Webfolio - Portfólio Digital Seguro para a Educação de Infância

Willian de Oliveira Silva - a46695

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Rui Pedro Lopes

Prof. Arlete Teresinha Beuren

Prof. Cristina Mesquita

Bragança

2020-2021

Webfolio - Portfólio Digital Seguro para a Educação de Infância

Willian de Oliveira Silva - a46695

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Rui Pedro Lopes

Prof. Arlete Teresinha Beuren

Prof. Cristina Mesquita

Bragança

2020-2021

Dedicatória

Dedico este trabalho aos meus amados avós, em especial ao meu avô Aluízio Camões de Oliveira (in memoriam), maior exemplo de um ser humano íntegro, ético e gentil.

Agradecimentos

Agradeço aos meus pais Marta Nunes de Oliveira Silva e Elvio Denis Siva, e minha irmã Nathalia de Oliveira Silva que sempre estiveram ao meu lado me apoiando ao longo de toda a minha trajetória, e por todo o esforço investido na minha educação.

Sou grato pela confiança depositada na minha proposta de projeto pelo professor Rui Pedro Lopes e professora Arlete Teresinha Beuren, orientadores do meu trabalho.

Também agradeço aos meus amigos Gabriel Lenin Silva Lima e Jonathan Galdino da Silva que sempre me ajudaram com suas experiências desde o início deste projeto de pesquisa. Também quero agradecer a Martinha Silva Lima e Marta Nunes de Oliveira Silva que participaram dos testes realizados na aplicação.

Por último, quero agradecer também à Universidade Tecnológica Federal do Paraná e ao Instituto Politécnico de Bragança e todo o seu corpo docente.

Resumo

Na educação infantil há diversas técnicas para avaliar o desenvolvimento das crianças através de experiência e de seu conhecimento acumulado durante o semestre letivo. Uma dessas técnicas utilizadas para avaliar o desenvolvimento das crianças é por meio de Portfólios.

O uso do portfólio na educação infantil permite a avaliação formativa do aluno, possibilitando a reflexão por parte do aluno, dos professores e dos pais da criança a fim de compreender o processo escolar da criança.

Este trabalho tem como objetivo principal o desenvolvimento de uma aplicação *web*, que permite o suporte para a construção de uma plataforma de armazenamento e partilha de portfólios digitais para a educação infantil. Além disso, pretende-se que seja feito um controle de acesso dos utilizadores para que seja possível o gerenciamento de acessos a determinados portfólios.

Para o desenvolvimento da aplicação, foram utilizadas tecnologias como TypeScript, React, Node.js, MongoDB e Redis. O sistema deve possuir funcionalidades como: criar portfólios, criar observações e gerenciar permissões de utilizador.

Após a implementação da aplicação, foram realizados testes de usabilidade no sistema, com pessoas que trabalham na área de educação infantil. Posteriormente, foram implementadas sugestões de melhoria para experiência do utilizador na aplicação.

Palavras-chave: portfólio, observação, interface, desenvolvimento *web*, API.

Abstract

In early childhood education there are several techniques to assess children's development through experience and their knowledge accumulated during the school semester. One of these techniques used to assess children's development is through Portfolios.

The use of the portfolio in early childhood education allows the formative assessment of the student, allowing a reflection on the part of the student, teachers and the child's parents in order to understand the child's school process.

This work has as main objective the development of a web application, which allows support for the construction of a platform for the storage and sharing of digital portfolios for early childhood education. In addition, it is intended that an access control of users be made so that it is possible to manage access to certain portfolios.

For the development of the application, technologies such as Typescript, React, Node.js, MongoDB and Redis were used. The system must have characteristics such as: creating portfolios, creating and managing the user.

After the implementation of the application, usability tests were carried out on the system, with people working in the field of early childhood education. Subsequently, suggestions for improving the user experience in the application were implemented.

Keywords: portfolio, observation, interface, development web, API.

Conteúdo

Lista de Figuras

Siglas

API *Application Programming Interface.* xi, 8, 10, 11, 13, 17, 39, 41, 66

BSON JSON *Binary.* 17

CSS *Cascading Style Sheets.* 5, 7, 18

DDoS *Distributed Denial of Service.* 40

DOM *Document Object Model.* 8, 13

ECMA *European Computer Manufacturer's Association.* 7

HATEOAS *Hypermedia as the Engine of Application State.* 13

HTML *HyperText Markup Language.* xi, 5, 7–9, 13, 18, 41

HTTP *HyperText Transfer Protocol.* 10–12, 14, 19

IP *Internet Protocol.* 40

JSON JavaScript *Object Notation.* ix, xi, 13, 14, 17, 19

JWT JSON *Web Token.* 5, 19, 39, 67

RBAC *Role-Based Access Control.* 5, 20

REST *Representational State Transfer.* ix, 8, 10

SPA *Single-Page Web Application.* 18

SQL *Structured Query Language.* xi, 16, 17, 67

UML *Unified Markup Language.* 21

URI *Uniform Resource Identifier.* 10, 11, 13

URL *Uniform Resource Locator.* 11, 67

WWW *World Wide Web.* xi, 11

XML *Extensible Markup Language.* ix, 8, 13, 14

Capítulo 1

Introdução

Na educação infantil há diversas técnicas para registar, estudar e perceber o desenvolvimento das crianças. Uma dessas técnicas baseia-se na informação recolhida durante as experiências de aprendizagem, armazenada em portfólios. O portfólio designa uma coleção de itens e informação, que ilustram e representam os diferentes aspetos do desenvolvimento da criança [1].

O uso do portfólio na educação infantil sustenta a formação da criança, possibilitando a reflexão delas próprias, dos professores e dos pais, ajudando-os a compreender o processo de desenvolvimento e permitir manter um registo que pode servir de base à continuidade do trabalho mesmo que por outro professor [2].

Tipicamente, os portfólios são arquivos físicos de objetos, incluindo texto, imagens, colagens, fotografias ou vídeos. Estes, devido às suas características, não permite um acesso fácil aos pais ou à comunidade, pois seria necessário estar fisicamente próximo para o poder explorar e compreender. Uma alternativa passa por recorrer a portfólios digitais. De igual forma, estes permitem armazenar observações, comentários, vídeos, fotografias e trabalhos das crianças, como base para a documentação pedagógica. Devido às suas características, o uso de portfólios digitais está crescendo, porém o seu uso possui limitações de recursos, sendo difícil o acesso a uma ferramenta totalmente gratuita. Desta forma, o desenvolvimento de uma plataforma segura, de ampla utilização e gratuita para a comunidade nacional dos educadores de infância é de grande relevância.

1.1 Enquadramento

Neste trabalho descreve-se o desenvolvimento de uma aplicação *web* com o propósito de providenciar uma ferramenta de armazenamento e partilha de observações, comentários, vídeos, fotografias e trabalhos das crianças, no âmbito dos principais atores, nomeadamente, crianças, auxiliares, educadores e pais. O conjunto destas informações colhidas constitui o portfólio digital. Tendo em conta que algumas informações armazenadas são sensíveis e sujeitas a rigorosas normas de privacidade, será necessário definir uma arquitetura de segurança e privacidade, incluindo identificação, autenticação e autorização dos utilizadores.

Assim, será necessário fazer o levantamento do estado da arte, com apoio de bibliografia científica relevante, para construir o enquadramento e os requisitos funcionais e não funcionais da aplicação. Adicionalmente, também deverá ser definido as ferramentas, *frameworks* e opções principais que poderão ser adotadas no desenvolvimento da aplicação.

1.2 Objetivos

O objetivo deste trabalho é desenvolver uma aplicação *web* para suporte à construção de uma plataforma de armazenamento e partilha de portfólios digitais para educação de infância.

1.3 Plano de Trabalho

A Figura 1.1 apresenta o cronograma do projeto com todas as suas tarefas e o tempo de duração de cada uma delas.

1.4 Estrutura do Documento

O documento está estruturado de forma a contemplar todas as fases do desenvolvimento da solução implementada. No Capítulo 2 são abordadas as tecnologias a serem usadas

2020/2021	Meses											
Tarefas	Jun	Jul	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai
Enquadrar e contextualizar a aplicação												
Desenvolver a análise do sistema												
Implementar a aplicação recorrendo a frameworks e linguagens adequadas												
Testar a aplicação em ambiente real												
Escrita da dissertação												

Figura 1.1: Cronograma do Projeto

no desenvolvimento e os motivos pelas quais elas foram escolhidas. No Capítulo 3 estão descritos a modelagem e o desenvolvimento da aplicação. No Capítulo 4 são descritos o desenvolvimento e soluções técnicas aplicadas durante o desenvolvimento do trabalho. O Capítulo 5 aborda questões relacionadas aos testes realizados e discussões. Por final, o Capítulo 6 aborda as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Contexto e Tecnologias

Neste Capítulo são apresentados as ferramentas e tecnologias utilizadas para o desenvolvimento do trabalho. Considerando que a proposta original do trabalho é desenvolver uma ferramenta *web*, as tecnologias a seguir estão relacionadas ao desenvolvimento *web*. Em resumo, para a criação do *back-end*, são utilizadas tecnologias como JavaScript, Node.js e Express. Enquanto que para a criação do *front-end*, são utilizadas tecnologias como HTML, *Cascading Style Sheets* (CSS), JavaScript, React e TypeScript. Por fim, para a base de dados é utilizado o MongoDB, para o armazenamento das informações da aplicação. Todas essas tecnologias estão alinhadas com a utilização de JSON *Web Token* (JWT) e *Role-Based Access Control* (RBAC) para garantir a integridade e a segurança do sistema. Para o controle e versionamento de código é utilizado git com a hospedagem na plataforma GitHub.

2.1 Contexto

Na abordagem de Mesquita [3], é apresentado um método para registrar observações de acompanhamento infantil em um aplicativo para telemóvel. Esta observação pode armazenar dados como textos, imagens, vídeos e áudios, contribuindo para criar um portfólio eletrônico, reduzindo deste modo o tempo criando diversos formulários em papel.

ChildDiary [4] desenvolveram uma plataforma para criar portfólios, observações, registro de rotinas diárias, gestão de crianças em lista de espera, envolvimento parental por meio de mensagens individuais e partilha de documentos e convites para reuniões. O sistema possui acesso por meio de computador, *tablet* e telemóvel, porém sua utilização gratuita é limitada. Funcionalidades como o número de educadores que podem acessar o sistema, número máximo de crianças, relatórios e visibilidade para Direção ou Coordenação são exemplos de recursos limitados no sistema. Deste modo existe a necessidade do usuário entrar em contato com os criadores da plataforma para receber um orçamento e implementar a ferramenta nomeada como ChildDiary na instituição de ensino.

Na aplicação SeeSaw [5], é possível criar portfólios digitais, contendo textos, imagens, áudios, além de permitir exportar o arquivo com o formato `PDF`. O portfólio pode ser usado por alunos para documentar de forma autônoma o que estão a aprender, além de poder partilhar o portfólio com os seus professores, pais ou colegas. Porém seu uso gratuito tem restrições, limitando recursos como atividades de classe por professor, criação de múltiplos *posters*, salvar trabalho como rascunho, criação e compartilhamento de atividades e portfólios de acompanhamento de série em série dos alunos, porém a plataforma é inteiramente construída em linguagem inglesa, não possuindo nenhuma tradução para o seu uso em outros idiomas.

Em outra abordagem, a plataforma EduClipper [6] pode ser acessada em *web* ou por meio de download no aplicativo telemóvel iOS. Na plataforma é possível criar e compartilhar conteúdo dinâmico, além de permitir obter *feedback* sobre as tarefas das crianças. Também é possível capturar e acompanhar o seu desenvolvimento e desempenho ao longo do tempo em portfólios de aprendizagem digital. A aplicação é gratuita para uso e não possui planos pagos.

O Three Ring é uma aplicação para telemóvel Android ou iOS, permitindo documentar atividades de sala de aula, criar portfólios multimídia e compartilhar com pais, administradores e outros professores [7]. Na aplicação é possível capturar fotos, vídeos, áudios e notas da aula, além de contar com armazenamento ilimitado. O uso da plataforma é gratuito.

O Easy Portfólio é uma aplicação para telemóvel iOS e Mobile, que não possui versão gratuita [8]. Na ferramenta os alunos e professores podem capturar e compartilhar seus trabalhos, como fotos, vídeos, gravações de áudios e músicas, endereços da *web*, documentos digitais criando um portfólio digital.

2.2 Tecnologias

Para dar corpo à aplicação, serão usadas ferramentas e tecnologias adequadas à programação e desenvolvimento de aplicações *online*, acedidas pela Internet por intermédio de um navegador.

2.2.1 JavaScript

Segundo Flanagan [9], o JavaScript é uma linguagem de programação *web* utilizado na maioria dos sites e por todos os navegadores modernos, como computadores de mesa, console de jogos, *tablets* e *smartphones*, tornando a linguagem de programação universal. O JavaScript é usado para especificar o comportamento de páginas e é geralmente utilizado em conjunto com tecnologias como HTML, que é utilizado para especificar o conteúdo das páginas, e CSS, utilizado para especificar a apresentação destas páginas.

O JavaScript foi criada pela Netscape em parceria com a Sun Microsystems, com o objetivo de criar interatividade para páginas *web*, com sua primeira versão introduzida no navegador Netscape 2.0 em 1995 [10]. Através do uso do JavaScript é possível, alterar e controlar de forma dinâmica a apresentação de um documento HTML, além de alterar aspectos como cor de fundo, textos, links, criar janelas *pop-ups*, apresentar mensagens ao usuário, interferir na barra de status, retirar menus, fechar e abrir janelas, também sendo possível manipular a folha de estilos do documento criando novas regras CSS.

A Netscape enviou a linguagem para a *European Computer Manufacturer's Association* (ECMA) para padronização e, após isso, a versão padronizada passou a chamar-se, formalmente, ECMAScript. Todos os navegadores *web* implementam a versão 3 e 5 do padrão ECMAScript [9].

O ECMAScript 6, também conhecido como “ECMAScript 2015” ou “E6”, foi implementado em 2015, visando apresentar mudanças para resolver problemas de desenvolvedores e adicionar benefícios, como o uso de declaração de variáveis como *let* e *const*, o uso da Programação Funcional com *arrow functions*, facilidade de trabalhar com módulos com o uso de *export/import*, além de sintaxe para uso de criação de linguagens específicas de domínios, *template literals* [11]. Muitos dos *browsers* não possuem suporte ao ES6, sendo necessário o uso de um transpilador, como o Babel, para transformar o código ES6 para ES5.

2.2.2 *Document Object Model*

O JavaScript pode transformar documentos HTML estáticos em aplicativos *web* interativos, através da manipulação da *Document Object Model* (DOM) intrínseca aos documentos HTML e XML [9]. A DOM pode ser representada como uma árvore de objetos, que possui nós representando marcações ou elementos HTML, como “<body>” e “<e>” representando nós de strings de texto [9].

A DOM especifica como os navegadores devem criar um modelo de página HTML e como o JavaScript pode acessar e alterar conteúdos desta página *web* enquanto ele está na janela do navegador [12]. A DOM na verdade é uma API, interface de programação de aplicativo que permite interação humano com programas, e sua representação como uma árvore de objetos pode ser vista na Figura 2.1.

2.2.3 Arquitetura REST

As APIs podem ser entendidas como uma maneira de dois aplicativos de computador se comunicarem um com outro usando uma linguagem comum entre eles [13]. As APIs também podem conter especificações, na qual o provedor descreve as funcionalidades ela fornece, quando as funcionalidades devem estar disponíveis, quais as suas restrições técnicas, como permissão para usar somente em determinada hora, e de poder restringir tipos de uso [13]. Além disso, a API pode fornecer ferramentas como mecanismos para

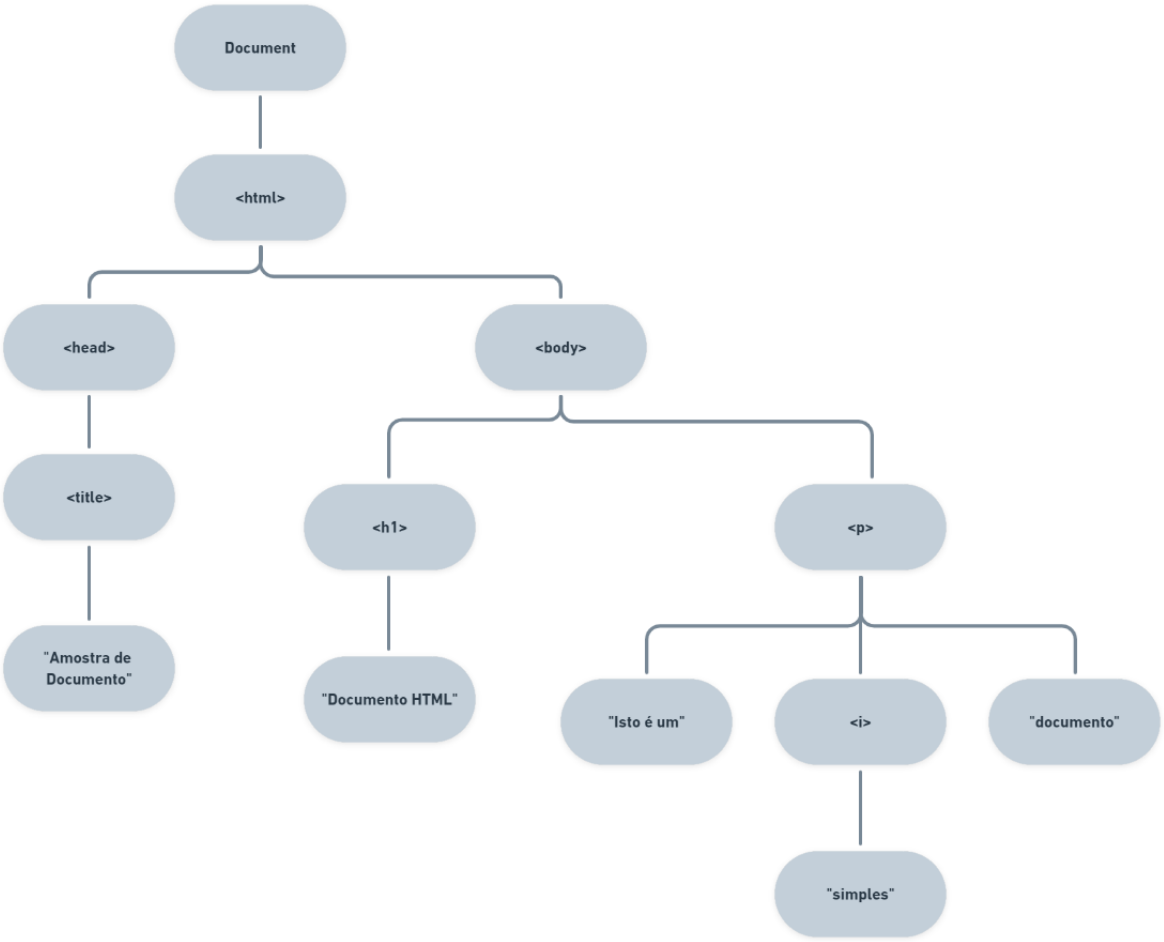

Figura 2.1: A representação em árvore de um documento HTML, adaptado de [9].

acessar os seus termos de uso, mecanismos para acessar a documentação da API, recursos de exemplos e informações operacionais sobre a integridade da API enquanto em uso, podendo assim, considerar uma API como um produto de Software [13].

Existem dois tipos de APIs, nomeadamente públicas e privadas [13]. A maior parte são privadas, pois elas necessitam ser usadas somente por equipes e parceiros por meio de acordos contratuais, enquanto as públicas são disponíveis para qualquer pessoa com pouco ou nenhum acordo contratual [13]. Além disso, a API pode ser criada de modo privado e depois se tornar público, ou vice-versa.

No ano de 2000, Roy Fielding desenvolveu em sua dissertação para Doutorado a descrita como REST, propondo usar *HyperText Transfer Protocol* (HTTP) para comunicação entre computadores [14]. O estilo REST é atualmente uma das formas mais populares para desenvolver APIs. REST é baseado no padrão HTTP e usa blocos de construção de serviços, dividindo o *namespace* em um conjunto de "recursos" com base no padrão *Uniform Resource Identifier* (URI) e usa verbos HTTP padrão como GET, POST, PUT e DELETE para mapear operações.

O padrão HTTP define alguns tipos diferentes de mensagens, nomeadamente GET, que obtêm um recurso, DELETE, para destruir um recurso, POST, criar um novo recurso com base na informação fornecida, PUT, Substitui um recurso com base na informação fornecida, e PATCH, que funciona como o PUT, porém permite a mudança em uma informação específica de um recurso, como a imagem de avatar de um utilizador [15].

De acordo com Masse [14], as APIs REST são servidores *web* especialmente desenvolvidos para oferecer suporte às necessidades de um site ou qualquer outro aplicativo. De forma geral, a API possui um conjunto de dados e funções que facilitam a troca de informações entre computadores, a API REST responde diretamente às solicitações do cliente [14], como pode ser visto na Figura 2.2.

As APIs REST são projetadas para atrair desenvolvedores de clientes a usar serviços da *web* "RESTful". Para uma API ser considerada RESTful, ela precisa ter três níveis distintos de maturidade, que são a URI, HTTP e Hipermídia [16]. A Hipermídia são dados enviado do servidor para o cliente para informar ao cliente quais solicitações HTTP

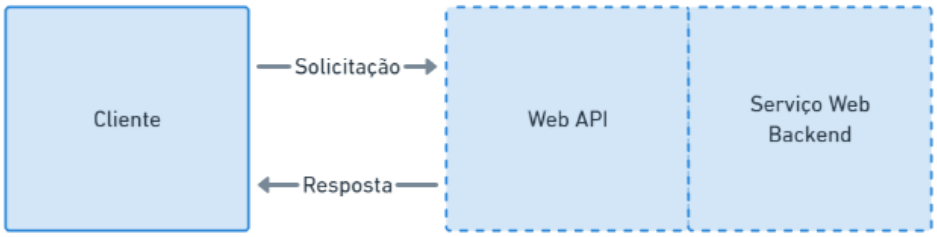

Figura 2.2: *Web* API, adaptado de [14].

ele pode fazer a seguir, enquanto a URI é um nome e endereço que identifica um recurso e o HTTP é um protocolo baseado em documento em que o cliente coloca um documento em um envelope e o envia para o servidor, informando ao servidor o que o cliente quer fazer com um recurso [15].

