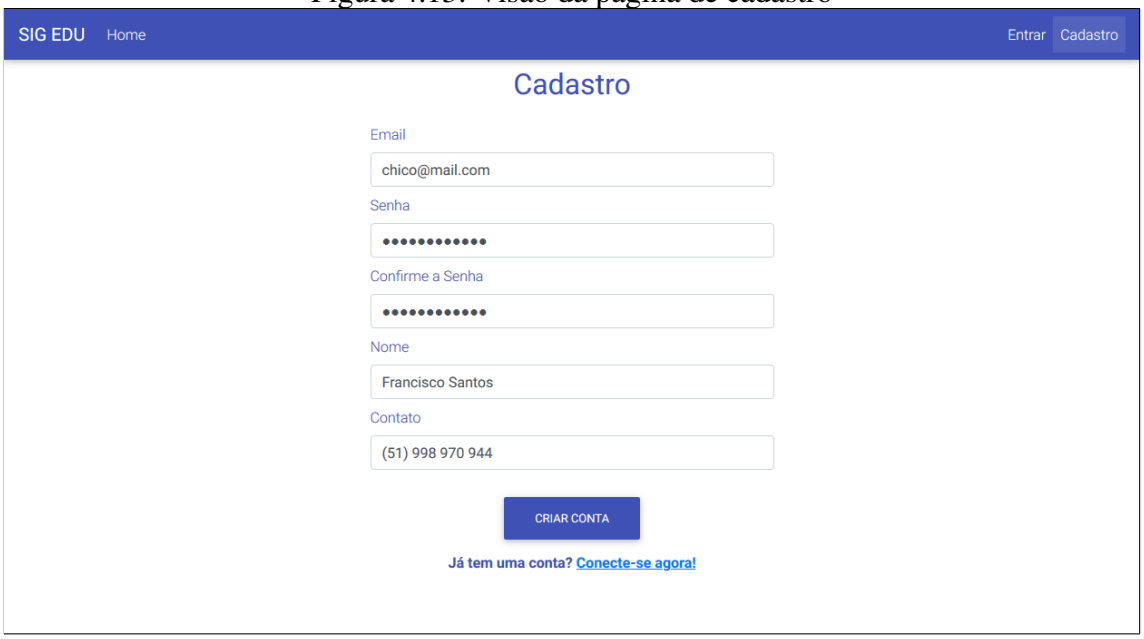
4.14 Páginas Principais

Esta seção é dedicada a dar um panorama das principais páginas *web* desenvolvidas para o sistema descrito neste trabalho. Estas páginas são compostas, principalmente, pelos componentes descritos na seção anterior.

4.14.1 Cadastro

Ao acessar a aplicação pela primeira vez, é necessário que o professor se cadastre no sistema. Isso pode ser feito através da página de cadastro (Figura 4.13), que pode ser acessada pela aba disponível na barra de navegação ou diretamente pela rota `/signup`. Os campos exibidos nesta página solicitam as principais informações do professor, que são utilizadas para a criação de uma nova conta.

Figura 4.13: Visão da página de cadastro



Fonte: Provido pelo autor.