Em um WWW ou em qualquer API RESTful, as *Uniform Resource Locators* (URLs) estão na parte inferior, identificando recursos, enquanto o protocolo HTTP está acima fornecendo acesso de leitura às representações e acesso de gravação ao estado de recurso. Por fim, a *Hypermedia* (Hipermídia) fica acima do HTTP, descrevendo a semântica do protocolo do site ou da API (Figura 2.3).

Figura 2.3: Pilha de tecnologia que forma a WWW, adaptado de [15].

Em 1993, Roy Fielding preocupado com problemas de escalabilidade da *web*, propôs

junto com sua equipe, uma implementação da *web* que fosse capaz de satisfazer uniformemente todas as restrições para que a *web* pudesse se expandir. Agrupou essas restrições em seis categorias e as referiu como o estilo arquitetônico RESTful [14]. Essas restrições arquitetônicas, ou também conhecidos como princípios, são "*Client-Server*", "*Stateless*", "*Cache*", "*Layered System*", "*Code on Demand*" e "*Uniform Interface*", detalhadas a seguir.

Client-Server é a arquitetura de rede mais encontrada na internet. Em seu funcionamento o componente cliente envia uma solicitação ao servidor através de um conector desejando que um serviço seja executado, o servidor por sua vez rejeita ou executa a solicitação e envia uma resposta de volta ao cliente [15]. O *Client-Server* pode ser implementado e inserido de forma independente, utilizando qualquer linguagem ou tecnologia, desde que esteja em conformidade com a interface uniforme da *web* [14].

Stateless tem como objetivo melhorar a escalabilidade do servidor, excluindo a necessidade do servidor manter consciência do estado do cliente além de sua solicitação atual, todo o estado do aplicativo, informações o caminho de um cliente específico por meio do aplicativo deve pertencer ao cliente [15].

Caching diz respeito aos dispositivos HTTP que possuem armazenamento automático de cópias de documentos populares, podendo reduzir as transferências de dados redundantes, economizando tempo para que as páginas carreguem mais rapidamente, economizando dinheiro em tarifas de rede, além de reduzir a demanda nos servidores de origem [17]. O cache é uma das restrições mais importantes da arquitetura, pois o armazenamento em cache dos dados de resposta pode ajudar a reduzir o custo geral da *web* e aumentar a disponibilidade geral e confiabilidade de um aplicativo, além de controlar a carga de um servidor *web* [14].

Uniform Interface refere-se à interação entre clientes e recursos mediadas por métodos HTTP básicos como GET, POST, PUT, DELETE [16]. As interações entre clientes, servidores e intermediários baseados em rede dependem da uniformidade de suas interfaces, se houver algum desvio deste padrão a comunicação *web* falha [14]. Essa restrição, por sua vez, detalha quatro adicionais, que são a identificação de recursos, como por exemplo

URI, a manipulação de recursos por meio de representações, que é uma forma de interagir com o recursos para representá-los de formas e formatos diferentes como HTML para um navegador da *web* e como JSON para um programa automatizado, sem alterar seu identificador, outras restrições são as de mensagens auto-descritivas, que transmitem detalhes adicionais sobre o estado do recurso, sugerindo por exemplo uma atualização de página, e por fim a restrição *Hypermedia as the Engine of Application State* (HATEOAS), que se refere à prática de fornecer uma lista de links com reconhecimento de estado para transmitir as ações disponíveis e permitir que o usuário percorra informações e aplicações de maneira direcionada [14].

Layered System são geralmente usados para segurança da aplicação. Ela adiciona intermediários baseados em rede como *proxies* e *gateways*, os implantando de forma transparente entre um cliente e servidor usando o interface uniforme, permitindo que os intermediários interceptar uma comunicação cliente-servidor para um propósito específico [14].

Code on Demand permite que servidores da *web* possam transferir temporariamente programas executáveis para clientes, ele estabelece um acoplamento entre tecnologias de servidores *web* e seus clientes, para que o cliente seja capaz de entender e executar o código ou download sob demanda do servidor, esta é a única restrição na arquitetura que é considerada como opcional [14].

2.2.4 XML ou JSON

A maioria das APIs aceitam solicitações PUT ou POST contendo dados formatados em XML ou JSON [13]. XML é similar ao HTML, porém os nomes das suas *tags* são diferentes, pois descrevem os dados que contém, sua sintaxe é mais restrita que o HTML e costuma funcionar melhor com diferentes plataformas e aplicativos, além de ser processado com os mesmos métodos DOM do HTML [12].

O JSON é um formato de texto padronizado e usado para troca de dados estruturados [16]. Este possui sintaxe semelhante a notação literal de objeto para representar um dado,

pode ser chamado de qualquer domínio e é menos prolixo (verboso) que o XML, além de ser normalmente utilizado com JavaScript [12]. Os dados JSON se parecem com código JavaScript ou Python, impondo restrições simples aos textos. Para um JSON ser válido, uma *string* deve estar entre aspas duplas [15]. Um exemplo de dados JSON pode ser visto na Figura 2.4.

Figura 2.4: Dados em formato JSON.

O navegador obtém mais facilmente uma estrutura de dados JavaScript de uma estrutura de dados JSON do que um documento XML, pois o navegador fornece uma interface JavaScript diferente para um analisador XML, enquanto uma *string* JSON é um programa JavaScript fortemente restrito, que funciona da mesma forma em qualquer navegador. Com o crescimento da *web* e o HTTP como plataforma, o uso do JavaScript também cresceu, sendo hoje possível encontrar JavaScript como linguagem de programação de navegadores *web*, servidores como Node.js, além de armazenamento de dados [18].

2.2.5 Node.js

Para suprir necessidades para usar o máximo o poder do processador, enquanto não mantém ocioso tarefas quando o mesmo realiza tarefas do tipo bloqueante, Ryan Dahl, no ano de 2009, com a ajuda de seus colaboradores, criaram o Node.js [19].

A arquitetura Node.js é *non-blocking thread* (não bloqueante), possuindo boa performance e consumo de memória, não sendo sujeito a *deadlocks* no sistema, além de ser uma plataforma escalável e de baixo nível, pois é possível programar com diversos protocolos de rede de internet ou utilizar bibliotecas que acessam recursos do sistema operacional. Além disso o Node.js é *single threaded*, pois cada aplicação possui a sua instância de um único processo, porém é possível usar programação assíncrona para executar funções em

background, em paralelo, e aguarda retorno através de funções como *callback* do JavaScript [19].

O Node.js é orientado a eventos, seguindo a mesma orientação de eventos do JavaScript *Client-side* (lado do cliente), trabalhando com eventos de *Input/output* (Entrada/Saída), como o evento de conectar um banco de dados ou abrir um arquivo.

Existem linguagens de programação do lado do servidor como PHP, Ruby, Python, ASP, ou Java. Contudo, o Node.js usou o V8, um poderoso motor JavaScript do Google Chrome, que pode executar milhares de instruções por segundo, para rodar em servidores e usar JavaScript para desenvolver aplicativos do lado do servidor, compartilhar código entre navegador servidor, passando a ser um dos mais populares *frameworks* do lado do servidor [20]. Através do Node.js foi possível utilizar diversas tecnologias totalmente em JavaScript como MongoDB, um banco de dados controlado por JavaScript, e Express, React.js, Vue.js, e Angular, que são *frameworks* de *front-end* JavaScript, facilitando na comunicação entre o navegador e servidor [20].

2.2.6 Express

Segundo [21], *frameworks* são uma forma prática de expressar design reutilizáveis. Os *frameworks* são de grande porte e possuem subsistemas que capturam a estrutura global de um projeto de sistema de software, oferecendo uma melhora significativa em desenvolvimento de software [21].

Para Roberts [22], *frameworks* são design reutilizáveis que podem ser parte de um sistema de software ou ser todo um sistema de software, como um conjunto de classes abstratas, que podem reduzir o custo de desenvolvimento de uma aplicação, pois permite reutilizar o design e o código, podendo ser implementados com linguagens de programação orientadas a objetos existentes.

framework para aplicativo da *web* pode fornecer funcionalidades para outros aplicativos da *web* e não somente apresentar uma coleção estática de conteúdo [23].

O Express é um *framework* mínimo e flexível para aplicativo da *web* do Node.js, que

fornece um conjunto robusto de recursos para a construção de aplicativos da *web* híbridos e com várias páginas [24].

Para Brown [23], o aspecto mínimo, está relacionado a filosofia do Express de fornecer a camada mínima entre seu cérebro e o servidor, permitindo expressão plena de suas ideias ao mesmo tempo que fornece algo útil.

O aspecto de flexível está relacionado com a filosofia do Express que você pode adicionar diferentes partes da funcionalidade conforme necessário, permitindo que você adicione o que você precisa quando precisar, reduzindo o desperdício de tempo, removendo e substituindo funcionalidades que não atendem aos requisitos [23].

2.2.7 *Data Management*

Os sistemas de gerenciamento de banco de dados são softwares para aplicativos que armazenam, descrevem e consultam dados de forma independente [25]. Todos os sistemas de gerenciamento de banco de dados possuem um componente de gerenciamento, que é responsável por gerenciar permissões de acesso, edição para os usuários, e inclui todos os dados armazenados em uma forma organizada para posterior consulta e manipulação de dados para avaliar ou editar os dados e informações.

Existem bancos de dados de linguagem de consulta relacional, como o SQL, que é uma linguagem descritiva, na qual suas consultas seguem um padrão básico, como o apresentado na Figura 2.5, e que pode ser aplicada em um ou várias tabelas para gerar uma tabela como resultado. Geralmente os bancos de dados SQL são projetados para integridade e proteção de de transações, exigido em aplicativos bancários ou software de seguros, entre outros [25]. Uma das desvantagens do uso de banco de dados relacionais é o seu controle de integridade de dados, que requer muito trabalho e poder de processamento, causando desvantagem na eficiência e desempenho, além perda na flexibilidade no processamento de dados [25].

Existem outras abordagens de banco de dados como o NoSQL, termo primeira vez

```
select * from Apps where AppName = 'Control'
```

Figura 2.5: Exemplo de uma Query SQL.

usado em 1998 para indicar o gerenciamento de dados não relacionais, que possui características diferentes do modelo SQL, pois seus dados não são armazenados em tabelas e a linguagem do banco de dados não é SQL.

O NoSQL possui foco em escalabilidade, restrições fracas ou nenhuma restrição de esquema, replicação fácil de dados e acesso fácil e fornecido por meio de APIs. Seus dados são armazenados em pares de valores-chave, colunas ou família de colunas, armazéns de documentos ou bancos de dados gráficos para posteriormente serem retornados através de um método de mapeamento, na qual as subtarefas retornam vários nós de computador e pares de valores-chave simples, que são extraídos e então agregados para ser retornado [25].

O MongoDB é um banco de dados NoSQL, que armazena informações na forma de coleções e documentos, em que cada coleção representa uma única entidade no aplicativo [20]. Uma coleção contém documentos, e um documento é uma instância da entidade que contém vários campos para representar o documento [20].

O MongoDB é um banco de dados poderoso, flexível e escalonável, que combina a capacidade de escalar horizontalmente com recursos como índices secundários, consultas, de intervalo, classificação, agregações e índice geoespaciais [26]. Os documentos modelados no MongoDB são do formato JSON, porém são armazenados em formato JSON *Binary* (BSON), indicando que o documento é um dicionário de pares de valores-chave, onde o valor pode ser nulo, matrizes de valor, objetos compostos de pares valores-chave, tipos primitivos de JSON, como número, string, booleano, além de tipos BSON primitivos, como datetime, ObjectId, UUID e regex [27].

É uma abordagem orientada a documentos de fácil uso que torna possível representar relacionamentos hierárquicos complexos com um único registro. Seus esquemas não são pré definidos, permitindo que chaves e os valores de um documento não tenham formato ou tamanhos fixos, tornando mais fácil o adicionar ou remover campos conforme necessário,

o que torna o desenvolvimento mais rápido para os desenvolvedores de Software [26]. Para haver comunicação entre MongoDB e Node.js, é necessário o uso de uma biblioteca chamada Mongoose [20].

2.2.8 React

De acordo com [28], o React é uma biblioteca popular, criada pela equipe Facebook, para criar interfaces de usuário e lidar com desafios associados a sites de grande escala baseados em dados. O React introduz novos paradigmas e muda status necessários para criar aplicativos e interfaces de usuário JavaScript escaláveis e sustentáveis, além de poder compor aplicativos da *web* de uma única página (*Single-Page Web Applications* (SPAs)) [29]. No React o código escrito se parece com HTML, e ele pode utilizar a sintaxe mais recente do ECMAScript [28].

React cria componentes que parecem ser mais concisa que outros *frameworks* como Ember.Js, ou Angularjs por que lidam com menor número de *tags* de *script* e arquivos, além de permitir uma sintaxe mais clara ao ler e atualizar o código.

Um aplicativo do lado do cliente (*client-side*) renderiza a maior parte de sua interface de usuário de um pacote de aplicativo inicial que é enviado apenas uma vez, portanto, uma vez que o navegador recebe o HTML inicial, ele usa JavaScript para modificar e não precisar depender do servidor para exibir novas páginas [23] . Os aplicativos do lado do cliente são geralmente chamados de SPAs, enquanto um aplicativo do lado do servidor (*server-side application*) é aquele em que as páginas do aplicativo são renderizados no servidor (como HTML, CSS e JavaScript) e é enviado ao cliente.

Os aplicativos da *web* de várias páginas (*Multipage and hybrid web applications*) são uma abordagem tradicional usadas por sites, em que cada página em um site é fornecida através de uma solicitação separada ao servidor, diferente dos SPAs, que permitem o *download* do site inteiro para o navegador do cliente, e após isso, a navegação se torna mais rápida porque há pouca ou nenhuma comunicação com o servidor, facilitando o uso e o serviço para *frameworks* como Angular e React.js [23].

2.2.9 TypeScript

O TypeScript é uma linguagem de código aberto baseado em JavaScript, que adiciona funções de tipo estático ao JavaScript [30]. O TypeScript é um superconjunto da linguagem JavaScript focado na produção de código seguro e previsível, com recursos como digitação estática, que torna a o trabalho JavaScript mais previsível para programadores familiarizados com linguagens como C# e Java [31].

O TypeScript fornece mensagens de erro úteis em seu editor. Enquanto digita, o TypeScript irá detectar os erros e avisar sobre eles, desta forma, reduz o uso de testes de unidades ou demais testes para identificação de erros [32].

O TypeScript se torna muito útil com o React, pois é possível usá-lo para definir interfaces de seus componentes, para que seja possível certeza de que o componente possui entradas corretas [33].

2.2.10 *JSON Web Token*

Para que informações de identidade e segurança sejam compartilhadas entre domínios de segurança e também ser usado como um mecanismo de autenticação de cliente, o JWT é uma codificação de *token* de segurança baseada em JSON capaz de promover este tipo de segurança [34]. O OAuth 2.0 Authorization Framework é um *framework* que fornece um método para fazer solicitações HTTP autenticadas a um recurso usando um *token* de acesso [35]. Um *token* de acesso é emitido para clientes por um servidor de autorização com a aprovação do proprietário do recurso [35].

De acordo com Hardt [35], o OAuth possui um mecanismo de extensibilidade que permite que você defina novos tipos de concessões de extensão para oferecer suporte a clientes adicionais para servir como ponte entre o OAuth e outras estruturas de protocolos. Desta forma um tipo de concessão de extensão que usa JWT Bearer Token pode ser usado para solicitar um *token* de acesso OAuth 2.0 quando um cliente deseja utilizar um recurso existente, expressa através da semântica do JWT, ou para uso como credenciais de cliente [34].

2.2.11 *Role-Based Access Control*

RBAC é uma tecnologia que reduz a complexidade e o custo da administração de segurança em aplicativos de rede, usando funções, hierarquias e restrições para organizar privilégios, fornecendo uma maneira segura e eficaz de gerenciar o acesso a informações de uma organização. O controle de acesso é aplicado sempre que um usuário faz logon em um sistema de computador multiusuário, e seu uso é essencial para preservar a confiabilidade, na qual exige que apenas usuários autorizados possam ler informações, e integridade das informações, na qual exige que apenas usuários autorizados possam alterar informações de maneiras autorizadas [36].

Capítulo 3

Análise

Neste capítulo são apresentado a análise e os diagramas *Unified Markup Language* (UML) associados, bem como a descrição de todas as tarefas desempenhadas pelo sistema e suas permissões.

3.1 Casos de Uso

Nesta seção é apresentado o diagrama de casos de uso geral e a descrição de cada (Figura 3.1).

O caso de uso "Realizar Login" permite que qualquer utilizador possa acessar o sistema. Para realizar o *login* é necessário inserir o e-mail e senha do utilizador. Em seguida, é verificado se os dados inseridos estão registrados na base de dados da aplicação, sendo que, caso contrário, o *login* não é realizado.

O caso de uso "Criar Conta" pode ser realizado por qualquer utilizador para criar uma conta na aplicação. O utilizador deve fornecer as informações de e-mail, nome, senha e instituição de ensino a qual pertence. Caso o e-mail informado já possua registro na aplicação, a criação de conta não é realizada.

O caso de uso "Recuperar Senha" permite a qualquer utilizador que esqueceu a sua senha, recuperar e alterar sua senha para aceder ao sistema. O utilizador deve inserir o e-mail para que uma mensagem de recuperação de senha seja enviado para o seu e-mail,

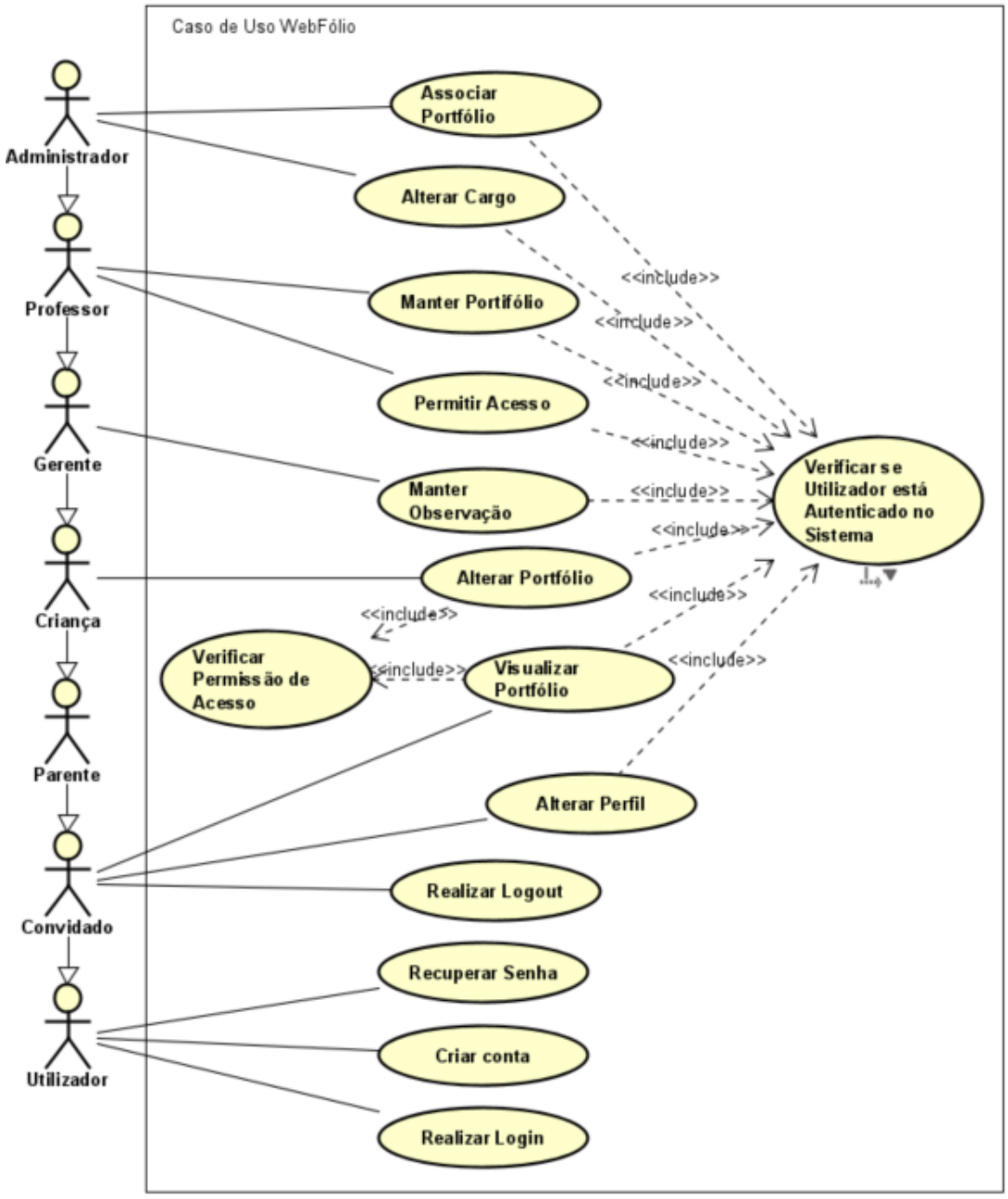

Figura 3.1: Diagrama de Casos de Uso Geral.

após isso, é possível a alteração de sua senha para aceder novamente na aplicação.

O caso de uso "Realizar Logout" está associado a todos os utilizadores que acederam na aplicação, que são considerados como atores do tipo "convidado". Este caso de uso permite que o utilizador possa sair do sistema. Para realizar o *logout* o utilizador deve selecionar a ação para sair do sistema. Caso o utilizador não faça o *logout*, ele será automaticamente redirecionado para a página principal quando acessar o endereço da aplicação.

O caso de uso "Alterar Perfil" está associado a qualquer utilizador logado no sistema. Este caso de uso permite que o utilizador realize alterações em seu perfil, podendo alterar informações como nome, e-mail, instituição e sua foto de perfil. Além disso é possível alterar a senha, o utilizador deve inserir a senha antiga e a nova senha. Caso o e-mail informado já possua registro na aplicação, a alteração de perfil não é realizada.