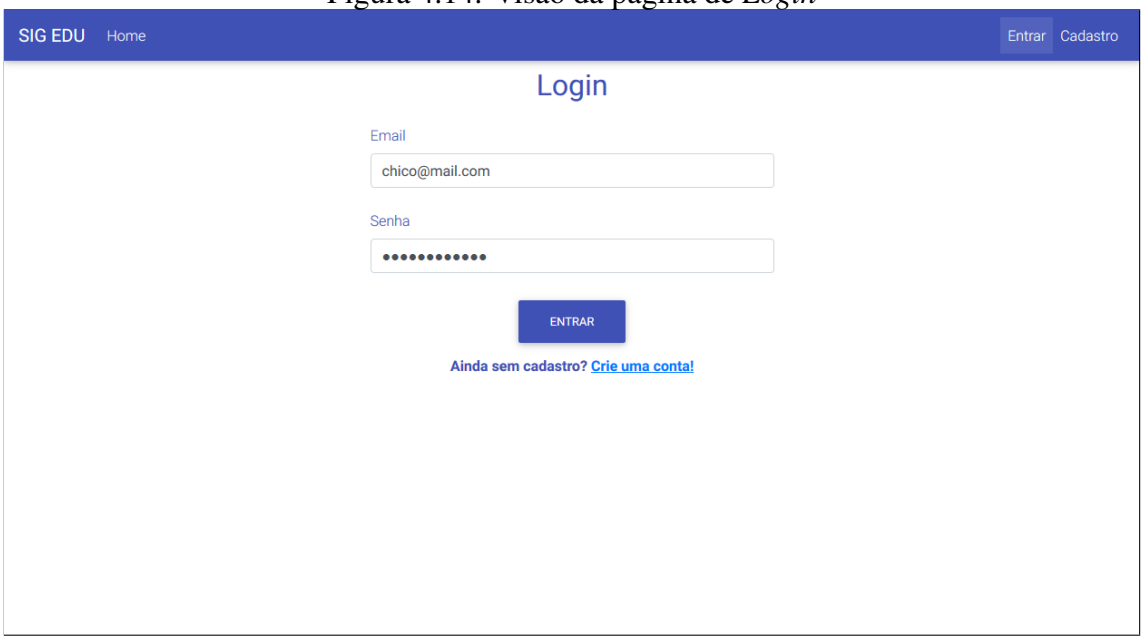
4.14.2 *Login*

A página de *Login* (Figura 4.14) é exibida quando o usuário clica na aba "Entrar"da barra de navegação e também pode ser acessada diretamente através da rota `/signin`. Autenticando-se com um *e-mail* e senha previamente cadastrados, é possível acessar o sistema.

4.14.3 Alunos

A página Alunos (Figura 4.15) exibe a lista de alunos criados pelo professor logado e possibilita a criação de novos alunos, através do clique no botão "Novo Aluno".

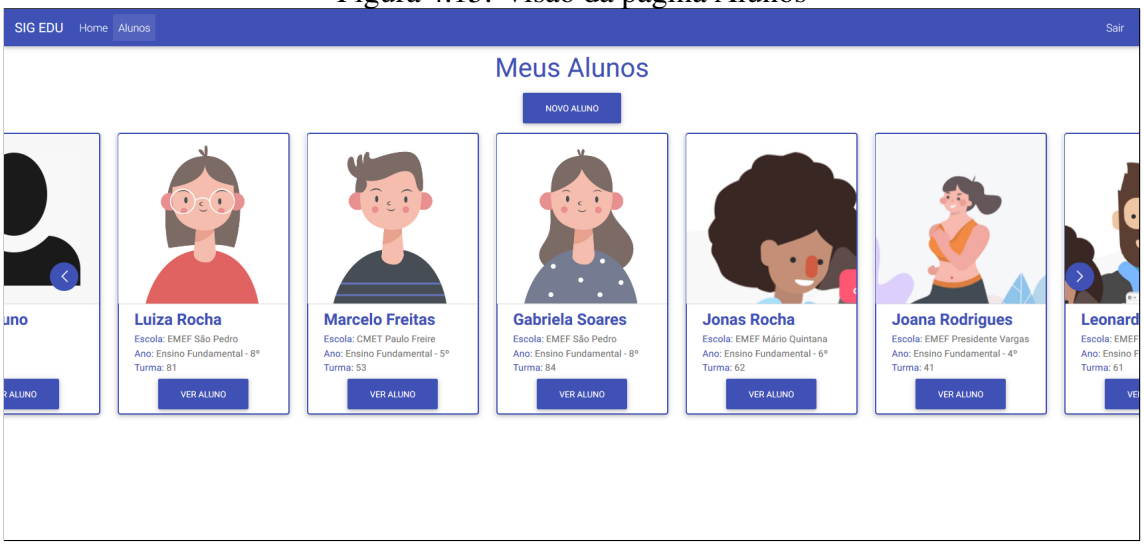
Figura 4.14: Visão da página de *Login*



Fonte: Provido pelo autor.

Esta página pode ser acessada através da barra de navegação ou pela rota `/students`.

Figura 4.15: Visão da página Alunos



Fonte: Provido pelo autor.

4.14.4 Aluno

A página do Aluno (Figura 4.16) compila todas informações sobre um aluno que são relevantes para o professor, podendo ser acessada através da página de Alunos ou diretamente pela rota `/student/:id`, onde `:id` é o identificador único gerado na criação

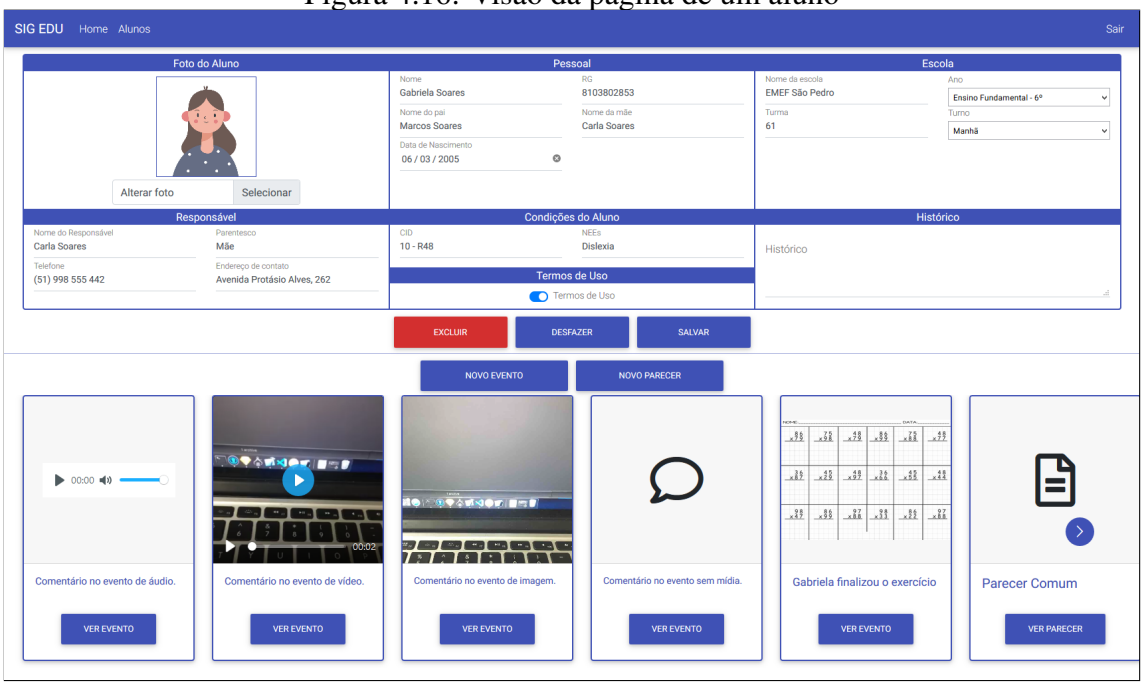
do aluno.

Exemplo de rota da página de um aluno:

`/student/4D00743D-031D-4D01-831B-A56DBA34B188`

Na parte superior são exibidas as informações sobre o aluno seguidas pelos botões para desfazer alterações, salvar e excluir o aluno. Na parte inferior é exibida a linha do tempo do aluno, com seus eventos e pareceres.

Figura 4.16: Visão da página de um aluno



Fonte: Provido pelo autor.

4.15 Responsividade

A aplicação desenvolvida conta com componentes responsivos. Isto quer dizer que sua representação visual é modificada dependendo do tamanho da tela do dispositivo utilizado pelo usuário. O controle de como cada objeto da página será exibido em cada tamanho de tela é feito através do uso de *media queries*[8], definindo as características do objeto para intervalos de tamanho de tela.

A Figura 4.17 apresenta um exemplo com as *media queries* utilizadas para definir o tamanho do componente *StudentCard* para diferentes tamanhos de tela.

A Figura 4.18 mostra a página de *login* para dispositivos móveis, nela é possível

[8]Recurso do CSS que permite que o conteúdo da página da web se adapte a diferentes tamanhos e resoluções de tela