O caso de uso "Visualizar Portfólio" está disponível para qualquer utilizador no sistema, desde que tenha permissão para visualizar um portfólio específico na aplicação. Sempre que for feita uma chamada para visualizar o portfólio, o caso de uso "Verificar Permissão de Acesso" é ativado para autenticar a entrada deste utilizador para visualizar o portfólio. Além disso, é sempre verificado pelo caso de uso "Verificar se Utilizador está Autenticado no Sistema", para que possa continuar a utilizar a aplicação. Caso o utilizador não esteja autenticado na aplicação, ele será automaticamente redirecionado para a página de *login* para que realize o seu *login* novamente.

O caso de uso "Alterar Portfólio" está associado aos atores de cargo "Criança", na qual permite que o utilizador desde tipo altere as informações de seu portfólio, como nome, idade, grupo ou instituição.

O caso de uso "Manter Observação" está associado aos atores do cargo "Gerente" no sistema. Este caso de uso permite aos gerentes as operações de criar, alterar, ver e excluir observações de um portfólio.

O caso de uso "Permitir Acesso" está associado aos atores do cargo "Professor", na qual permite o acesso de um outro utilizador a um portfólio específico. O utilizador deve inserir o e-mail do outro utilizador na qual ele quer permitir o acesso ao portfólio.

O caso de uso “Manter Portfólio” está associado aos atores do cargo “Professor”, na qual permite aos professores operações de criar, alterar, ver e excluir portfólios. Para criar um portfólio é necessário inserir informações como nome da criança, idade e grupo a qual ela pertence.

O caso de uso “Alterar Cargo” está associado aos atores do cargo “Administrador”. O caso de uso permite que o utilizador do cargo “Administrador” possa alterar o cargo de um outro utilizador, para que ele possa ter mais opções de atividades e funcionalidades dentro do sistema, como criar novas observações ou portfólios.

O caso de uso “Associar Portfólio” está associado aos atores do tipo “Administrador”, na qual ele associa portfólios aos utilizadores, para que seja possível visualizar, alterar ou excluir um portfólio ou observação de acordo com o cargo do utilizador.

3.2 Atores

Esta seção aborda os atores e seus papéis no sistema. O sistema possui ao todo seis atores, sendo eles: administrador, professor, gerente, criança, parente e convidado. Cada um destes atores podem realizar ações no sistema, a cada vez que aumenta o grau de cargo de um utilizador, suas funcionalidades no sistemas também aumentam. Estes cargos e suas respectivas funcionalidades estão descritas nas subseções seguintes.

3.2.1 Convidado e Parente

O convidado é o utilizador mais comum no sistema. Assim que o utilizador realizar o *login* no sistema, ele passa a ter o cargo de convidado. Com este cargo o utilizador pode acessar as funcionalidades de alterar perfil, visualizar portfólio desde que tenha permissão e realizar *logout*. O cargo de Parente possui as mesmas funcionalidades do cargo de convidado porém serve para categorizar o utilizador como parente de uma criança.

3.2.2 Criança

A criança possui as mesmas funcionalidades dos cargos convidado e parente. Além disso este cargo possui a funcionalidade de poder alterar as informações como nome, idade e turma de seu portfólio.

3.2.3 Gerente

O gerente possui todas as outras funcionalidades apresentadas anteriormente e, além disso, possui a funcionalidade para criar, ler, alterar e excluir observações de portfólios. Este cargo é direcionado a pessoas que possuem papel de suporte do professor, na qual ele pode ajudar na construção do portfólio das crianças, podendo criar novas observações ou alterar informações das observações no portfólio da criança em que ele possui permissão.

3.2.4 Professor

O professor é o responsável por criar os portfólios no sistema. Ele possui todas as funcionalidades apresentadas nos cargos anteriores e, adicionalmente, possui as funcionalidades de criar, ler, alterar seus portfólios e permitir acesso, na qual permite que outros utilizadores além dele terem acesso a um portfólio específico seguindo sua respectivo cargo com suas permissões.

3.2.5 Administrador

O administrador possui todas as funcionalidades do sistema. Ele é responsável por alterar os cargos dos outros utilizadores e, por isso, possui a funcionalidade alterar cargo. Outra funcionalidade que o administrador possui é a de associar portfólio, que permite que ele controle o acesso dos utilizadores, adicionando ou removendo permissão para visualizar os portfólios da instituição.

3.3 Diagrama de Sequência

O Diagrama de Sequência da figura 3.2 apresenta algumas ações que um utilizador do cargo “Professor” pode exercer no sistema. Na primeira atividade, o professor envia seus dados através de uma interface para criar uma conta na aplicação. Os dados inseridos são enviados e armazenados no banco de dados da aplicação através do objeto responsável pelos portfólios, para então apresentar uma resposta ao utilizador de que o cadastro foi concluído com sucesso.

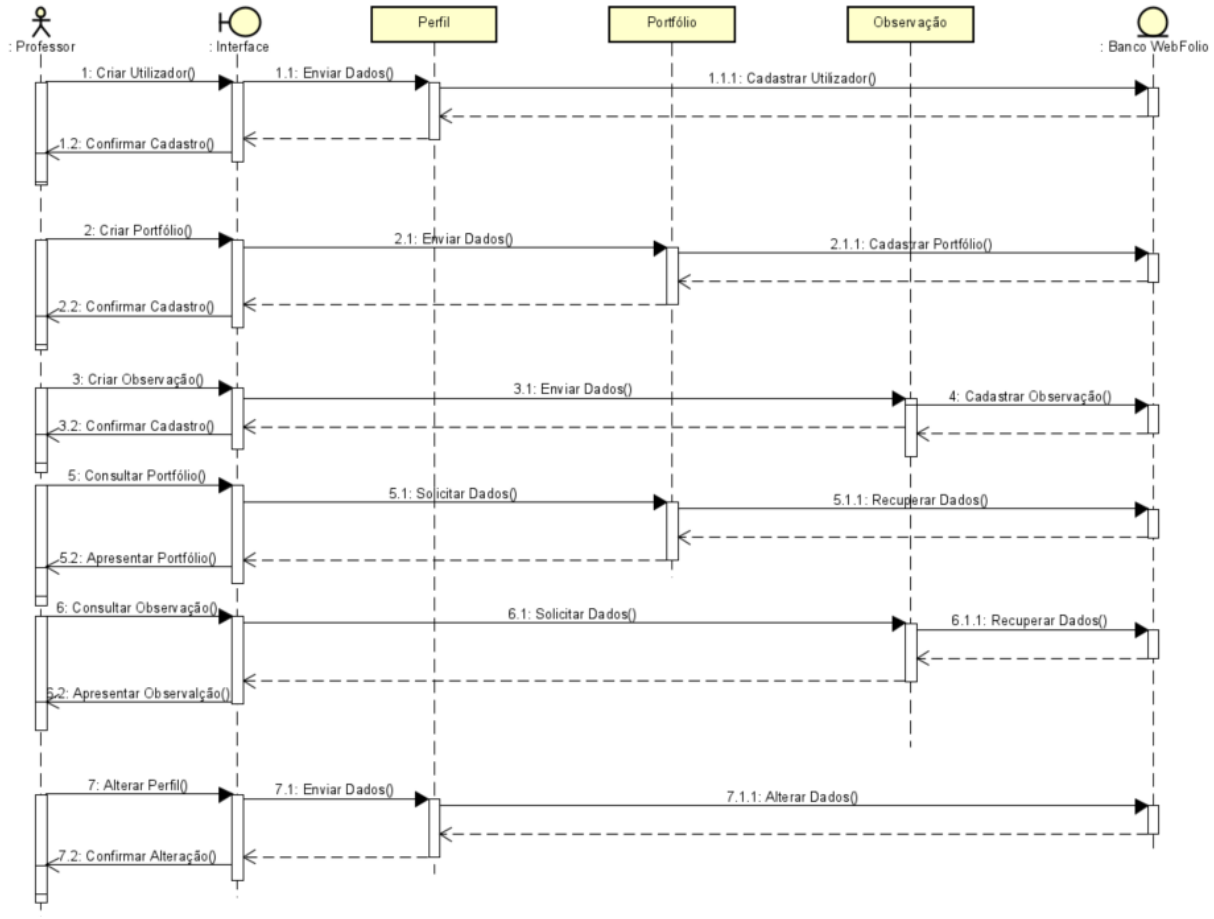

Figura 3.2: Diagrama de Sequência do utilizador do cargo professor.

Em sequência, em sua segunda atividade no sistema, é apresentado a criação de um portfólio na aplicação. Novamente os dados inseridos através da interface são enviados para o banco de dados da aplicação e, posteriormente, é confirmado se o cadastro do

portfólio foi realizado com sucesso. O mesmo se repete para a criação da observação, na qual os dados são enviados pelo utilizador através de uma interface que aciona o objeto responsável por enviar os dados para o banco de dados da aplicação para o cadastro da observação.

Na quinta atividade, ao consultar um portfólio, o utilizador solicita dados de um portfólio específico no sistema. Este objeto, por sua vez, faz a requisição dos dados o banco de dados e apor fim apresenta os dados do portfólio para o utilizador. O mesmo ocorre para consultar uma observação, na qual a solicitação de uma observação é buscado no banco de dados e apresentado ao utilizador.

Na última atividade da figura 3.2, sétima atividade, o utilizador envia dados através da interface para alterar seus dados de perfil. Os dados novamente são enviados pelo objeto responsável pelo perfil para o banco de dados da aplicação para registrar a nova mudança e assim apresentar uma confirmação na alteração do perfil.

O Diagrama de Sequência da figura 3.3 apresenta algumas ações que um utilizador do cargo "Administrador" pode exercer no sistema. Na primeira atividade, o utilizador solicita os dados dos usuários do sistema, através da consulta do objeto *Dashboard* Administrativo, que busca os dados dos utilizadores no banco de dados da aplicação e os apresenta.

Na segunda atividade, o administrador informa os dados para alterar o cargo de um utilizador na aplicação. Os dados são enviados para o banco de dados da aplicação e uma confirmação de alteração é enviada para o cliente.

A terceira atividade exercida pelo utilizador associa um portfólio a um utilizador no sistema, para que ele tenha acesso permitido para visualizar o portfólio, além de poder alterar o informação do portfólio ou de suas observações, mediante ao seu tipo de cargo no sistema. A mesma operação ocorre, os dados da associação são enviados para o banco de dados da aplicação e é apresentado uma resposta com a confirmação da alteração para o usuário. As atividades posteriores consistem nas mesmas atividades exercidas pelo utilizador do cargo professor apresentado anteriormente.

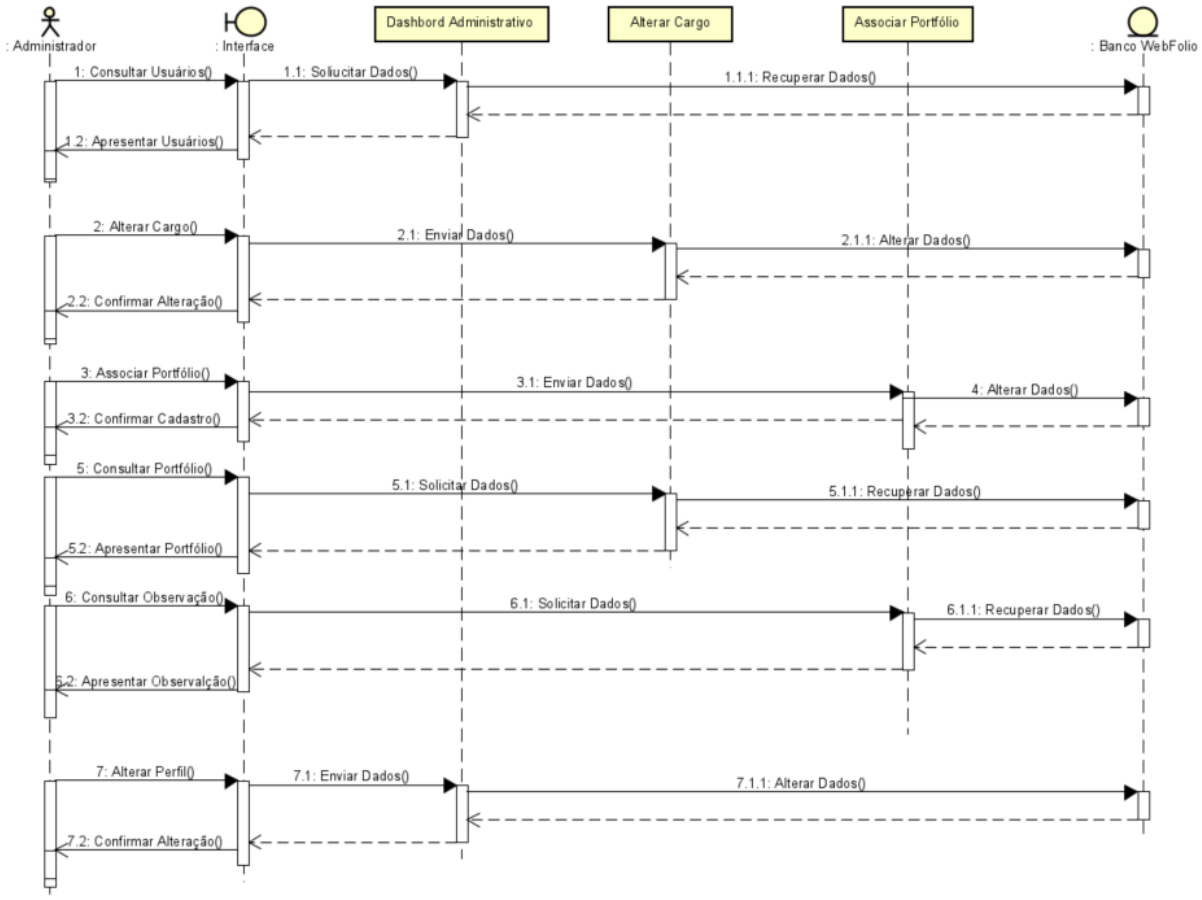

Figura 3.3: Diagrama de Sequência do utilizador do cargo administrador.

3.4 Diagrama de Atividades

Nesta seção é apresentado o diagrama de atividade e a descrição de seu fluxo. Esta abordagem apresenta o seu fluxo desconsiderando as limitações de cargos no sistema, sendo assim, esta abordagem leva em consideração que o utilizador pode acessar todas as atividades da aplicação (Figura 3.4).

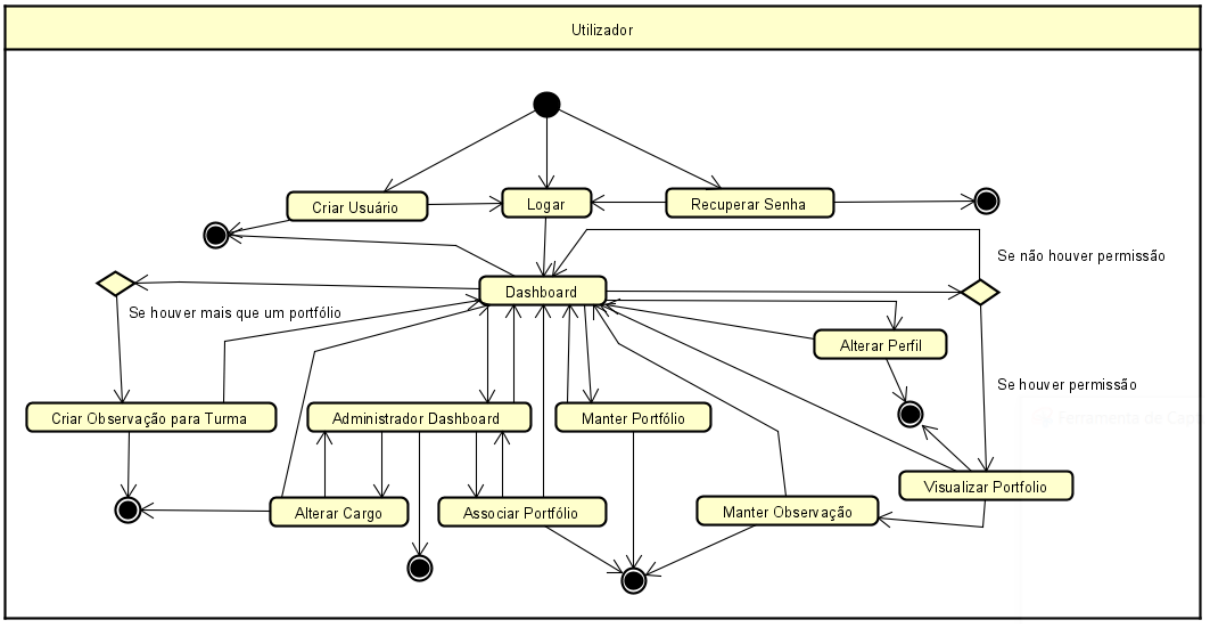

Figura 3.4: Diagrama de Atividade.

Inicialmente, o utilizador pode ter acesso às atividades "Recuperar Senha" ou criar um novo utilizador através da atividade "Criar Utilizador" antes de efetuar o *login* no sistema. Após efetuar o seu *login* no sistema representado pela atividade "Logar", o utilizador é direcionado à página principal da aplicação. A página de *Dashboard*, representado pela atividade "*Dashboard*". A partir do *Dashboard* é possível acessar as atividades de alterar perfil do utilizador. Para que possa alterar suas informações no sistema, e após sua alteração o utilizador, é direcionado ao *Dashboard*. É possível criar ou alterar um portfólio representado pela atividade "Manter Portfólio" na figura 3.4. Assim que criado ou alterado um portfólio o utilizador é redirecionado para a *Dashboard*.

Caso houver mais que um portfólio criado pelo utilizador, é habilitada a atividade

"Criar Observação para Grupo", na qual é possível ele criar uma observação para diversos portfólios simultaneamente. Após criado esta observação, o utilizador é enviado para a *Dashboard*.

A atividade "Visualizar Portfólio" pode ser acessada somente se o utilizador possuir permissão para acessar aquele portfólio. Quando houver a permissão, o utilizador pode criar, ver, editar ou excluir as observações representadas pela atividade "Manter Observação", e posteriormente é enviado para a atividade "Visualizar Portfólio".

A partir da *Dashboard* é possível acessar a *Dashboard* de administrador representado pela atividade "*Administrador Dashboard*". Através da página de *Dashboard* é possível exercer as atividades de "Alterar Cargo", que altera o cargo do próprio ou de outro utilizador no sistema, e "Associar Portfólio", que pode associar os portfólios da instituição a um ou mais utilizadores. Após concluir uma dessas atividades o utilizador é redirecionado para a página de *Dashboard* de administrador.

De qualquer parte do sistema é possível voltar para a página principal representado pela atividade "*Dashboard*", além disso, também é possível finalizar a aplicação em qualquer parte do sistema.

3.5 Diagrama de Classes

Nesta diagrama são representadas cinco classes, no qual é possível visualizar as relações entre si. A classe `User_token` é utilizado para recuperar a senha do utilizador na aplicação, o utilizador após obter *token*, é permitido criar uma nova senha para logar na aplicação. A relação mostra que Um utilizador pode ter mais de um *token* criado para a recuperação de senha.

A relação entre a classe `User` e `Portfolio`, mostra que um utilizador pode possuir mais de um portfólio, porém o cada portfólio pode ter somente um utilizador. A classe `Portfolio` pode conter muitas observações, relacionamento com a classe `Observation`, enquanto cada observação pode somente estar associado a somente um portfólio. A classe

`Observation` pode conter muitos arquivos, entidade `File`, enquanto um `File` pode somente estar associado a uma observação somente.

A classe `User` possui os campos de identificação de utilizador, nome, senha, avatar, e-mail e instituição (Figura 3.4). Além disso o utilizador possui os métodos `create_portfolio`, indicando que o utilizador pode criar portfólios na aplicação, `create_observation`, permite que o utilizador crie observações no sistema, `update_profile`, permite que o utilizador altere suas informações no sistema, e *recover_password*, permite ao utilizador recuperar a senha de utilizador caso ele tenha perdido.

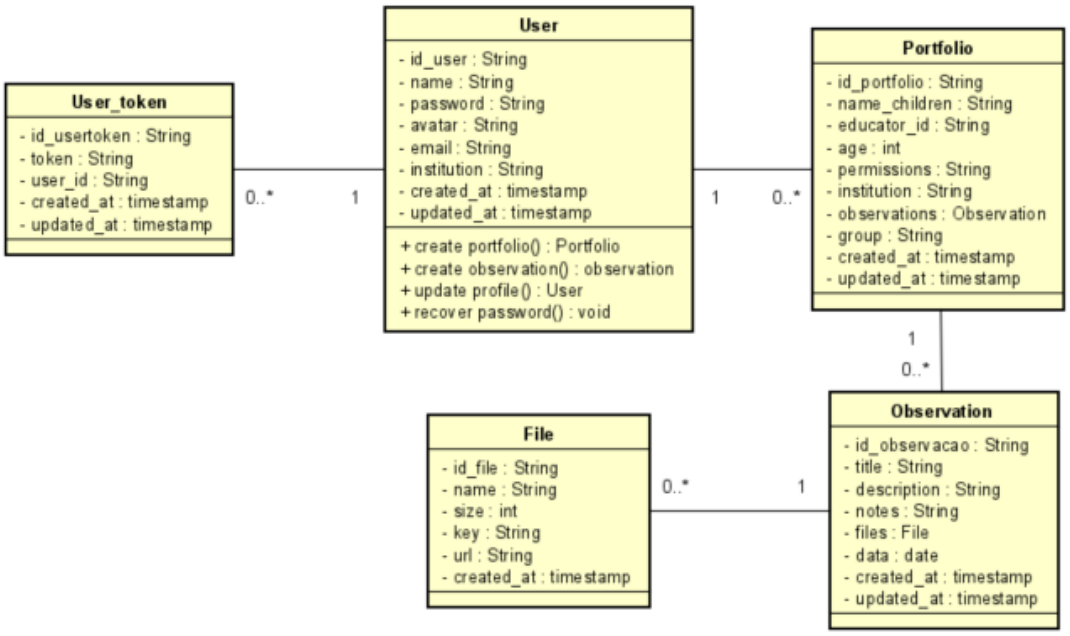

Figura 3.5: Diagrama de Classe da aplicação Webfolio.

Para a classe `Portfolio`, ele possui os atributos de identificação do portfólio, nome da criança, idade, instituição, id do educador criador do portfólio e por fim suas permissões de acesso. A classe `Observation` possui os atributos de identificação da observação, titulo da observação, descrição da atividade, data da observação, arquivos da observação e por fim as notas, que podem se referir ao desempenho particular de uma criança durante uma atividade.