Figura 4.17: Exemplo de *media queries* do componente *StudentCard*

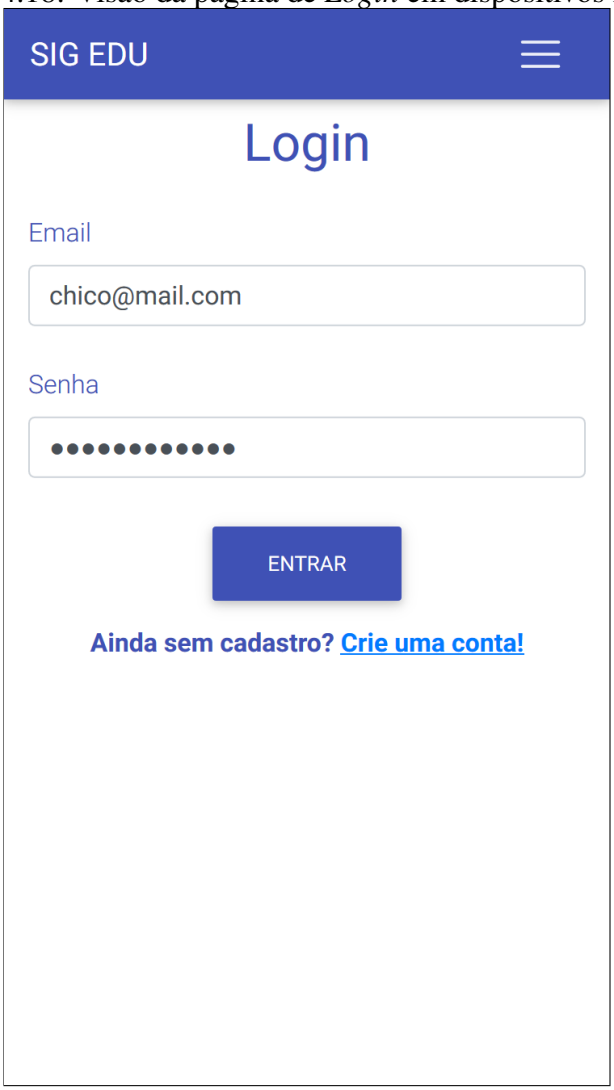
```css
/* SMARTPHONES */
@media only screen and (min-width: 300px) {
  .student-card {
    height: 350px;
    width: 150px;
    font-size: 0.5rem;
  }
}

/* SMARTPHONES LANDSCAPE */
@media only screen and (min-width: 480px) {
  .student-card {
    height: 400px;
    width: 200px;
    font-size: 0.6rem;
  }
}

/* TABLETS AND DESKTOPS */
@media only screen and (min-width: 1024px) {
  .student-card {
    height: 450px;
    width: 300px;
    font-size: 0.9rem;
  }
}
```

Fonte: Provido pelo autor.

observar a mudança na visualização da barra de navegação que é exibida em formato de "menu sanduíche".

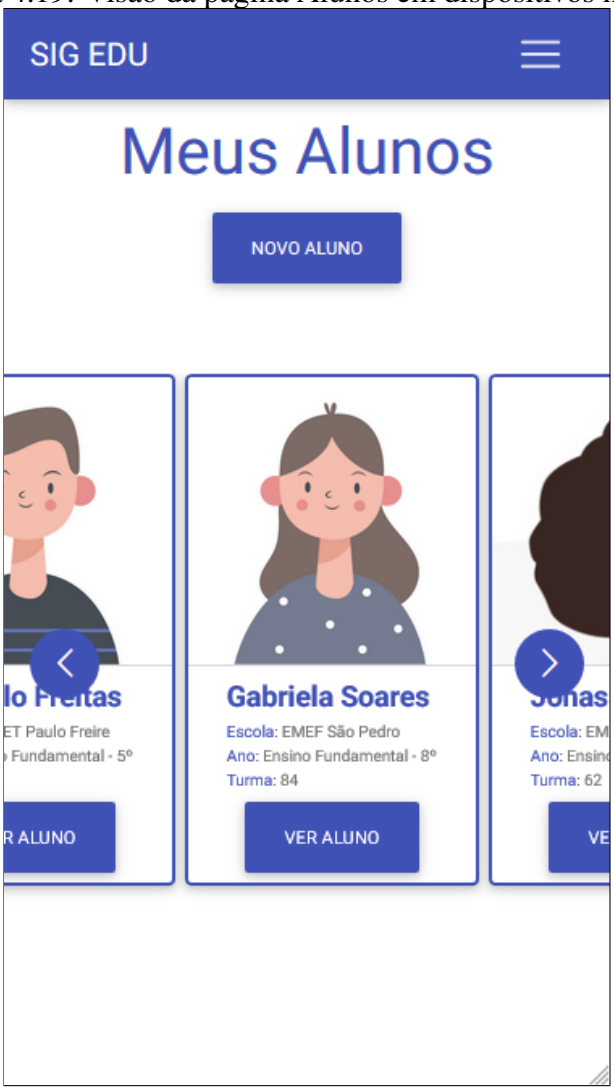
A Figura 4.19 apresenta a página Alunos para dispositivos móveis, nela além da mudança na visualização da barra de navegação, também é possível observar as mudanças nos componentes *StudentsCarousel* e *StudentCard* que têm seus tamanhos e fontes ajustados para uma melhor visualização em telas menores.

Figura 4.18: Visão da página de *Login* em dispositivos móveis



Fonte: Provido pelo autor.

Figura 4.19: Visão da página Alunos em dispositivos móveis



Fonte: Provido pelo autor.

5 AVALIAÇÃO DO SISTEMA

Neste capítulo é feita a apresentação e análise dos resultados dos testes realizados com usuários. Os testes foram feitos através da execução de tarefas pré-definidas e da avaliação através de um questionário sobre a usabilidade do sistema.

5.1 Perfil dos Usuários

O principal perfil dos usuários que participaram dos testes foi o de professor de Salas de Integração e Recursos, visto que são o público alvo do sistema. Porém, dada a simplicidade das tarefas e com o objetivo de aumentar a quantidade de usuários participantes, também foram aplicados testes com pessoas que não se encaixam no perfil principal.

Participaram dos testes 14 usuários, sendo eles: 7 professores de Salas de Integração e Recursos e 7 usuários que não fazem parte do público alvo da aplicação

5.2 System Usability Scale

O System Usability Scale (SUS) é um método, aplicado através de um questionário, utilizado para avaliar a usabilidade de um sistema. É um método extremamente popular, principalmente por apresentar um balanço entre ser cientificamente apurado e por sua aplicação não ser demorada, tanto para o usuário quanto para o pesquisador (TEIXEIRA, 2015).

Segundo o site Usability.gov (2019), os principais benefícios de utilizar o SUS são:

- É uma escala muito fácil de administrar aos participantes;
- Pode ser usada em amostras pequenas com resultados confiáveis;
- É válida - pode efetivamente diferenciar entre sistemas utilizáveis e inutilizáveis.

O questionário consiste em 10 afirmações, que devem ser avaliadas pelo usuário em uma escala de 1 a 5, onde 1 significa "Discordo Completamente"e 5 significa "Concordo Completamente". As afirmações são apresentadas abaixo:

1. Eu acho que gostaria de usar esse sistema com frequência;

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

5.3 Protocolo de testes

Com base nas histórias de usuário, foram criadas 13 tarefas para serem executadas pelos testadores, são elas:

1. Criar uma conta;
2. Criar um aluno;
3. Adicionar uma foto para o aluno;
4. Preencher a seção "Pessoal"do aluno;
5. Salvar as alterações do aluno;
6. Criar um evento com mídia;
7. Visualizar um evento;
8. Excluir um evento;
9. Criar um parecer;
10. Visualizar um parecer;
11. Excluir um parecer;
12. Excluir um aluno;
13. Sair da conta;

Após a realização das tarefas, os usuários foram convidados a responder um questionário contendo as 10 questões do SUS e outras 3 perguntas utilizadas para cálculo do tempo médio de realização das tarefas e perfilamento dos usuários.

As 3 perguntas extras foram:

1. Você é ou já foi professor de Sala de Integração e Recursos?
2. Você ainda participa ativamente do ambiente de Salas de Integração e Recursos?
3. Quanto tempo foi necessário para a realização das 13 tarefas?

5.4 Resultados

Nesta seção são apresentados os resultados dos testes e na subseção seguinte, os resultados são analisados. As respostas do questionário SUS são demonstradas na Tabelas 5.1, 5.2 e 5.3, sendo foram simplificadas em: concordo, neutro e discordo.

A Tabela 5.1 contém as respostas do questionário SUS obtidas nos testes com professores de Sala de Integração e Recursos. Participaram dos testes 7 professores, sendo que 4 deles ainda participam ativamente do ambiente de Salas de Integração e Recursos. A pontuação média obtida foi de 89,2 pontos.

A Tabela 5.2 contém as respostas do questionário SUS obtidas nos testes com usuários que não fazem parte do público alvo da aplicação. Foram aplicados 7 testes e a pontuação média obtida foi de 84,6 pontos. É possível observar que a maioria desses usuários respondeu que não gostariam de usar o sistema com frequência, o que é compreensível, pois eles não são beneficiados por seu uso.

A Tabela 5.3 contém as respostas compiladas de todos os usuários testadores. No total, participaram dos testes 14 pessoas e a pontuação média obtida considerando todos participantes foi de 87 pontos.

5.4.1 Análise dos Resultados

A diferença pontuações obtidas entre os testes com professores de Salas de Integração e Recursos e os testes com usuários comuns foi de apenas 4,6 pontos. Por este motivo e por sua maior relevância para este trabalho, serão avaliados os resultados obtidos nos testes com o público alvo do sistema, apresentados na Tabela 5.1.

O tempo médio para a realização das 13 tarefas propostas foi de 9 minutos e 44 segundos. Ou seja, os professores levaram em média 45 segundos para realizar cada tarefa em seu primeiro contato com a aplicação.

Os resultados mostram que a avaliação do sistema obteve 89,2 pontos de média

entre os 7 testes realizados com professores de Salas de Integração e Recursos, uma nota na escala SUS altamente satisfatória (BROOKE, 1996).

Além disso, nenhum dos respondentes achou o sistema desnecessariamente complexo, todos se sentiram confiantes ao usar o sistema e nenhum precisou aprender coisas novas antes de conseguir usar o sistema.