A classe `File` contém as informações necessárias para guardar o tamanho do arquivo,

seu nome, e caminho para acessar o arquivo.

Apesar de um portfólio estar ligado a um utilizador, ele ainda pode ser acessado por outros utilizadores, contanto que esses utilizadores estejam em sua de lista de permissão. Porém o portfólio possui apenas um utilizador criador deste respectivo portfólio.

3.6 *Mockups*

Nesta seção são apresentadas todos os *mockups* utilizados para a construção das interfaces projetadas para o sistema, assim como a descrição e técnicas usadas para elaborar as interfaces.

Para a construção dos *mockups* foi utilizado o Software Figma [37], um editor gráfico usado para prototipagem de projetos *web*. No desenvolvimento de todos os *mockups* foi utilizado um *grid*, ou grade, que é constituído por margens e colunas para auxiliar no alinhamento e harmonia dos conteúdos na interface (Figura 3.6). Este *grid* possui 12 colunas com tamanho de 64 pixels, com margin de 160 pixels de distância das bordas da interface, que possui o tamanho padrão de 1440 pixels para aplicações desktop, e possui 32 pixels de distância entre cada coluna.

Para a criação de todo o conteúdo das interfaces, foi utilizado para tamanhos de fontes e espaçamento tamanho múltiplo de oito ou quatro para facilitar a harmonia e o alinhamento dos conteúdos nas páginas. Para construir uma harmonia com cores para a interface, foi utilizado cores complementares como o rosa e o verde para os componentes da aplicação. Os títulos e descrições levam a cor verde enquanto a cor rosa é utilizado para botões e formulários dentro da aplicação, que pode ser visto nas figuras 3.7 e 3.8. Além disso é utilizado a cor branco para o fundo de formulários, textos e ícones na aplicação, enquanto a cor preto é usada para títulos das páginas dentro da aplicação, como pode ser vista nas figuras 3.9 e 3.10. A ferramenta Adobe Illustrador Draw [38] foi utilizado para o desenvolvimento do logo da aplicação, que pode ser vista em todas as interfaces da aplicação.

Para a página de cadastro, foi reutilizado a mesma estrutura da página de *login*, desta

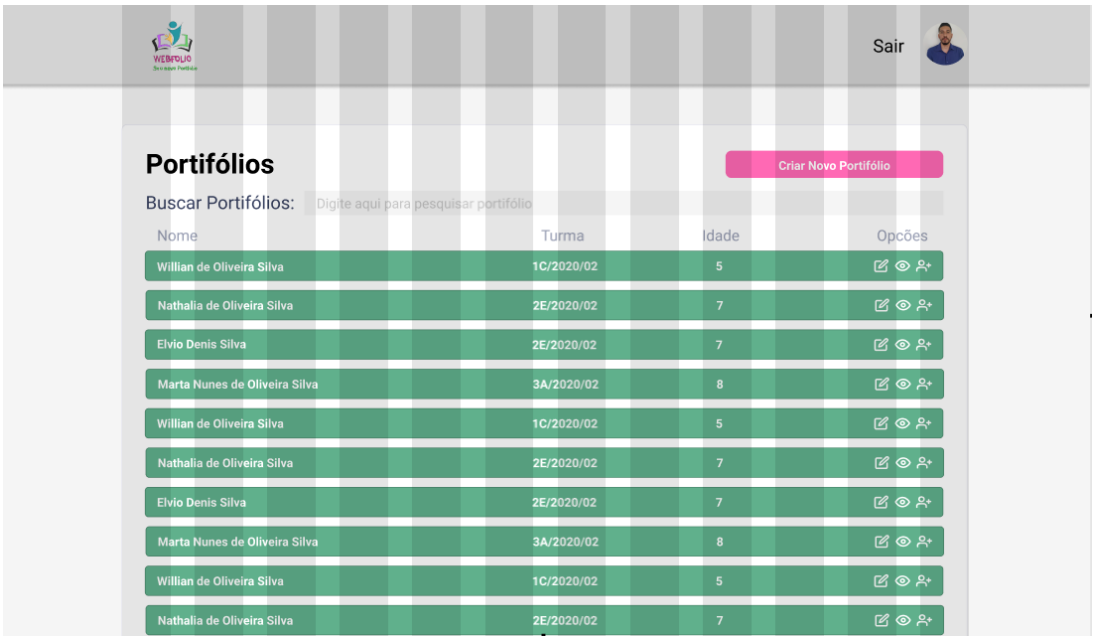

Figura 3.6: *Mockup* da página de *Dashboard* com *grid*.

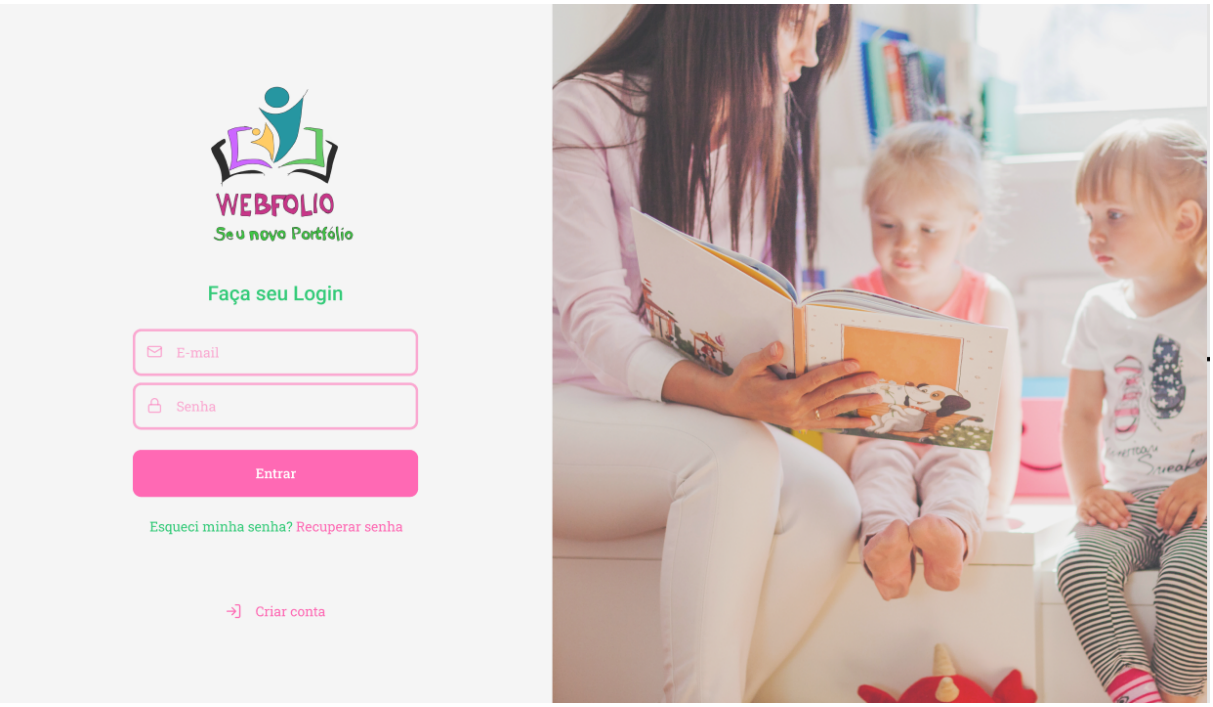

Figura 3.7: *Mockup* da página de *login*.

vez, com a adição de mais campos, como instituição e nome, que podem ser visualizadas na figura 3.8.

Figura 3.8: *Mockup* da página de cadastro de utilizador.

Para facilitar a utilização da aplicação, todas as ações no sistema foram simplificadas para serem de identificadas facilmente e serem intuitivas. Para a página de *Dashboard* teve como objetivo deixar todos os portfólios criados pelo utilizador para sua fácil visualização em uma tabela. Esta tabela contém o nome da criança, sua turma e idade, e contém três ícones para indicar as ações do utilizador naquele portfólio. O primeiro ícone significa edição das informações do portfólio, o segundo ícone significa a visualização das observações do portfólio, e o terceiro ícone, indica que o utilizador pode permitir um outro utilizador a ter acesso a este portfólio. A página de *Dashboard* possui um menu na parte superior da aplicação, que esta incluso em todas as páginas dentro da aplicação, na qual contém um botão "sair" para fazer o *logout* do sistema, uma pequena imagem com a foto de perfil do utilizador e o logo da aplicação. Além disso, a página contém um botão para criar um novo portfólio e um menu de busca para auxiliar na procura de um portfólio

específico na aplicação. Todas estas informações podem ser vistas na figura 3.9.

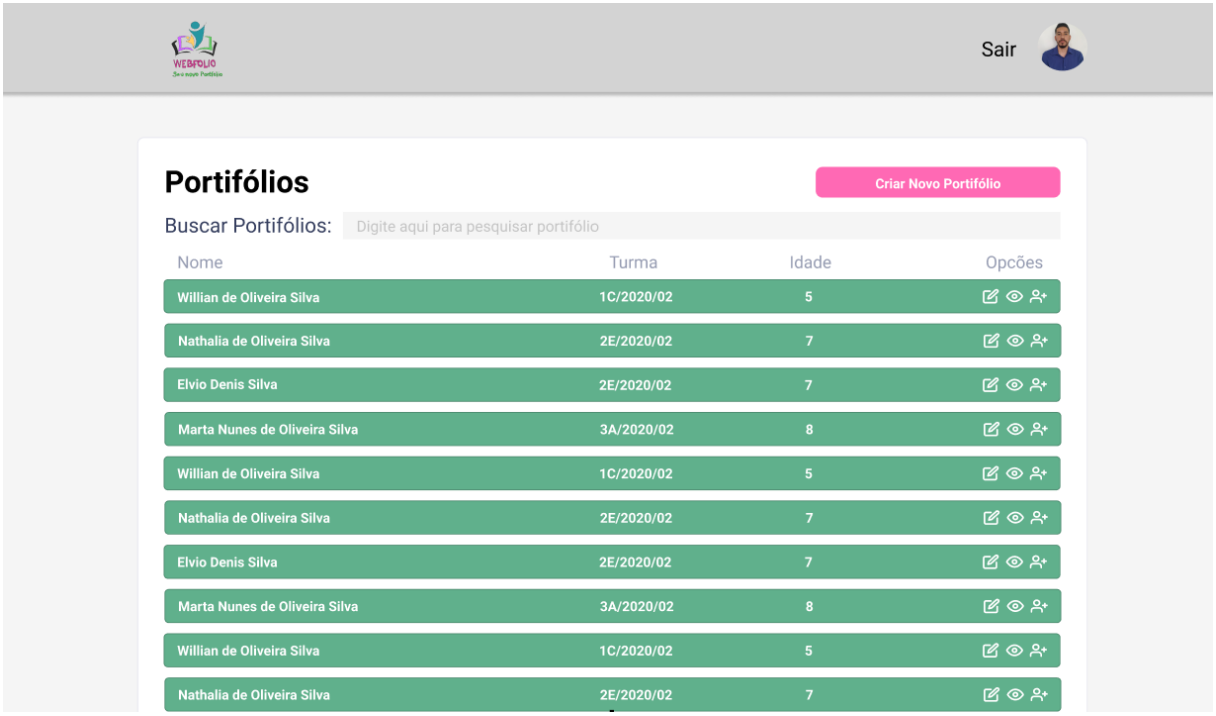

Figura 3.9: *Mockup* da página de *Dashboard*.

Na interface da cadastro do portfólio, apresentado na figura 3.10, contém somente um formulário com as informações necessárias para a criação do portfólio da criança, informações como o nome da criança, sua idade e grupo, além de possuir um botão para cadastrar o novo portfólio.

Similar a página de cadastro de portfólio, a página de cadastro de observação, na figura 3.11, possui somente suas informações necessárias para o seu cadastro no sistema. Estas páginas também serão reutilizadas para a alteração de informação dos respectivos documentos.

Na interface de perfil do utilizador, a página contém as informações do utilizador, dados como seu nome e e-mail, com a possibilidade de serem alteradas, além de possui campos para permitir que o utilizador troque sua, informando sua senha atual e uma nova senha, assim como a confirmação de sua nova senha (Figura 3.12). Para finalizar o processo do formulário há um botão para confirmar as mudanças feitas no perfil.

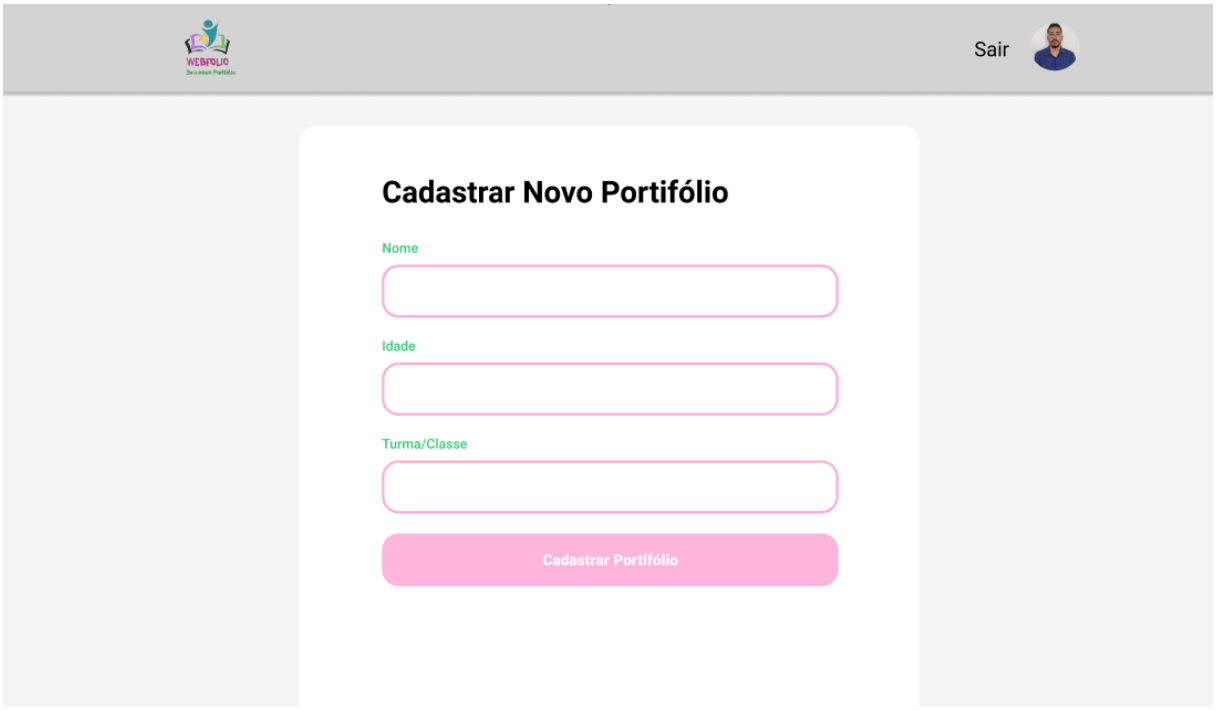

Figura 3.10: *Mockup* da página de cadastro de um novo portfólio.

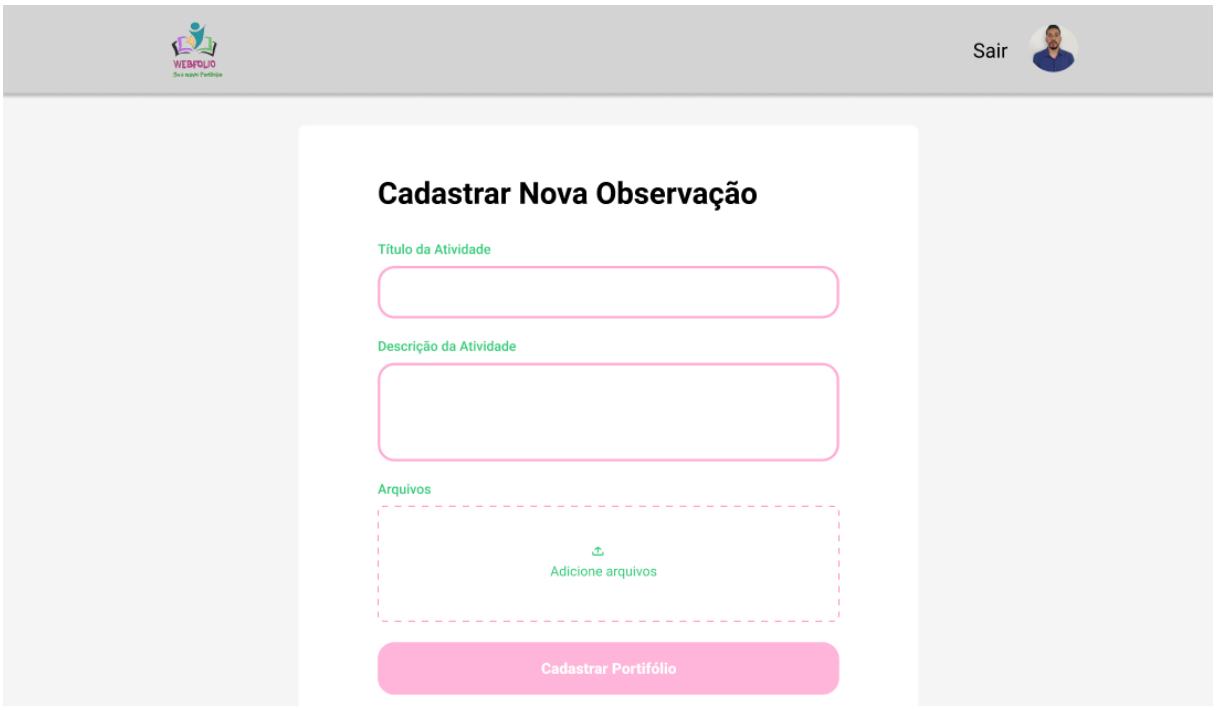

Figura 3.11: *Mockup* da página de cadastro de uma nova observação.

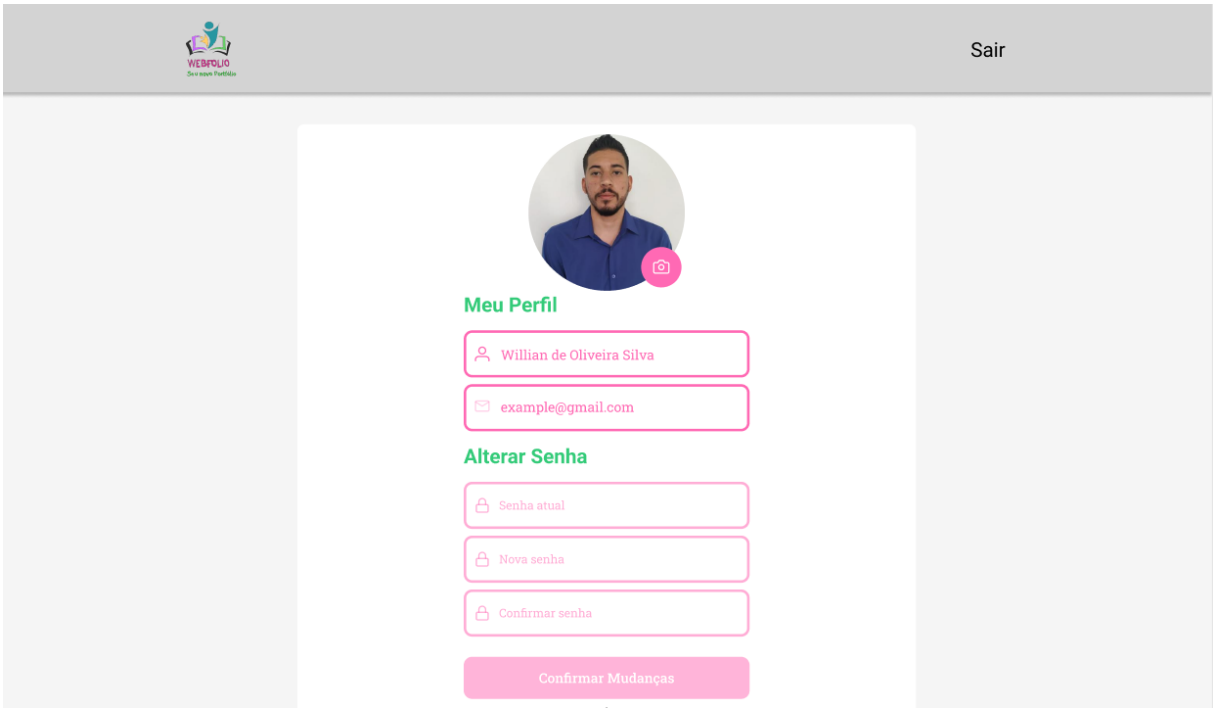

Figura 3.12: *Mockup* da página de perfil do utilizador.

Por fim, a página de portfólio possui o nome da criança do portfólio como título da página, o nome do educador logo abaixo para indicar quem é o professor responsável pelo portfólio da criança e um botão com a descrição de criar uma nova observação para aquele portfólio ao lado (Figura 3.13). Esta página possui blocos, nomeados *cards*, que possuem informações de uma observação da criança. A observação constitui-se de título da atividade, descrição da atividade, e arquivos (áudios, vídeo, imagens), além de possuir dois ícones, o primeiro indica a excluir aquela observação, enquanto o segundo ícone indica a alteração daquela observação.