Tabela 5.1: Respostas do questionário SUS: professores de Salas Integração de Recursos

Questão	Afirmação	Concorda	Neutro	Discorda
Q1	Eu acho que gostaria de usar esse sistema com frequência.	85,7%	14,3%	0%
Q2	Eu acho o sistema desnecessariamente complexo.	0%	0%	100%
Q3	Eu achei o sistema fácil de usar.	71,4%	28,6%	0%
Q4	Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	14,3%	14,3%	71,4%
Q5	Eu acho que as várias funções do sistema estão muito bem integradas.	85,7%	14,3%	0%
Q6	Eu acho que o sistema apresenta muita inconsistência.	0%	14,3%	85,7%
Q7	Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	85,7%	14,3%	0%
Q8	Eu achei o sistema atrapalhado de usar.	14,3%	0%	85,7%
Q9	Eu me senti confiante ao usar o sistema.	100%	0%	0%
Q10	Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	0%	0%	100%

Tabela 5.2: Respostas do questionário SUS: usuários comuns

Questão	Afirmação	Concorda	Neutro	Discorda
Q1	Eu acho que gostaria de usar esse sistema com frequência.	0%	28,6%	71,4%
Q2	Eu acho o sistema desnecessariamente complexo.	0%	0%	100%
Q3	Eu achei o sistema fácil de usar.	85,7%	0%	14,3%
Q4	Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	0%	0%	100%
Q5	Eu acho que as várias funções do sistema estão muito bem integradas.	71,4%	28,6%	0%
Q6	Eu acho que o sistema apresenta muita inconsistência.	0%	0%	100%
Q7	Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	85,7%	14,3%	0%
Q8	Eu achei o sistema atrapalhado de usar.	0%	0%	100%
Q9	Eu me senti confiante ao usar o sistema.	85,7%	14,3%	0%
Q10	Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	0%	0%	100%

Tabela 5.3: Respostas do questionário SUS: todos respondentes

Questão	Afirmação	Concorda	Neutro	Discorda
Q1	Eu acho que gostaria de usar esse sistema com frequência.	42,9%	21,4%	35,7%
Q2	Eu acho o sistema desnecessariamente complexo.	0%	0%	100%
Q3	Eu achei o sistema fácil de usar.	78,6%	14,3%	7,1%
Q4	Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	7,1%	7,1%	85,7%
Q5	Eu acho que as várias funções do sistema estão muito bem integradas.	78,6%	21,4%	0%
Q6	Eu acho que o sistema apresenta muita inconsistência.	0%	7,1%	92,9%
Q7	Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	85,7%	14,3%	0%
Q8	Eu achei o sistema atrapalhado de usar.	7,1%	0%	92,9%
Q9	Eu me senti confiante ao usar o sistema.	92,9%	7,1%	0%
Q10	Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	0%	0%	100%

6 CONCLUSÕES

O presente trabalho se propôs a desenvolver uma aplicação *web* para facilitar o processo de documentação e organização dos professores de Salas de Integração e Recursos, fornecendo uma solução com interface amigável, segura e compatível com a versão iOS já existente. A aplicação cumpre seu objetivo de universalizar o acesso ao sistema, visto que, atualmente, são raras as pessoas que não possuem acesso algum dispositivo com acesso à internet.

Desde sua definição, o principal foco do trabalho foi dar continuidade a um sistema integrado de gestão educacional. O sistema foi idealizado pelo professor coorientador Prof. Me. Francisco Dutra dos Santos Jr. que continuará conduzindo o projeto, com o objetivo principal de aprofundar a integração de sistemas que facilitem e qualifiquem a prática docente dos professores das Sala de Integração e Recursos .

De forma geral, os resultados alcançados foram satisfatórios e indicam que a aplicação é fácil de utilizar, intuitiva e consistente. O trabalho implementou as treze histórias de usuário priorizadas e a aplicação desenvolvida foi aprovada pelos professores de Salas de Integração e Recursos, obtendo uma pontuação no sistema de avaliação de usabilidade SUS de 89,2.

6.1 Limitações e Trabalhos Futuros

Apesar de ter seu objetivo atingido, o trabalho apresenta algumas limitações e ainda existe espaço para melhorias e trabalhos futuros. As limitações já identificadas na aplicação são:

- Falta de padronização entre os *timestamps* gerados pelas diferentes aplicações: As linguages JavaScript, usada na aplicação *web*, e Swift, na aplicação iOS, geram *timestamps* em formatos diferentes. Esta discrepância impossibilitou a integração completa dos bancos de dados;
- Falta de limitação dos tamanhos das mídias enviadas por usuários: na situação atual do sistema, os usuários podem enviar mídias sem restrição de tamanho. Com o tempo, isso pode se tornar custoso para armazenar no banco de dados;
- Inexistência de uma funcionalidade de recuperação e troca de senha: Caso um usuário venha a se esquecer de suas credenciais, a única forma de recuperá-las, na situ-

ação atual do sistema, é falando com um dos administradores do sistema.