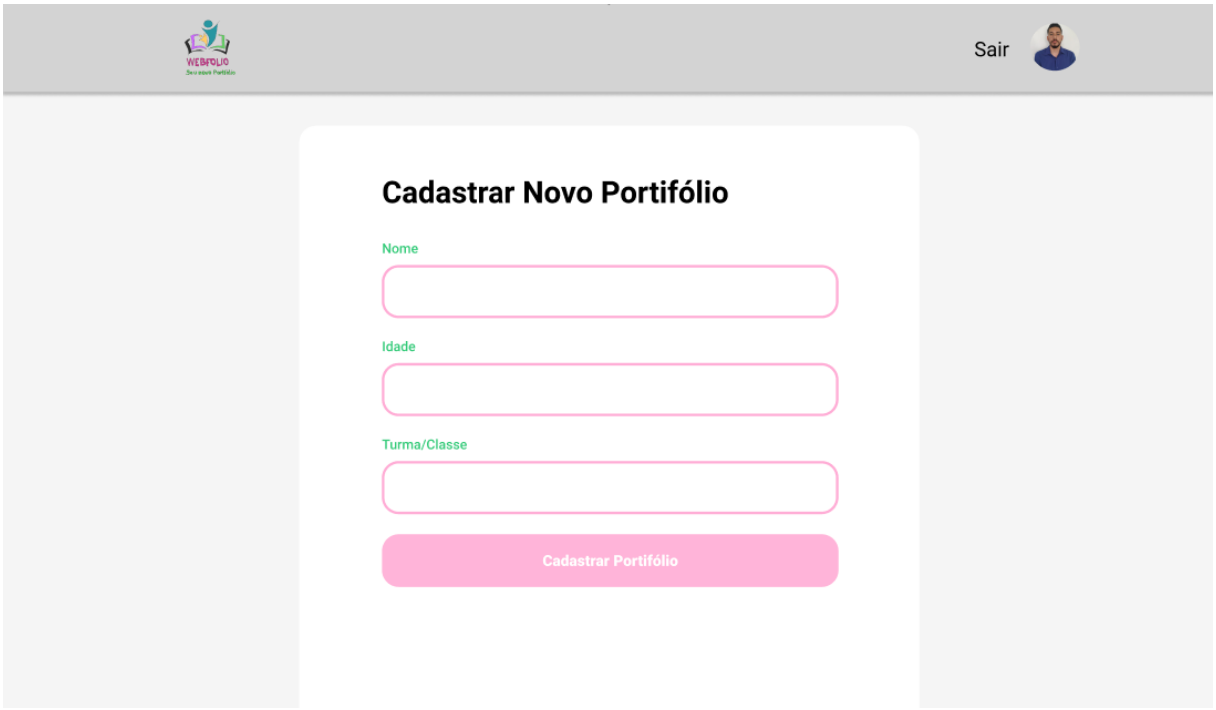

Figura 3.13: *Mockup* da página de portfólio.

Capítulo 4

Desenvolvimento e Implementação

Neste capítulo são descritos os pontos mais relevantes, soluções técnicas aplicadas e dificuldades encontradas durante a implementação do trabalho, além das interfaces obtidas e suas descrições.

4.1 Segurança e controlo de acesso

Para estabelecer uma comunicação entre a aplicação *front-end* e a base de dados foi desenvolvido uma API RESTful. Para proteger o acesso das informações por pessoas não autorizadas no sistema e ter acesso as utilizações do sistema é necessário um *token* de autenticação JWT, que pode ser gerado após o utilizador efetuar o seu *login* no sistema. Ao informar o e-mail e senha para autenticação no sistema, a rota "/sessions" é responsável por criar o *token* JWT. Ao receber uma requisição ela realiza uma busca no banco de dados da aplicação e verifica se a senha e o e-mail informados estão corretos, caso estejam corretos, o *token* JWT é criado.

O *token* JWT, figura 4.1, é uma criptografia que possui informações que são separadas por pontos. A primeira informação é chamada de *headers* e se refere ao tipo de algoritmo utilizado para criar o *token*, a segunda informação é chamada de *payload*, que possui as informações do usuário, como id do utilizador, nome e e-mail, excluindo somente

a informação de senha do utilizador, porque essas informações podem ser utilizadas posteriormente na aplicação. A última parte do *token* é chamada de assinatura, esta parte assegura que o *token* não foi modificado externamente por outro utilizador.

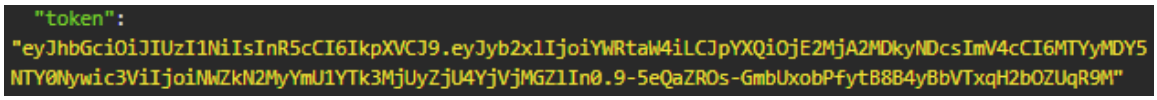

Figura 4.1: *Token* de acesso formado pela aplicação.

Algumas rotas, além de exigirem o *token* de autenticação, também exigem um certo nível de permissão no sistema. Para atualizar informações de um utilizador, por exemplo, é necessário que o utilizador que iniciou a requisição seja um utilizador que possua o cargo "Administrador". Na figura 4.2 é apresentado como é utilizado a definição da rota para atualizar o cargo de um utilizador. Os *middlewares* são capazes de executar qualquer código, alterar objetos, além de encerrar ciclo de requisição e resposta e chamar a próxima função *middleware* que está na pilha. O primeiro parâmetro passado é uma *string* com o caminho da rota, a seguir os outros parâmetros passados são funções *middlewares* que são executados em ordem na qual foram passadas. A função "ensureAuthenticated" verifica se o utilizador possui um *token* válido no sistema, enquanto a segunda função, chamada "checkUserIsAdmin", verifica se o utilizador possui o cargo necessário para a requisição. Por fim, a última função *middleware* é a função que contém a lógica para atualizar o cargo do utilizador.

```
usersRouter.put('/:user_id', ensureAuthenticated,
  checkUserIsAdmin, UserController.update);
```

Figura 4.2: Exemplo de código que utiliza funções de *middleware*.

Para evitar ataques de *Distributed Denial of Service* (DDoS), foi adicionado a ferramenta Express rate limit à aplicação. Ela salva o endereço *Internet Protocol* (IP) de cada utilizador que acessa a aplicação em um banco de dados predefinido, neste caso o Redis [39], que é um armazenamento de estruturas de dados em memória que utiliza memória distribuída de chave-valor. Juntamente ao IP, também é salvo o *timestamp* em que o

acesso é ocorrido para, desta maneira, fazer a tratativa de negar ou não o acesso deste utilizador e assim prevenir um ataque.

4.2 Documentação

Para facilitar o entendimento de rotas e de instalação da aplicação, a API foi hospedada e documentada na plataforma de hospedagem GitHub [40]. As instruções de instalação, utilização das rotas e ferramentas utilizadas para a criação da aplicação foram descritas em linguagem Markdown e são convertidas para texto HTML, figura 4.3.

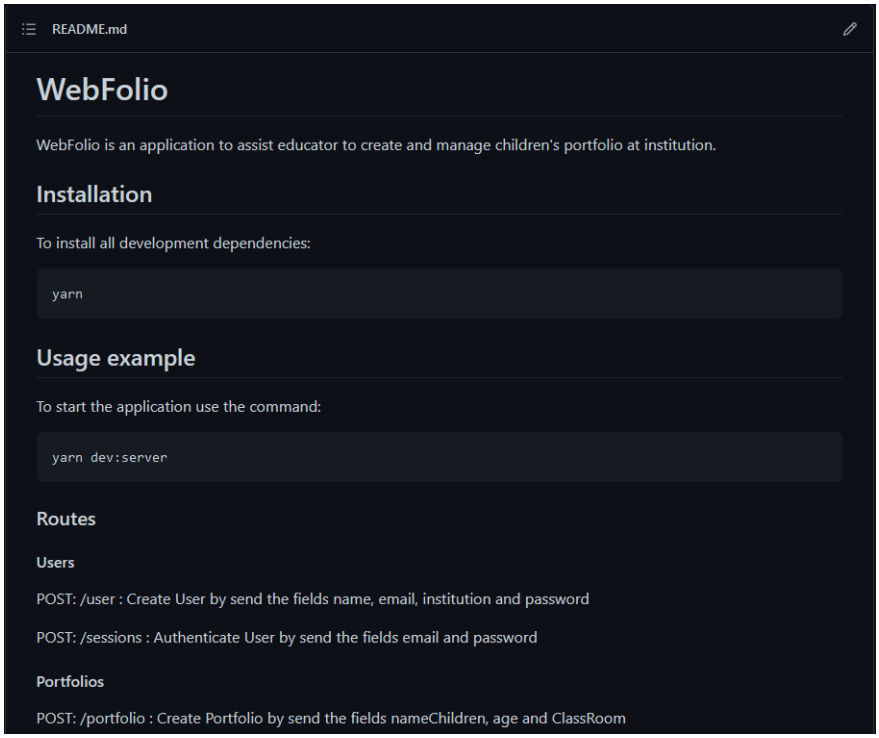

Figura 4.3: Parte da Documentação da aplicação *back-end*.

Semelhantemente a aplicação *frontend* do sistema também foi documentada com instruções de instalação, ferramentas, além de ter uma demostração de como o sistema funciona através de um *gif*, figura 4.4.

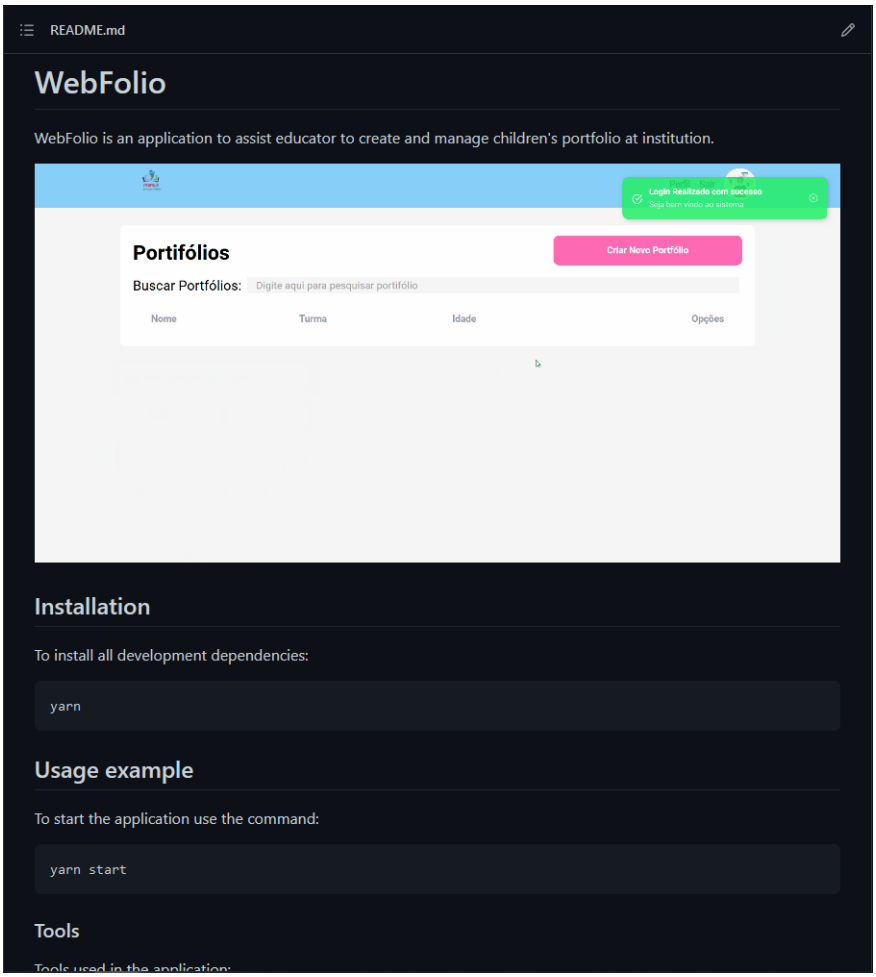

Figura 4.4: Parte da Documentação da aplicação *front-end*.

4.3 Interfaces da Aplicação

A partir das modelagens e análises propostas, foi possível obter todas as interfaces da aplicação. A tecnologia utilizada para desenvolver a aplicação *front-end* foi o React com TypeScript. A figura 4.5 apresenta a interface final de *login* do utilizador. A proposta da aplicação é ser de simples uso para o utilizador, para realizar o *login* no sistema o utilizador deve informar seus dados de e-mail e senha.

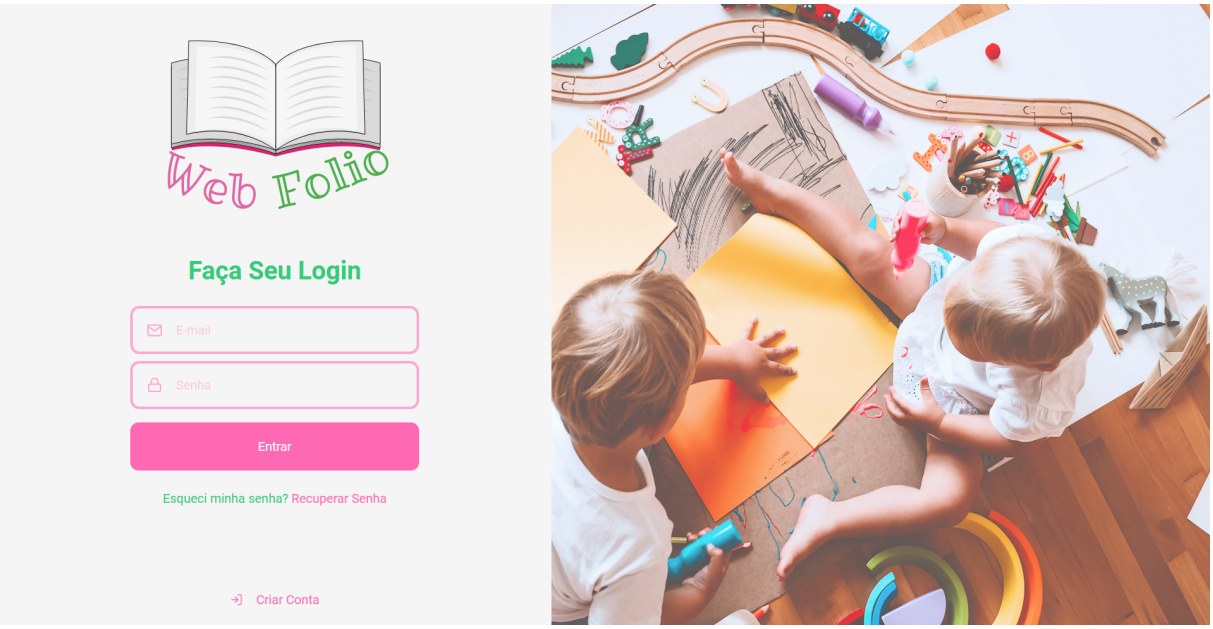

Figura 4.5: Interface da página de *login* da aplicação.

A interface de *signup*, figura 4.6, é responsável por apresentar as informações necessárias para o cadastro do utilizador na aplicação. Esta interface é similar a página de *login*, porém possui mais campos em seu formulário. Para auxiliar no cadastro do utilizador, quando o botão cadastrar é acionado, caso o utilizador não tenha preenchido um campo ou caso alguma informação estiver incorreta, como um e-mail inválido, é acionado um *toast*, que pode também ser chamado de mensagem de alerta ou notificação, no canto superior direito da interface, indicando qual foi o erro do utilizador para criar cadastro. O sistema possui ao todo três tipos diferentes de *toasts*, um para indicar falta de preenchimento de campo específico na interface, outro para indicar que houve um erro na operação ou que

o usuário inseriu uma informação incorreta para o sistema, como indicar um e-mail ou senha inválidos para fazer *login* no sistema, e por fim, o último *toast* é para informar que um procedimento na aplicação foi realizada com sucesso. Os *toasts* são acionados em todo a aplicação, e podem ser vistos na figura 4.7.

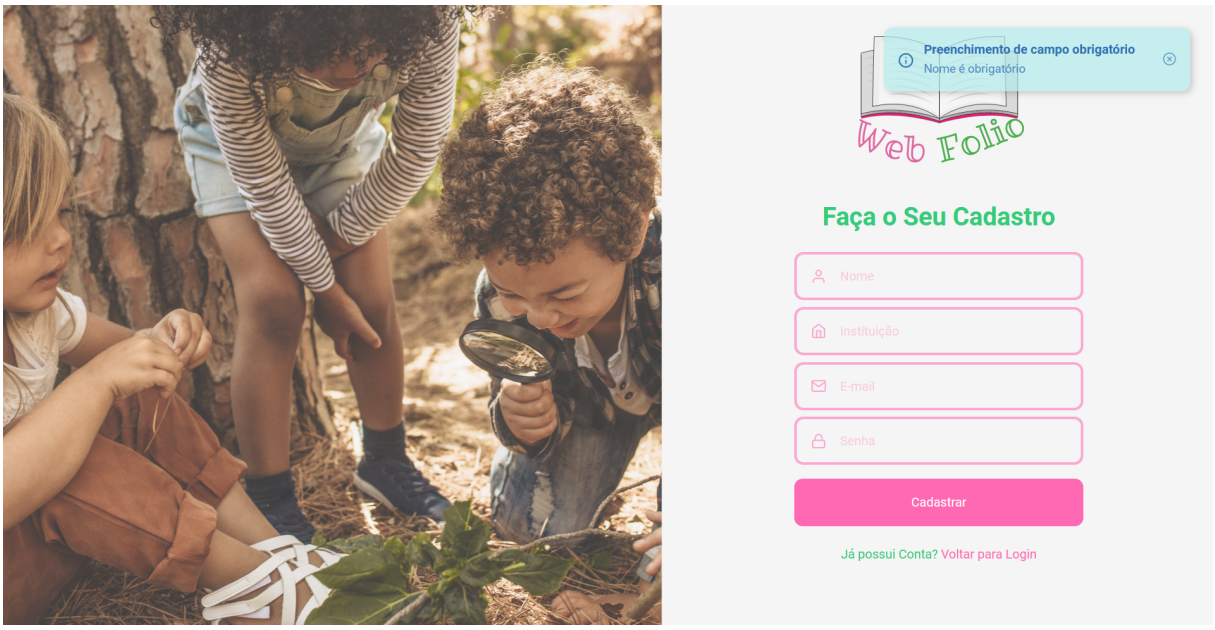

Figura 4.6: Interface da página de cadastro da aplicação.

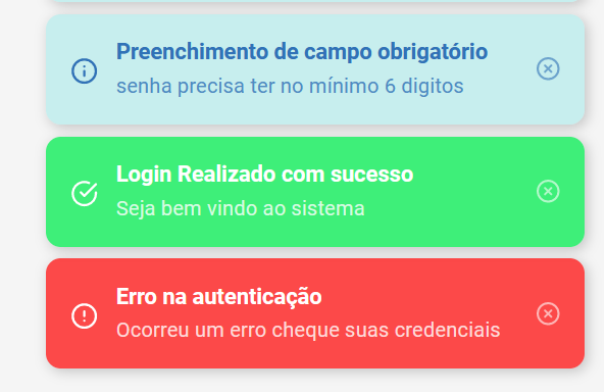

Figura 4.7: Exemplo de mensagens informativas do sistema.

As interfaces das páginas de recuperar senha e resetar senha seguem a mesma base das interfaces de *login* e da página de cadastro do sistema. A interface da página de recuperar senha, figura 4.8, possui apenas um campo para inserir o e-mail, enquanto a interface

da página de resetar senha, figura 4.9, possui os campos de nova senha e confirmar nova senha, além disso, para efetuar a recuperação da senha é necessário enviar o *token* de utilizador, que é recebido juntamente com o endereço na qual o utilizador deve acessar para de resetar a senha. O *token* então é enviado automaticamente quando o utilizador inserir as informações de nova senha e a de confirmação da senha.

Figura 4.8: Interface da página de recuperar senha da aplicação.

Para a interface da página de *Dashboard*, figura 4.10, foi adicionado ao menu uma opção para administrador no sistema, que só pode ser visto por utilizadores que possuam o cargo de "Administrador", para acessar recursos exclusivos para administradores. Além disso, quando houver mais de um portfólio criado pelo utilizador, um novo botão é adicionado a página, chamado de "Criar Observação para Sala", que permite ao utilizador criar uma observação para mais de uma criança por vez. Para facilitar o acesso a informação do portfólio de uma criança, o nome e ou o formulário podem ser clicados para acessar o portfólio da criança escolhida. O filtro de tabela de portfólios, figura 4.11, pode ser usado para auxiliar na busca de um portfólio específico, otimizando assim a busca de portfólios no sistema.

A figura 4.12, apresenta como os portfólios recebem cores diferentes para diferenciar

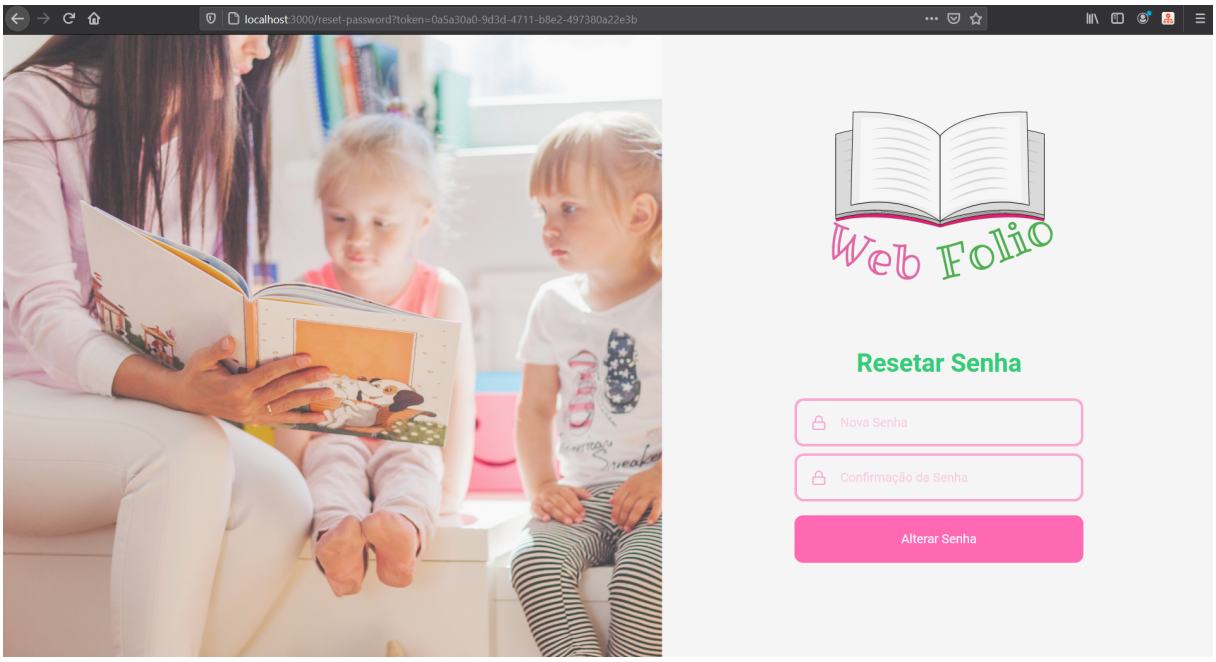

Figura 4.9: Interface da página de resetar a senha da aplicação.

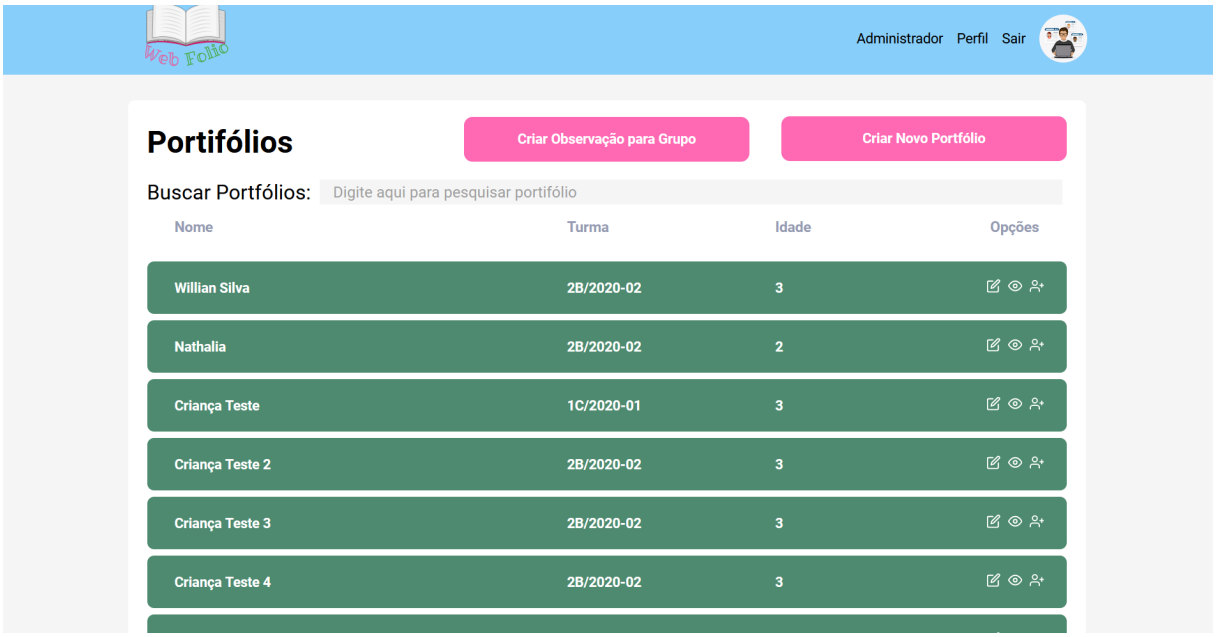

Figura 4.10: Interface da página de *Dashboard* da aplicação.

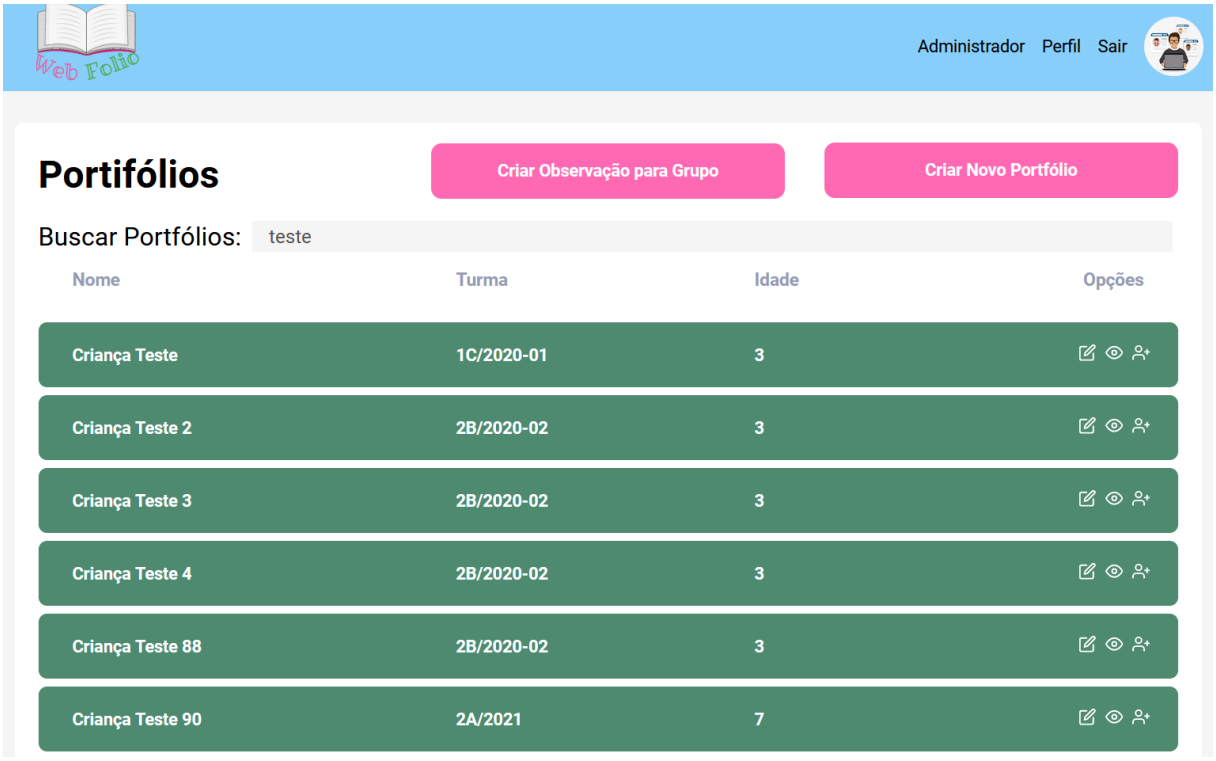

Figura 4.11: Interface da página de *Dashboard* utilizando o filtro de tabela.

as suas permissões. Na tabela, a cédula que possui a cor verde, indica que o portfólio foi criado pelo utilizador logado no sistema, este portfólio possui três opções de ícones, na qual o primeiro ícone está relacionado com a funcionalidade de editar portfólio, o segundo ícone possui como funcionalidade redirecionar o utilizador para a página do portfólio, para sua visualização completa, e o terceiro e último ícone possui a funcionalidade de redirecionar o utilizador para a página de convidar outro utilizador, para que este outro utilizador possa visualizar este portfólio. A cédula na tabela que possui a cor rosa indica que o usuário foi convidado através de e-mail para visualizar o portfólio, por isto esta tabela contém somente um ícone com a funcionalidade de direcionar o utilizador para a página do portfólio. A cédula com a cor roxa indica que o utilizador foi associado a um portfólio através de uma ação do administrador, como este utilizador possui também as permissões de professor, ele pode editar, visualizar ou convidar outro utilizador através dos ícones disponíveis.

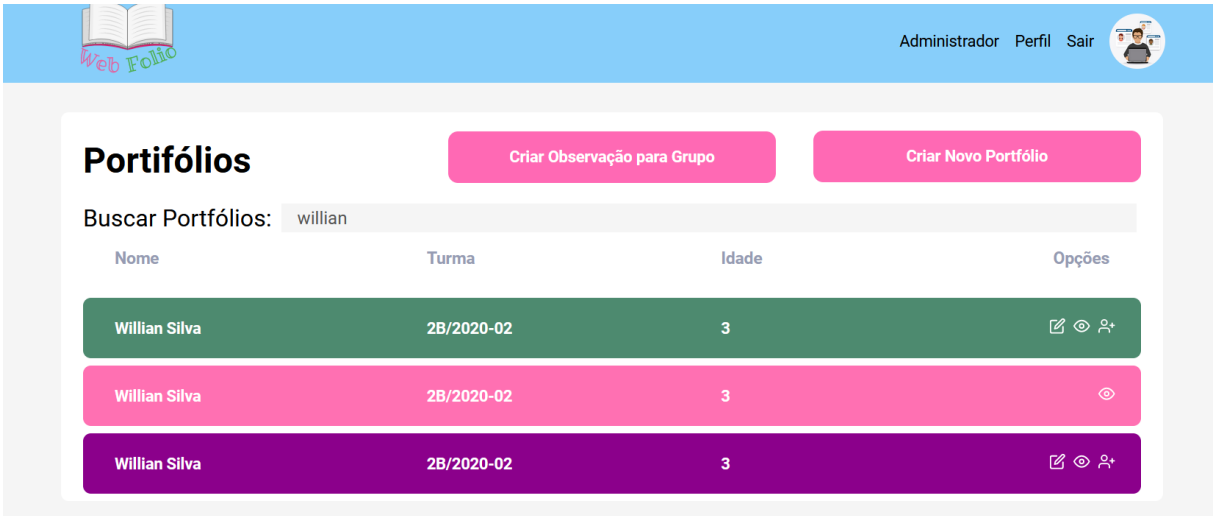

Figura 4.12: Interface da página de *Dashboard* com os possíveis permissões de portfólio.

Na interface de cadastrar portfólio, figura 4.13, no formulário da página, foram incluídos *placeholders* para auxiliar na informação ao utilizador para saber qual informação é necessária no campo a ser preenchido. Para evitar problemas na busca de grupos na aplicação, é sugerido o uso do ano ou semestre para melhor identificar uma turma no sistema.

A interface de alterar portfólio, figura 4.14, possui um campo a mais no formulário, o campo de instituição, para o caso da criança ter alterado de instituição. A interface de alterar portfólio também possui as informações buscadas do banco de dados da aplicação, as informações do portfólio escolhido, ao clicar no botão confirmar alterações, as informações do portfólio são atualizadas. Por questões de praticidade, foi adicionado a essas páginas um botão para voltar a página principal, que se localiza acima do título.

Na interface da página do portfólio, figura 4.15, as observações receberam subtítulo para ajudar na identificação da atividade da criança. Através do uso de bibliotecas como "react-audio-player" e "react-player" foi possível a execução de arquivos como áudio, vídeo e *gif*.

Para a interface da página de cadastrar uma nova observação, figura 4.16, também foram adicionados *placeholders* para auxiliar o utilizador no preenchimento dos campos. O zoom da figura 4.16 foi reduzido para facilitar a visualização completa do formulário.

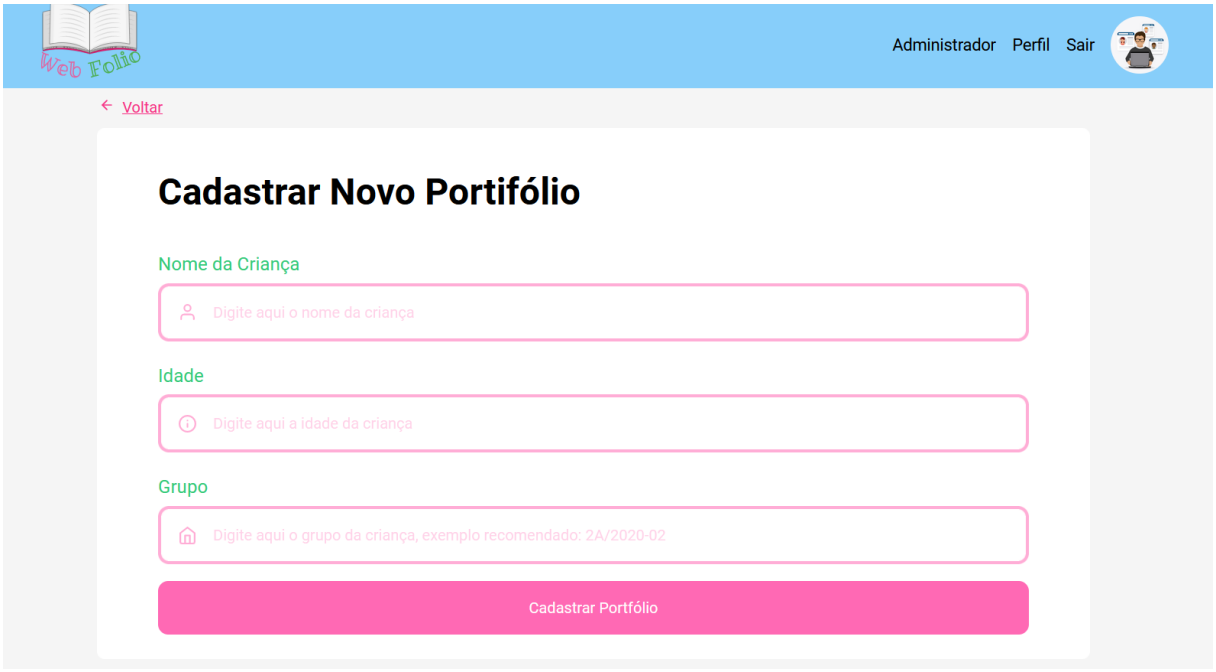

Figura 4.13: Interface da página de criar portfólio da aplicação.

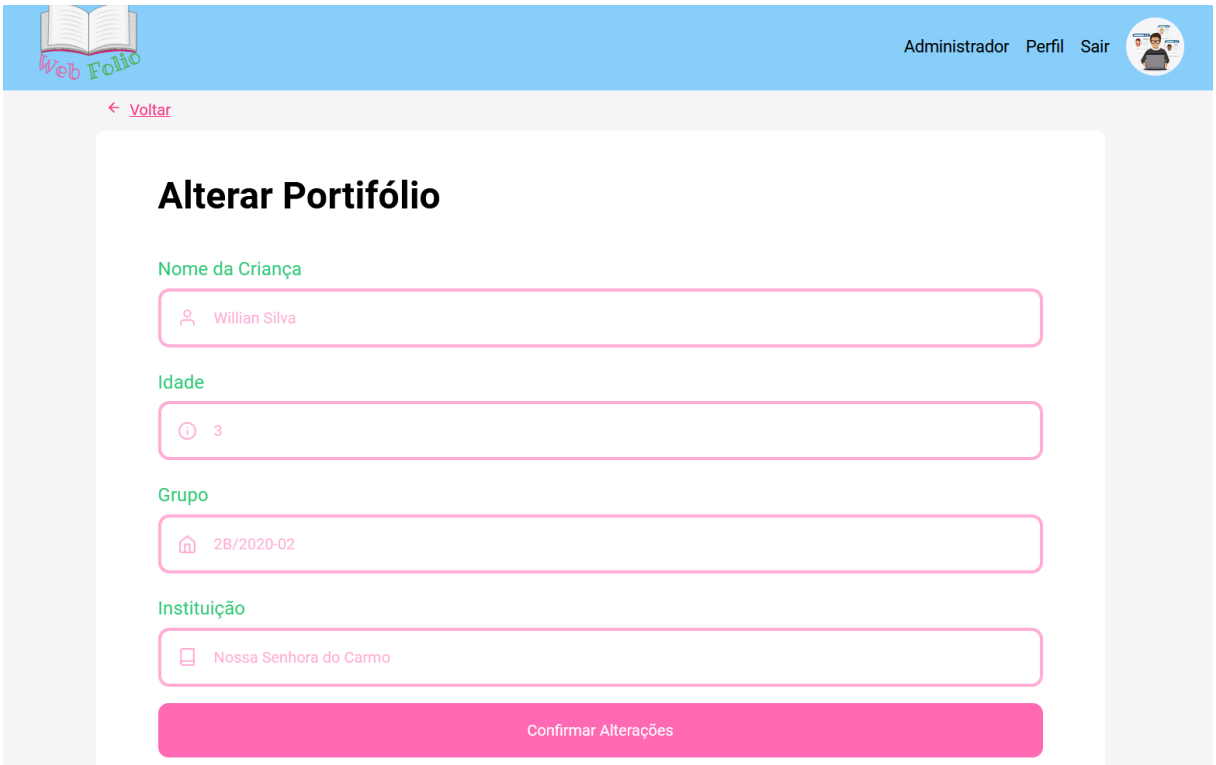

Figura 4.14: Interface da página de atualizar portfólio da aplicação.

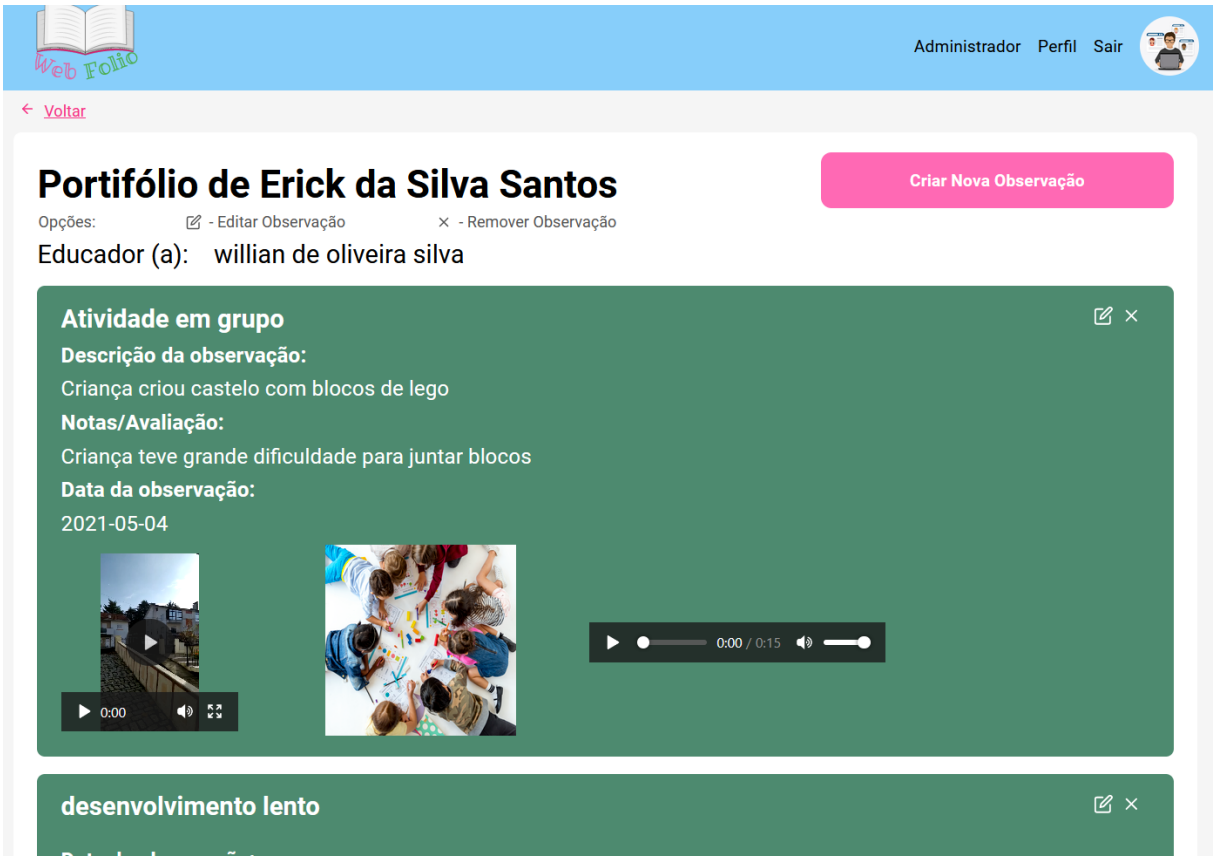

Figura 4.15: Interface da página de portfólio da aplicação.

Para criar uma observação é necessário preencher pelo menos algum campo do formulário, que possui os campos de título da observação, descrição, data da observação, notas ou avaliação da criança, enquanto a interface da página de alterar observação, figura 4.17, tem seu funcionamento similar a página de alterar portfólio, na qual ela traz todas as informações da observação para sua atualização de informações.

Na interface da página de cadastrar observação para turma, figura 4.18, o formulário possui todas as informações para criar uma observação para um grupo, exceto as notas e avaliação por se tratar de uma informação exclusiva daquela criança, no entanto, o formulário contém um novo campo selecionável para buscar um grupo específico na aplicação, permitindo escolher quais as crianças participaram da observação, figura 2.5.

Na interface da página de perfil, figura 4.20, houve a adição do botão voltar, para retornar o utilizador a página de *Dashboard*. Nesta página o utilizador pode alterar as informações de nome do utilizador, e-mail ou instituição sem precisar alterar a senha. Caso queira atualizar a senha, deve-se informar a senha antiga, a nova senha, e a confirmação da nova senha.

A interface da página de *Dashboard* do administrador, figura 4.21, pode ser somente acessado por um utilizador do tipo "Administrador". Nesta página é apresentado todos utilizadores que possuem o mesmo nome de instituição no sistema. As informações apresentadas dos utilizadores são o seus nomes, e-mails, cargo e como opções são apresentados ícones, na qual o primeiro ícone indica alterar o cargo do administrador e o segundo ícone indica associar um portfólio no sistema com este utilizador. Esta página também contém um filtro de tabela para auxiliar na busca por nome dos utilizadores da instituição. Para a página de Alterar Cargo, na figura 4.22, há somente um formulário com uma única opção para selecionar um novo cargo para o utilizador e um botão para realizar a atualização do cargo do portfólio.