Apesar da existência dessas limitações, o desempenho das funcionalidades implementadas na aplicação não é afetado. Além da resolução das limitações listadas acima, também existem trabalhos futuros importantes já identificados, como:

- Documentação técnica do código, para facilitar a manutenção e o entendimento por outros desenvolvedores eventualmente adicionados ao projeto;
- Trabalho de *UX/Design* em cima da aplicação para garantia da melhor experiência para os usuários;
- Inclusão de outros documentos como: Adequação Curricular, Plano Desenvolvimento Individual etc;
- Criação de perfis de usuário;
- Criação de uma página de administração, para controlar os professores cadastrados.

REFERÊNCIAS

ABRAMOV, D. **Mixins Are Dead. Long Live Composition**. 2015. <https://medium.com/reactbrasil/meu-primeiro-higher-order-component-a376efc654a8>.

BECK, K. **Planning Extreme Programming**. Addison-Wesley Professional, 2000. ISBN 0201710919. Disponível em: <https://www.xarg.org/ref/a/0201710919/>.

BERNARDO, K. **O que são métodos ágeis?** 2015. <https://www.culturaagil.com.br/o-que-sao-metodos-ageis/>.

BROOKE, J. **"SUS-A quick and dirty usability scale."Usability evaluation in industry**. CRC Press, 1996. ISBN: 9780748404605. Disponível em: <https://www.crcpress.com/product/isbn/9780748404605>.

BUNA, S. **The Complete Introduction to React**. 2019. <https://jscomplete.com/learn/complete-intro-react>.

CHACON, S.; STRAUB, B. **Pro git: Everything you need to know about Git**. Second. Apress, 2014. Disponível em: <https://git-scm.com/book/en/v2>.

CHRZANOWSKA, N. **12 Top Applications Written in Node.js - Examples from Big Companies**. 2017. <https://www.netguru.com/blog/top-companies-used-nodejs-production>.

COHN, M. Advantages of the “as a user, i want” user story template. **Mountaingoat Software. Retrieved on May**, v. 15, p. 2012, 2008. Disponível em: <https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template>.

COPES, F. **A developer’s introduction to React**. 2018. <https://jaxenter.com/introduction-react-147054.html>.

DEVATHON. **React vs. Angular: Which JavaScript framework should you use?** 2019. <https://devathon.com/blog/react-vs-or-angular-js-javascript-framework-use/>.

Encyclopaedia Britannica. **Client-server architecture**. 2019. <https://www.britannica.com/technology/client-server-architecture>.

Facebook Inc. **Components and Props**. 2019. <https://reactjs.org/docs/components-and-props.html>.

Facebook Inc. **Higher-Order Components**. 2019. <https://reactjs.org/docs/higher-order-components.html>.

Facebook Inc. **Introducing JSX**. 2019. <https://reactjs.org/docs/introducing-jsx.html>.

Facebook Inc. **React.Component**. 2019. <https://reactjs.org/docs/react-component.html>.

Facebook Inc. **State and Lifecycle**. 2019. <https://reactjs.org/docs/state-and-lifecycle.html>.

Facebook Inc. **Virtual DOM and Internals**. 2019. <https://reactjs.org/docs/faq-internals.html>.

FERREIRA, G. M. **SIR-EDU: Sistema Integrado de Recursos Educacionais para a Gestão do Acompanhamento de Alunos com Necessidades Especiais**. 2017. <https://lume.ufrgs.br/handle/10183/168934>.

FIREBASE, G. **Cloud Firestore**. 2019. <https://firebase.google.com/docs/firestore>.

FIREBASE, G. **Firebase**. 2019. <https://firebase.google.com/>.

FIREBASE, G. **Firebase Authentication**. 2019. <https://firebase.google.com/docs/auth>.

FREITAS, F. P. M.; SCHNECKENBERG, M. A gestão da educação especial inclusiva no brasil: uma análise histórica das constituições nacionais e leis de diretrizes e bases. **Imagens da Educação**, v. 4, n. 1, p. 64–76, 2014.

GARLAN, D.; SHAW, M. **An Introduction to Software Architecture**. Pittsburgh, PA, USA, 1994.