A página de associação de portfólios, figura 4.23, contém todos os portfólios criados na instituição. A tabela contém informações sobre o portfólio como o nome da criança, seu grupo, idade e contém três opções, na qual o primeiro ícone indica associar o utilizador para aquele portfólio, o segundo ícone indica retirar a associação do utilizador para com

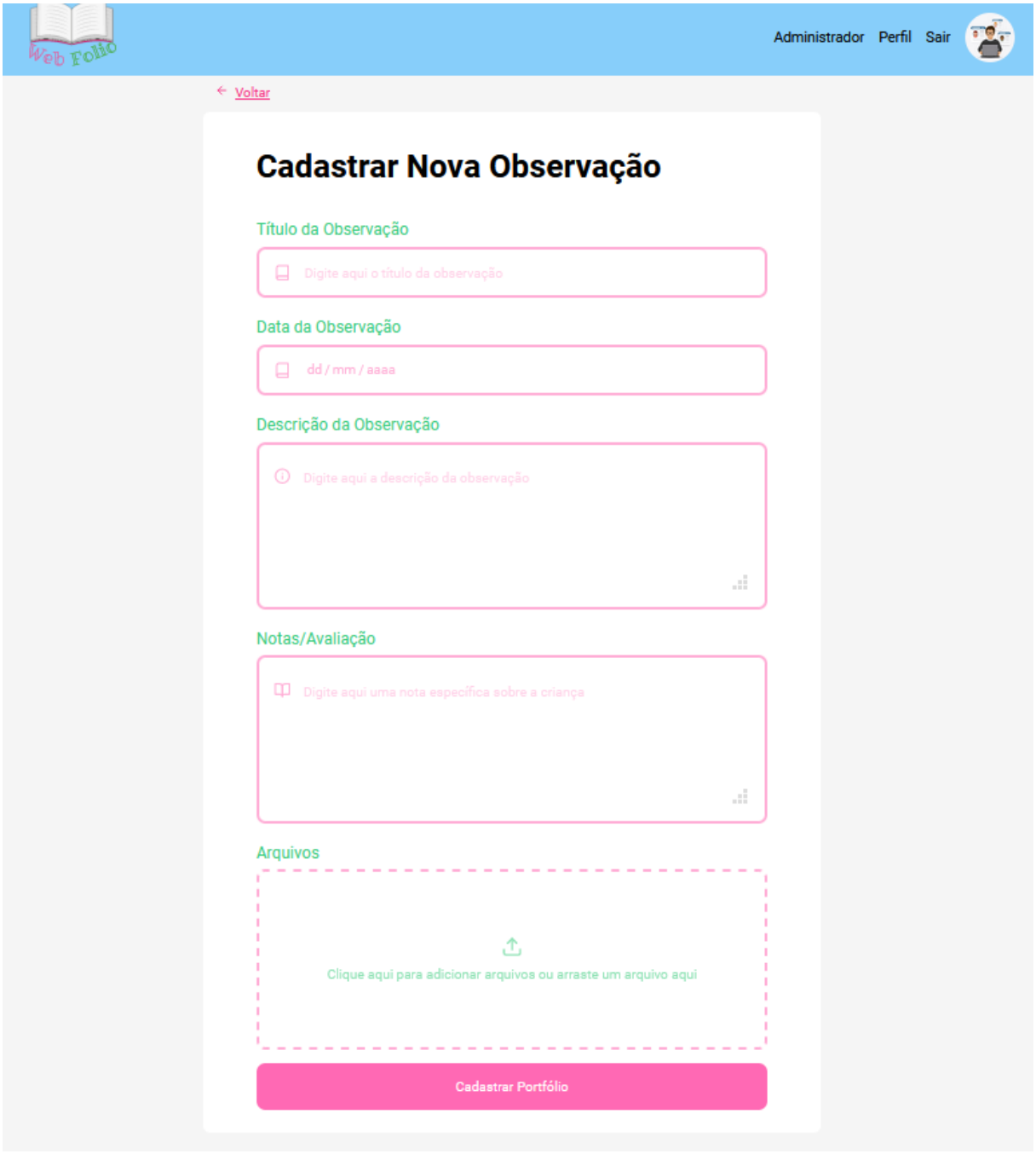

Figura 4.16: Interface da página de criar observação da aplicação.

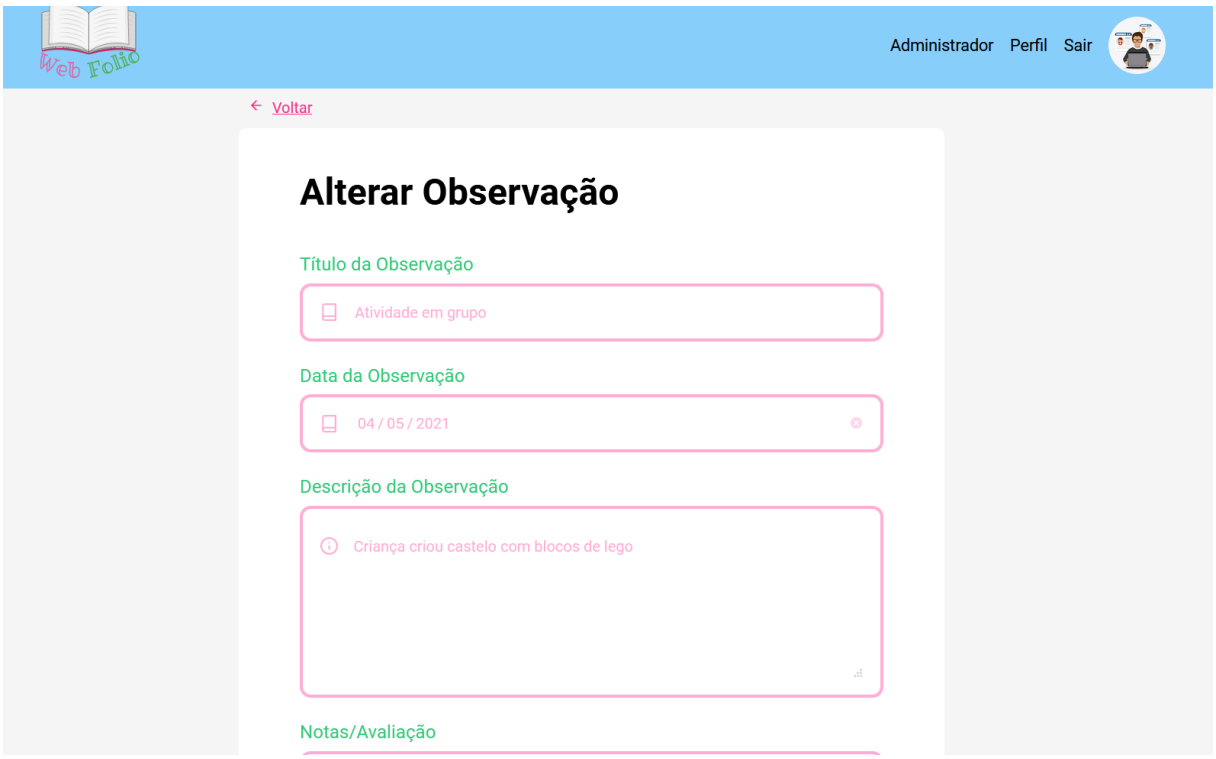

Figura 4.17: Interface da página de alterar uma observação da aplicação.

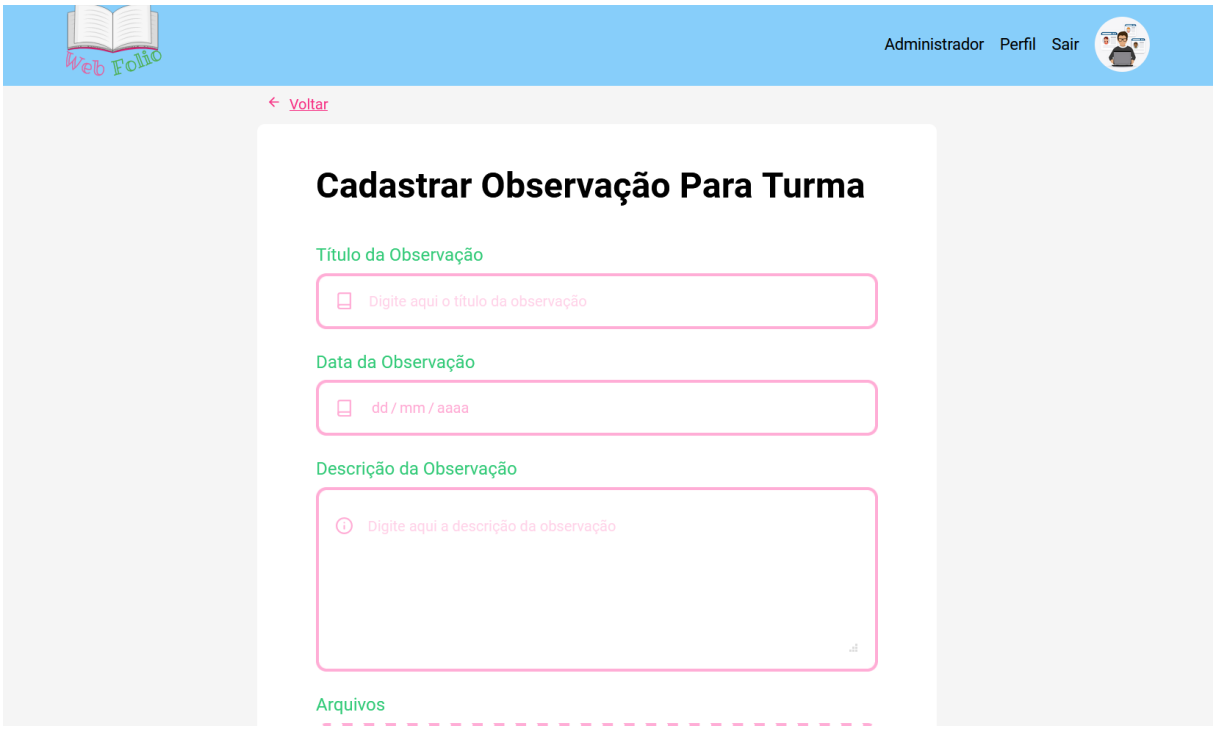

Figura 4.18: Interface da página de criar observação para um grupo.

Figura 4.19: Caixa de seleção do grupo escolhido.

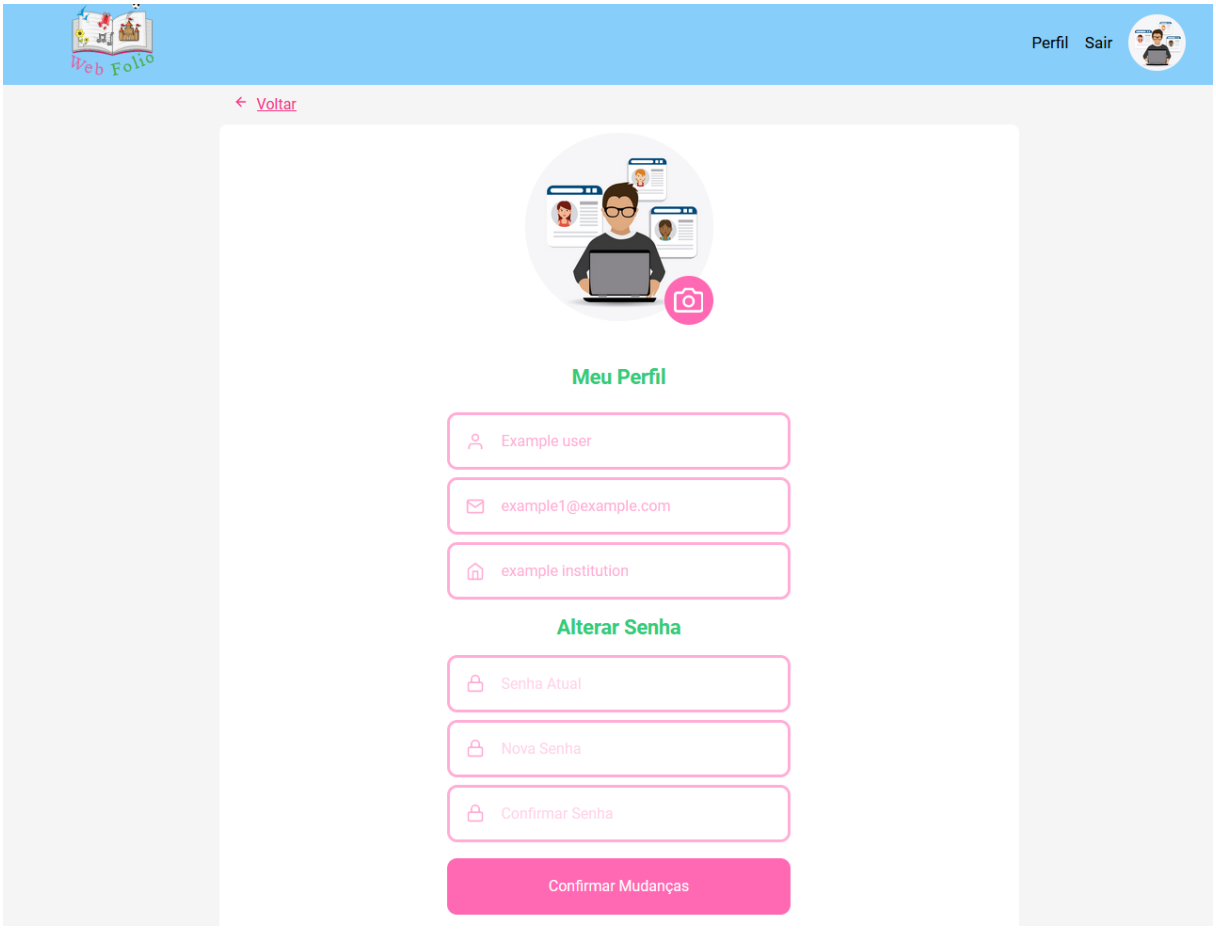

Figura 4.20: Interface da página de perfil do utilizador no sistema.

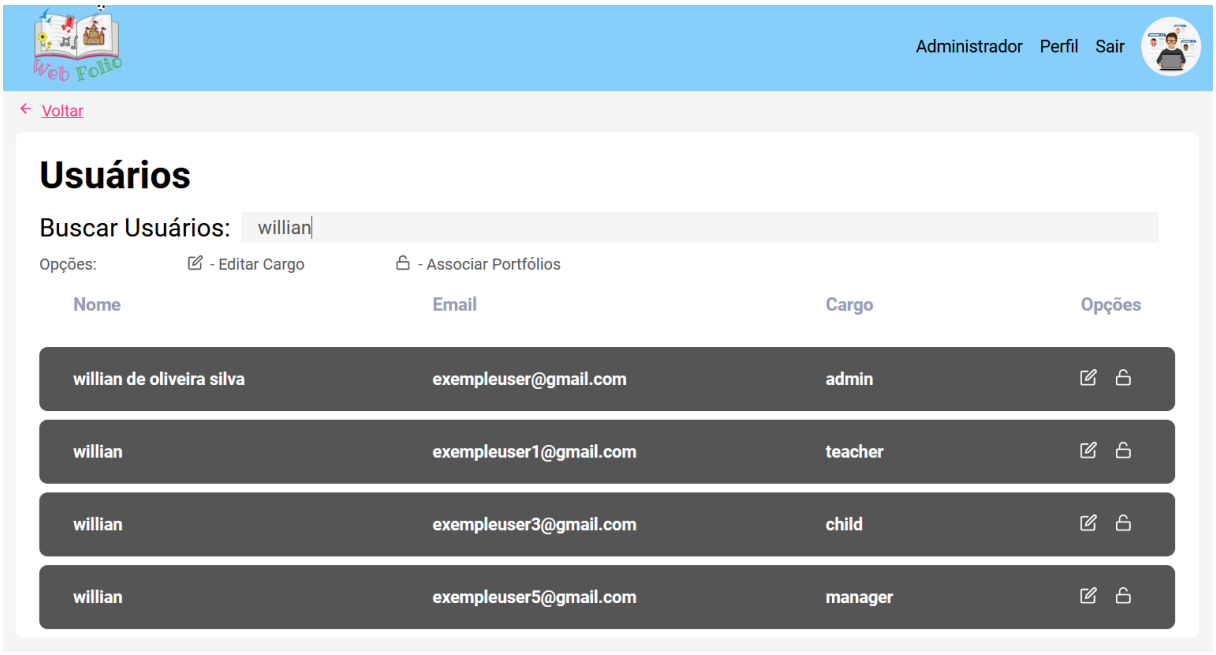

Figura 4.21: Interface da página de *Dashboard* administrativo.

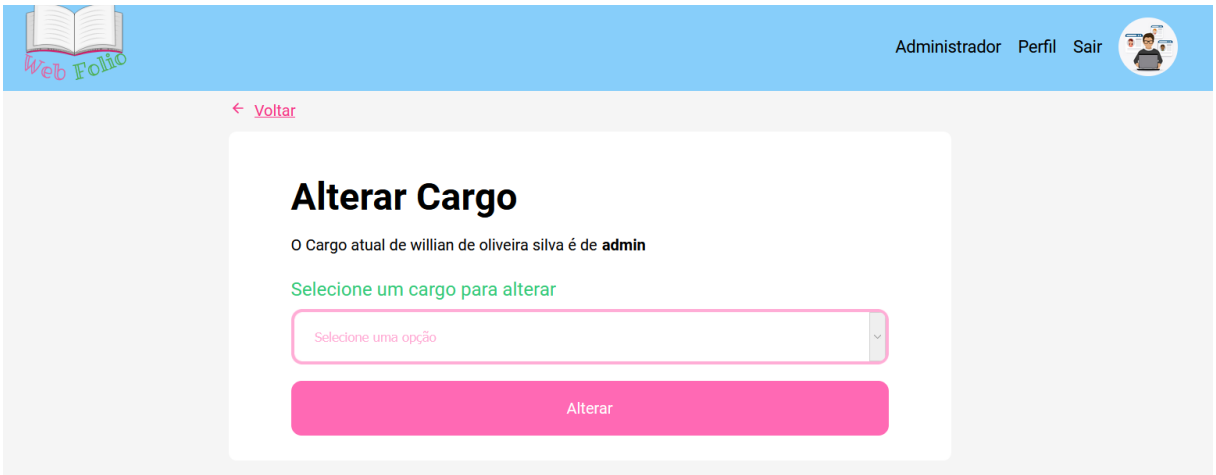

Figura 4.22: Interface da página de atualizar cargo de um utilizador.

aquele portfólio e o terceiro ícone indica deletar o portfólio da aplicação.

Figura 4.23: Interface da página de associação de portfólios na aplicação.

Para solucionar responsividade do *layout* no sistema, foi utilizado Media Queries para adaptar o uso da aplicação em telemóveis, desta forma toda a navegação no sistema pode ser feita também por telemóvel.

Para as páginas de *login* e cadastro de utilizador, figura 4.24, o conteúdo na página foi limitado para conter somente o logo da aplicação e o formulário para a inserção das informações do utilizador.

Para a página de *Dashboard* da versão *mobile*, figura 4.25, os botões para criar portfólio e criar observação para turma foram substituídos por botões com ícones para manter o *layout* padrão da aplicação. Na Lista de portfólios da tabela, os ícones das opções, nomes e turmas agora são exibidas em bloco, uma abaixo da outra, para facilitar sua visualização.

Na página de portfólio para a versão telemóvel, figura 4.26, as informações agora também são exibidas em bloco, além disso, o botão de criar uma nova observação foi

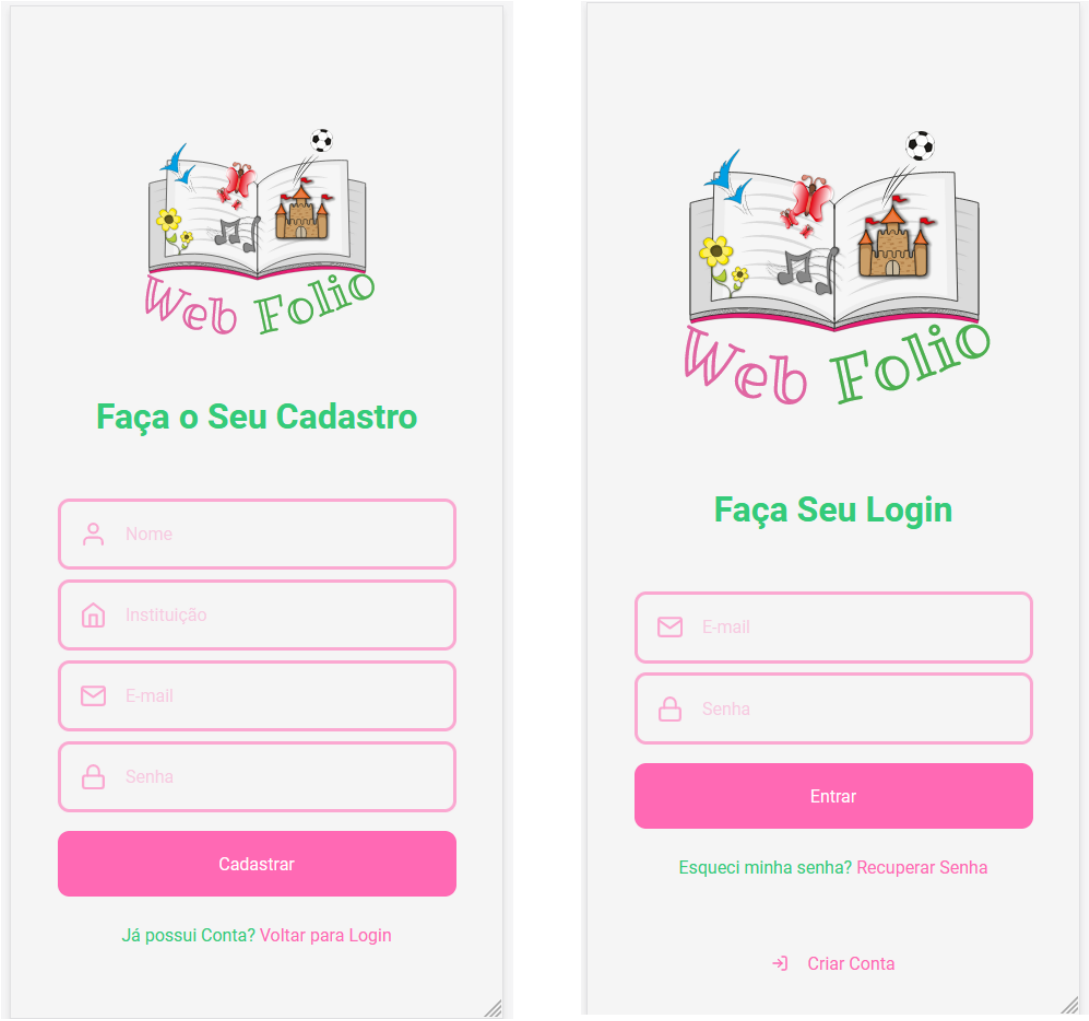

Figura 4.24: Páginas de *login* e cadastro de utilizador da versão telemóvel.

Figura 4.25: Página de *Dashboard* da versão mobile.

substituído por outro botão com ícone para também manter o *layout* padrão da aplicação.

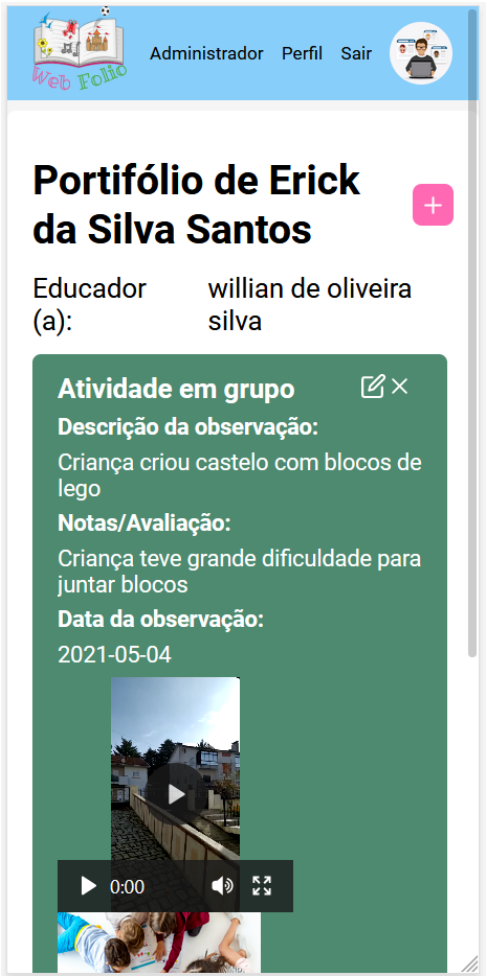

Figura 4.26: Página de portfólio da versão telemóvel.

Para todos os formulários do sistema para a versão para telemóvel, como por exemplo na figura 4.27, não houveram mudanças em seu conteúdo, houve somente o redimensionamento do conteúdo para manter o padrão de *layout* das páginas que possuem formulários.

A página de administração e a página de associação de portfólios, figura 4.28, tiveram mudanças somente em sua apresentação, na qual os ícones e informações dos portfólios agora são exibidas em bloco.

Figura 4.27: Página de cadastrar uma nova observação e alterar uma observação da versão para telemóvel.

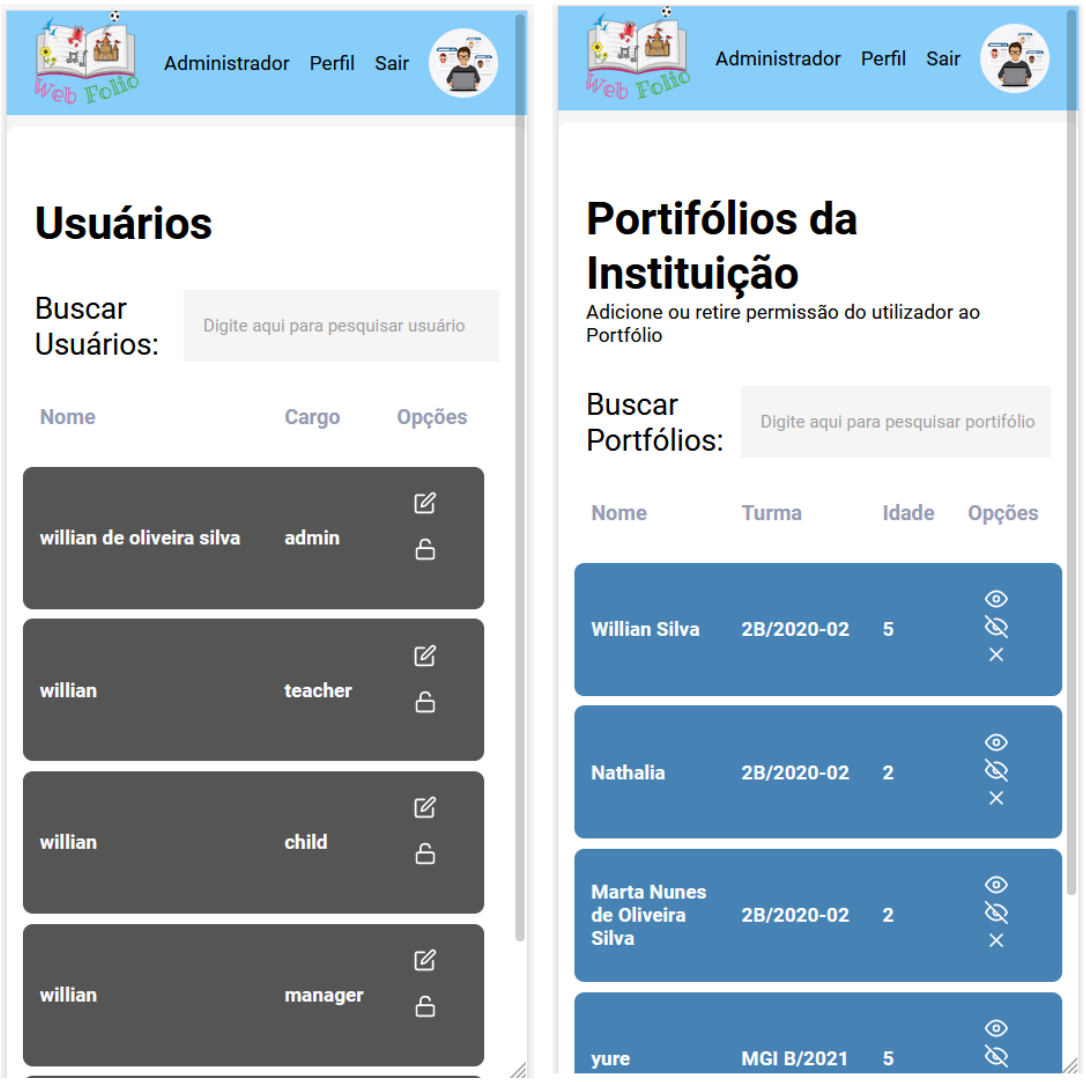

Figura 4.28: Página de *Dashboard* administrativo e de associação de portfólios versão telemóvel.

Capítulo 5

Testes e Discussão

Este capítulo apresenta os testes realizados para verificar que o projeto desenvolvido cumpre os objetivos assumidos e resolve, de facto, o problema descrito na Análise e modelação.

5.1 Criando Observações

Existem duas formas para criar observações no sistema. A primeira forma é através de um portfólio de uma criança. Desta forma, o sistema cria uma observação somente para aquela criança, com informações como título da observação, sua descrição, data da atividade, notas como uma forma de avaliação específica para a criança, além de arquivos, como áudio ou vídeo, sobre a atividade.

A segunda forma consiste em criar uma observação para uma ou mais crianças ou para um ou mais grupos. Nesta opção, todas as informações podem ser enviadas, exceto a informação sobre as notas ou avaliação da criança, pois se trata de uma informação exclusiva para cada criança. A informação sobre notas ou avaliação da criança pode ser posteriormente editado pelo utilizador e adicionado para a observação da criança.

5.2 Associando Portfólios

Existem duas formas de associar um utilizador do sistema aos portfólios e assim permitir o seu acesso a um portfólio do sistema. A primeira forma de associar um utilizador a um portfólio é através de um convite enviado por um utilizador de cargo "Professor", para convidar um utilizador por este método, é necessário a inserção do e-mail do utilizador que irá visualizar o portfólio da criança. Quando ocorre esta permissão, qualquer utilizador que tenha sido incluído desta forma no sistema, possui a permissão para somente visualizar o portfólio da criança, não podendo alterar nenhuma informação contida neste portfólio, independentemente de seu cargo no sistema.

A segunda forma de associar um utilizador aos portfólios do sistema é através da associação feita por um utilizador do cargo "Administrador", desde que o utilizador e o portfólio sejam da mesma instituição, a associação pode ser feita ou desfeita. Desta forma o utilizador associado ao portfólio pode exercer as funções correspondentes ao seu cargo no sistema, como alterar informações no portfólio, ou criar observações.

Um utilizador pode conter as duas formas de associações no sistema. Desta forma, por exemplo, um utilizador do cargo "Professor" pode conter na aplicação, portfólios criados por ele, com suas devidas permissões, pode conter acesso a portfólios de outros utilizadores do cargo "Professor" da mesma instituição, na qual permite a ele exercer também suas funções de "Professor" sobre estes portfólios, e além disso ele pode ser convidado através da funcionalidade de associação através de e-mail e ter acesso a outros portfólios, até mesmo de outras instituições, porém, com o seu acesso limitado para somente visualizar o portfólio da criança. Caso haja uma alteração na informação de e-mail do utilizador convidado, ele deve ser convidado novamente com o novo e-mail para obter novamente a permissão para poder visualizar o portfólio.

5.3 Testes

Foram realizados testes de usabilidade para verificar as funcionalidades da interface e para avaliar a experiência do utilizador na aplicação. Os testes ocorreram após o término do desenvolvimento de todas as interfaces do sistema.

Foram selecionadas três participantes para realizar os testes no sistema, todas as participantes possuem diplomas do ensino superior com formação acadêmica em pedagogia. Os testes foram realizados através do uso da ferramenta TeamViewer [41], software de acesso remoto e controle remoto, na qual os participantes utilizaram desta ferramenta para acessar remotamente ao computador com a aplicação Webfolio. Além disso foi proposto o uso de um formulário com atividades descritas, para serem assinaladas quando são realizadas pelo utilizador enquanto navega pelo sistema, auxiliando o participante no uso de todas as funcionalidades do sistema. As participantes puderam testar todas as funcionalidades com a permissão do cargo "Professor".

Através dos testes foi possível identificar e corrigir eventuais problemas existentes no sistema, como erros de quebra de linha em portfólios, não permitir a criação de uma observação para turma se não houver um portfólio selecionado, adição de novas informações como a data da atividade, substituir ícones para melhor representar a funcionalidade, alterar a ordem de ícones no sistema e criar um tutorial para exibir as funcionalidades dos ícones do sistema, como pode ser visto na figura 5.1.

Todos os testes foram gravados, e estão disponíveis para acesso e visualização no endereço "https://drive.google.com/drive/folders/1Hv3mOKqZswUierLnOjW0AiuYLOCq7d4G?usp=sharing".

5.4 Deploy

Para realizar o *deploy* da aplicação, foi utilizado a plataforma Heroku [42], plataforma em nuvem como um serviço capaz de suportar diversas linguagens de programação, tanto para o *back-end* quanto para o *front-end* da aplicação. A escolha desta plataforma se

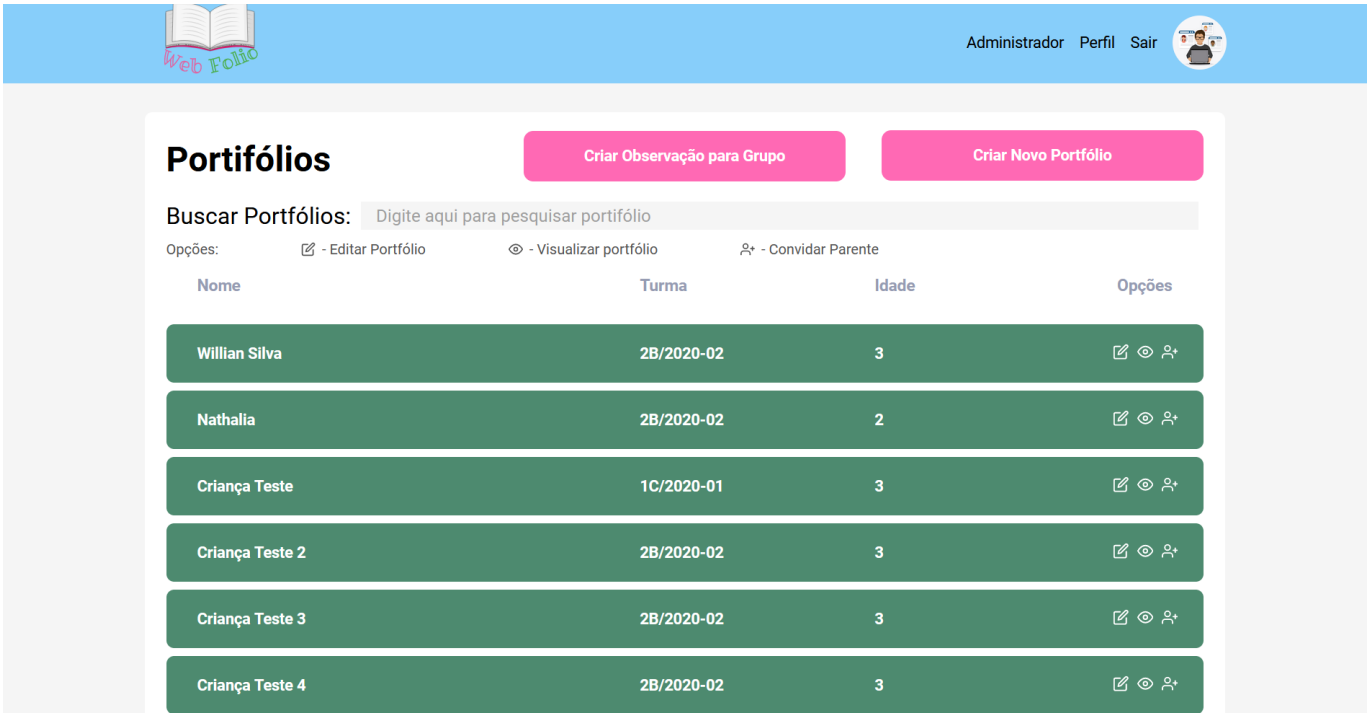

Figura 5.1: Página de *Dashboard* com tutorial.

deu pelo fato de ser mais simples e gratuita. A aplicação foi disponibilizada no endereço "https://frontend-webfolio.herokuapp.com/", porém não conta com serviços e funcionalidades de recuperação de senha, convite para visualizar portfólio e envio de arquivos.

Para realizar o *deploy* da aplicação TypeScript com React, que foi desenvolvida utilizando o ambiente *create react app*, foi necessário preparar a aplicação para entrar em ambiente de produção, criando uma variável ambiente para realizar comunicação com a API e a inserindo na plataforma Heroku. Posteriormente é necessário interligar a conta da aplicação Heroku com a conta GitHub, e então selecionar o repositório GitHub que contém a aplicação *front-end* Webfolio. A plataforma Heroku então aciona o script de *build* na aplicação, que assim converte todo o código TypeScript para JavaScript legível para *browser*.

Para realizar o *deploy* da aplicação Node.js, que também foi utilizado a plataforma Heroku, novamente foi necessário criar variáveis ambientes para e as inserir na plataforma Heroku, desta vez foram usadas variáveis ambiente para comunicação do banco de dados

MongoDB, uma para guardar a senha segredo do JWT *token*, uma variável ambiente para guardar o URL da aplicação, e outras variáveis ambiente para guardar senha, porta e *host* do banco de dados Redis. É necessário interligar outra vez a conta da aplicação Heroku com a conta GitHub, e então selecionar o repositório GitHub que contém a aplicação *back-end* Webfolio.

5.5 Discussão

Acredita-se que poderia ser utilizado banco relacional SQL para um melhor tratamento das relações de controle de acesso dos utilizadores. Em questão de testes, poderia ter sido implementado testes automatizados para reduzir erros e reduzir os testes necessários para verificar se todas as funcionalidades estavam corretas na aplicação.

A funcionalidade que permite a criação de uma observação para mais de um portfólio possibilitando a redução de tempo gasto para o desenvolvimento dos portfólios no sistema e o uso de Media Queries para possibilitar o uso da aplicação em dispositivos *mobile* são os pontos onde se foi para além dos objetivos iniciais.

Capítulo 6

Conclusões

Neste capítulo é apresentado as conclusões e conhecimentos gerados relacionados com o trabalho desenvolvido, além de sugestões para possíveis trabalhos futuros.

Considerando que o principal objetivo deste trabalho baseou-se no desenvolvimento de uma ferramenta *web* para auxílio na criação de portfólio para educação infantil, permitindo o envio de diversos tipos de arquivo, a aplicação provou-se útil e pronta para uso. Para além disso, esperava-se que a ferramenta permitisse que utilizadores tenham diferentes papéis a serem desempenhados na aplicação e que também houvesse um método de controle sobre eles. Desta forma, acredita-se que o sistema desenvolvido cumpre satisfatoriamente os objetivos propostos e espera-se que a aplicação possa auxiliar e facilitar o registro dos professores para com o no acompanhamento e desenvolvimento das crianças nas instituições de ensino.

No sistema desenvolvido, a maior dificuldade enfrentada foi devido a implementação de um controle de acessos aos utilizadores, pois sua arquitetura provou ser de difícil implementação. Este facto encontra-se associado à necessidade de fazer autenticação e controlo de acesso ao nível de cada recurso, o que requer um mecanismo muito flexível e eficiente para verificar as permissões de acesso.

Para trabalhos futuros são sugeridos uma funcionalidade para ocultar a exibição de uma observação de um portfólio que ainda não está completa, aplicar internacionalização

na aplicação para permitir o uso da aplicação em outros idiomas, e o uso de ferramentas para avaliar e impedir arquivos enviados para a aplicação que contenham conteúdo inapropriado para sua visualização ou reprodução.

Bibliografia

[1] E. Shores, *Manual de portfólio : um guia passo a passo para o professor*. Porto Alegre: Artmed, 2001, ISBN: 9788573077629.

[2] V. F. A. Neves e C. Moro, "Avaliação na educação infantil: um debate necessário", *Estudos em avaliacao educacional*, vol. 24, n.º 55, pp. 272–303, 2013.

[3] C. Mesquita e R. P. Lopes, "Preschool observation supported by smartphone applications", *INNODOCT/13 New changes in technology and innovation*, 2013.

[4] Childdiary, *Quem mais quer poupar 1 hora por dia com a comunicação e documentação pedagógica?*, `https://childdiary.net/pt/`, [Online; accessed 07-March-2021], 2020.

[5] Alison, *How does Seesaw work?*, `https://help.seesaw.me/hc/en-us/articles/115003755186-How-does-Seesaw-work-`, [Online; accessed 07-March-2021].

[6] Educlipper, *educlipper*, `https://edshelf.com/tool/educlipper/`, [Online; accessed 07-March-2021], 2021.

[7] T. Ring, *Three Ring*, `https://sites.google.com/site/mportfolios/tools-apps/three-ring`, [Online; accessed 07-March-2021].

[8] E. Portfolio, *Easy Portfolio*, `https://edshelf.com/tool/easy-portfolio/`, [Online; accessed 07-March-2021], 2021.

[9] D. Flanagan, *JavaScript: o guia definitivo*. Bookman Editora, 2004.

[10] M. S. Silva, *JavaScript-Guia do Programador: Guia completo das funcionalidades de linguagem JavaScript*. Novatec Editora, 2010.

[11] N. C. Zakas, *Understanding ECMAScript 6: the definitive guide for JavaScript developers*. No Starch Press, 2016.

[12] J. Duckett, G. Ruppert e J. Moore, *JavaScript & jQuery: interactive front-end web development*. Wiley, 2014.

[13] D. Jacobson, D. Woods e G. Brail, *APIs: A strategy guide*. "O'Reilly Media, Inc.", 2011.

[14] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. "O'Reilly Media, Inc.", 2011.

[15] L. Richardson, M. Amundsen, M. Amundsen e S. Ruby, *RESTful Web APIs: Services for a Changing World*. "O'Reilly Media, Inc.", 2013.

[16] L. Richardson e S. Ruby, *RESTful web services*. "O'Reilly Media, Inc.", 2008.

[17] D. Gourley, B. Totty, M. Sayer, A. Aggarwal e S. Reddy, *HTTP: the definitive guide*. "O'Reilly Media, Inc.", 2002.

[18] M. Amundsen, *Building Hypermedia APIs with HTML5 and Node: Creating Evolvable Hypermedia Applications*. "O'Reilly Media, Inc.", 2011.

[19] C. R. Pereira, *Aplicações web real-time com Node. js*. Editora Casa do Código, 2014.

[20] G. Lim, *Beginning Node.js, Express & MongoDB Development*. Independently Published, 2019, ISBN: 9781078379557. URL: `https://books.google.pt/books?id=q-tjxwEACAAJ`.

[21] C. W. Krueger, "Software reuse", *ACM Computing Surveys (CSUR)*, vol. 24, n.º 2, pp. 131–183, 1992.

[22] D. Roberts, R. Johnson et al., "Evolving frameworks: A pattern language for developing object-oriented frameworks", *Pattern languages of program design*, vol. 3, pp. 471–486, 1996.

[23] E. Brown, *Web development with node and express: leveraging the JavaScript stack*. O'Reilly Media, 2019.

[24] Express, *Express*, `https://expressjs.com/`, [Online; accessed 10-March-2021].

[25] A. Meier e M. Kaufmann, *SQL & NoSQL databases*. Springer, 2019.

[26] K. Chodorow, *MongoDB: the definitive guide: powerful and scalable data storage*. "O'Reilly Media, Inc.", 2013.

[27] R. Copeland, *MongoDB Applied Design Patterns: Practical Use Cases with the Leading NoSQL Database*. "O'Reilly Media, Inc.", 2013.

[28] A. Banks e E. Porcello, *Learning React: functional web development with React and Redux*. "O'Reilly Media, Inc.", 2017.

[29] C. Gackenheimer, *Introduction to React*. Apress, 2015.

[30] Typescript, *O que é TypeScript?*, `http://www.typescriptlang.org/`, [Online; accessed 07-March-2021], 2021.

[31] A. Freeman, *Essential TypeScript*. Springer, 2019.

[32] B. Cherny, *Programming TypeScript: making your JavaScript applications scale*. O'Reilly Media, 2019.

[33] M. Ivanov, *Fullstack React with TypeScript: Learn Pro Patterns for Hooks, Testing, Redux, SSR, and GraphQL*.
, 2020.

[34] M. Jones, B. Campbell e C. Mortimore, "JSON Web Token (JWT) profile for OAuth 2.0 client authentication and authorization Grants", *May-2015.{Online}. Available: https://tools. ietf. org/html/rfc7523*, 2015.

[35] D. Hardt et al., *The OAuth 2.0 authorization framework*, 2012.

[36] D. Ferraiolo, D. R. Kuhn e R. Chandramouli, *Role-based access control*. Artech House, 2003.

[37] Figma, *Figma*, `https://www.figma.com/`, [Online; accessed 20-March-2021].

[38] A. C. Cloud, *Adobe Illustrator*, `https://www.adobe.com/pt/`, [Online; accessed 10-March-2021].

[39] Redis, *Redis*, `https://redis.io/`, [Online; accessed 20-March-2021].

[40] GitHub, *GitHub*, `https://github.com/`, [Online; accessed 20-March-2021].

[41] TeamViewer, *Teamviewer*, `https://www.teamviewer.com/en/`, [Online; accessed 20-March-2021].

[42] Heroku, *Heroku:Cloud Application Platform*, `https://www.heroku.com/`, [Online; accessed 20-March-2021].