GeeksforGeeks. **React.js (Introduction and Working)**. 2019. <https://www.geeksforgeeks.org/react-js-introduction-working>.

GITHUB. **GitHub Hello World**. 2016. <https://guides.github.com/activities/hello-world/>.

JUNIOR, F. D. dos S. **As políticas de educação especial na Rede Municipal de Ensino de Porto Alegre : 1989-2000**. 2002. <https://lume.ufrgs.br/handle/10183/165434>.

LAMIM, J. **Afinal, o que é Frontend e o que é Backend?** 2014. <https://www.oficinadanet.com.br/post/13541-afinal-o-que-e-frontend-e-o-que-e-backend>.

LONGO, H. E. R.; SILVA, M. P. A utilização de histórias de usuários no levantamento de requisitos Ágeis. **IJKEM**, Florianópolis, SC, Brasil, v. 3, 2014. ISSN 2316-6517. Disponível em: <https://alged.webnode.com/_files/200000276-12d2f13c9e/HistoriaDeUsuarios_texto.pdf>.

LUCAS, E. A. **Sistema de Gestão e Acompanhamento Educacional**. 2018. <https://lume.ufrgs.br/handle/10183/190194>.

MALDONADO, L. **Entendendo o DOM (Document Object Model)**. 2018. <https://tableless.com.br/entendendo-o-dom-document-object-model/>.

MARUTA, R. **Meu primeiro Higher-Order Component**. 2018. <https://medium.com/reactbrasil/meu-primeiro-higher-order-component-a376efc654a8>.

MCKENZIE, S.; NAKAZAWA, C.; KYLE, J. **Yarn: A new package manager for JavaScript**. 2016. <https://engineering.fb.com/web/yarn-a-new-package-manager-for-javascript/>.

Ministério da Educação. **Diretrizes Operacionais da Educação Especial para o Atendimento Educacional Especializado na Educação Básica**. 2009. <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=428-diretrizes-publicacao&Itemid=30192>.

MOZILLA. **CSS: Cascading Style Sheets**. 2019. <https://developer.mozilla.org/en-US/docs/Web/CSS>.

MOZILLA. **HTML: Hypertext Markup Language**. 2019. <https://developer.mozilla.org/pt-BR/docs/Web/HTML>.

MOZILLA. **Introduction to the DOM**. 2019. <https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction>.

MOZILLA. **Polyfill**. 2019. <https://developer.mozilla.org/pt-BR/docs/Glossario/Polyfill>.

MUNDEL, C. F. **Modelagem do processo de atendimento em salas de recursos para alunos com necessidades educacionais especiais**. 2019. <https://lume.ufrgs.br/handle/10183/198550>.

Netscape. **Netscape and Sun Announce JavaScript, the Open, Cross-Platform Object Scripting Language for Enterprise Networks and the Internet**. 1995. <https://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html>.

Opus Software. **Node.js – O que é, como funciona e quais as vantagens**. 2018. <https://www.opus-software.com.br/node-js>.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide**. 2017. <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf>.

Scrum.org. **What is ScrumBut?** 2019. <https://www.scrum.org/resources/what-scrumbut>.

SILVA, A. R. da C. **SIGMA - Sistema de Gestão e Acompanhamento Móvel de Alunos Portadores de Necessidades Educacionais Especiais**. 2019. <https://lume.ufrgs.br/handle/10183/198601>.

Stack Overflow. **Developer Survey Results 2019**. 2019. <https://insights.stackoverflow.com/survey/2019>.

STORAGE, G. **Cloud Storage**. 2019. <https://firebase.google.com/docs/storage>.

STORI, D. **Primeiros passos em PaaS com Heroku**. 2013. <https://www.devmedia.com.br/primeiros-passos-em-paas-com-heroku/29465>.

TEIXEIRA, F. **O que é o SUS (System Usability Scale) e como usá-lo em seu site**. 2015. <https://brasil.uxdesign.cc/o-que-%C3%A9-o-sus-system-usability-scale-e-como-us%C3%A1-lo-em-seu-site-6d63224481c8>.

TRELLO. **O que é Trello?** 2017. <https://help.trello.com/article/708-what-is-trello>.

Usability.gov. **System Usability Scale (SUS)**. 2019. <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>.

WELLS, C. J. **A Brief History of CSS**. 2018. <http://www.technologyuk.net/computing/website-development/introduction-to-css/introduction.shtml>.

ÁGIL, D. **SCRUM**. 2014. <https://www.desenvolvimentoagil.com.br/scrum/>.