REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022001663-0**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 04/04/2022, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: RepositORE

Data de publicação: 04/04/2022

Data de criação: 04/04/2022

Titular(es): FUNDAÇÃO UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - FUERN

Autor(es): SEBASTIÃO EMIDIO ALVES FILHO; THALIA KATIANE SAMPAIO GURGEL; KEFTON DAVID NUNES DE MELO; CLEYTON CARLOS SANTOS COSTA

Linguagem: HTML; JAVA SCRIPT; OUTROS

Campo de aplicação: ED-03; IF-02; IF-04

Tipo de programa: AP-01; GI-01

Algoritmo hash: SHA-512

Resumo digital hash:
680326749e0c6e3dc00a542ffe75ba67a79fe9500535c4291c2508941075fb13a38c7be1104e9be646941dacb74454d084
97951e5b00ab0a6ead821b745d43c0

Expedido em: 12/07/2022

Aprovado por:
Joelson Gomes Pequeno
Chefe Substituto da DIPTO - PORTARIA/INPI/DIRPA Nº 02, DE 10 DE FEVEREIRO DE 2021

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT
DEPARTAMENTO DE INFORMÁTICA – DI

KEFTON DAVID NUNES DE MELO

REPOSITORE: PERSPECTIVA DO BACK-END

MOSSORÓ - RN
2022

KEFTON DAVID NUNES DE MELO

REPOSITORE: PERSPECTIVA DO BACK-END

Relatório apresentado ao curso de Ciência da Computação da Universidade do Estado do Rio Grande no Norte como requisito da disciplina de Trabalho de Diplomação, sob a orientação do(a) Prof. Dr. Sebastião Emidio Alves Filho.

MOSSORÓ - RN

2022

KEFTON DAVID NUNES DE MELO

REPOSITORE: PERSPECTIVA DO BACK-END

Registro de software apresentado como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovada em: 27/04/2022

Banca Examinadora

Prof. Dr. Sebastião Emídio Alves Filho
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Me. Ceres Germanna Braga Morais
Universidade do Estado do Rio Grande do Norte - UERN

Me. Thalia Katiane Sampaio Gurgel
Universidade do Estado do Rio Grande do Norte - UERN

SUMÁRIO

1. INTRODUÇÃO

O uso da robótica no ensino de alunos, por meio de estudos, programação, montagem e implementação da robótica através de kits de robótica, denomina-se robótica educacional. A robótica educacional tem como objetivo permitir o desenvolvimento do conhecimento teórico e prático do aluno, como também ajuda no desenvolvimento de outras habilidades cognitivas, tais como produtividade, interação, resolução de problemas, autonomia e entre outros..

Uma das maneiras de aplicação de conteúdos de robótica educacional se dá por meio de Objetos de Aprendizagem (OAs). Objetos de Aprendizagem são conteúdos estruturados, digitais ou não, destinados ao ensino e desenvolvimento de alunos.

Os objetos de aprendizagem de robótica educacional muitas vezes utilizam kits de robótica. Os diversos kits de robótica existentes são compostos por motores, sensores, linguagens de programação e etc, possibilitando uma grande variedade de tipos de atividades de robótica aplicáveis ao ensino.

1.1 JUSTIFICATIVA

Gurgel(2019) realizou uma pesquisa que apontou a existência de diversos conteúdos de robótica na internet, porém a grande maioria dos conteúdos está distribuída de maneira desorganizada e dispersa. A pesquisa também constatou que existe um déficit de plataformas de compartilhamento de OAs de robótica educacional que possuam descrição de campos através de metadados, dificultando mais o reconhecimento de conteúdos de robótica por parte dos motores de busca

Devido ao déficit de plataformas de robótica com descrição de metadados em campos de OAs de robótica educacional, foi conceitualizado por Gurgel(2019) o RepositORE, um repositório de Objetos de Robótica Educacional com a aplicação do padrão de metadados Dublin Core para os campos dos objetos, sendo uma plataforma que poderia ser utilizada por professores, alunos e entusiastas da robótica para compartilhar e encontrar OAs.

Uma versão protótipo do RepositORE foi implementada por Gurgel(2019) durante sua pesquisa, porém não tendo sido disponibilizada ao público. Uma continuação do RepositORE, aplicação de novas funcionalidades, mudanças no tratamento de alguns campos do padrão Dublin core e a atualização das versões das tecnologias utilizadas mostrou-se necessária, implicando também na necessidade de reconstrução do sistema, devido às tecnologias Angular e Firebase, que são as principais tecnologias do sistema, terem sofrido várias alterações que impossibilitaram a compatibilidade com a versão inicial da plataforma.

1.2 OBJETIVOS

O objetivo deste trabalho é a implementação do backend da plataforma RepositORE, permitindo o armazenamento e o acesso de dados de forma confiável e eficiente, a adição de novas funcionalidades, a garantia de compatibilidade com o padrão Dublin Core revisado proposto por Gurgel(2019) e a hospedagem da plataforma, permitindo que quaisquer usuários possam utilizar o RepositORE.

1.3 TRABALHOS RELACIONADOS

O RepositORE utiliza uma descrição de campos dos Objetos de Aprendizagem com metadados. Durante o estudo de Gurgel(2019) foi feito um levantamento sobre diferentes padrões de metadados, sendo que dentre os padrões selecionados para comparação, foi selecionado o padrão de metadados Dublin Core, devido a sua simplicidade, flexibilidade e extensibilidade. No decorrer do estudo, foi feita uma pesquisa e comparação de diferentes trabalhos que aplicavam o padrão de descrição de metadados Dublin Core.

Foi feito por Gonçalves *et al.(2013)* uma adaptação do padrão de metadados Dublin Core ao caso da Biblioteca Digital da Assembleia Legislativa do Estado de Minas Gerais. O padrão Dublin Core foi escolhido pelos autores devido à capacidade de adaptação e extensibilidade do padrão, permitindo uma adaptação de metadados ao Código de Catalogação Anglo Americano 2 (AACR2).

Roseto & Nogueira(2002) realizaram um trabalho que consistiu no uso do padrão Dublin Core para a descrição de conteúdos na biblioteca digital de teses da USP. O padrão acabou sendo escolhido pelos autores devido sua extensibilidade, o

que permitiu a personalização do Dublin Core para a aplicação desejada pelos autores.

O padrão Dublin Core foi utilizado para a Descrição de Obras Raras na Web: A Coleção da Biblioteca Brasiliana Digital, proposto por Pires(2012). Também foi feita uma adaptação do padrão às necessidades do autor.

Tendo em vista a capacidade de extensão e adaptação do padrão Dublin Core, Gurgel(2019) propôs em seu estudo uma revisão e adaptação do padrão para o cadastro de objetos de robótica educacional. Esses campos revisados foram utilizados durante o cadastro e exibição de dados.

2. ARQUITETURA DO SISTEMA

O RepositORE é uma aplicação web, desenvolvida com Angular, utilizando primariamente Typescript, Javascript, Html e Css.

Páginas do sistema:

- Página principal: Essa página dá uma pequena introdução ao sistema, e exibe os cinco últimos objetos de aprendizagem de robótica educacional postados por usuários no sistema. A página principal é exibida para qualquer usuário, mesmo que ele não tenha uma conta no RepositORE.
- Login: Nesta página, o usuário pode fazer login no sistema através de uma conta Google ou com uma conta de e-mail criada no sistema. Caso não possua conta de e-mail, ele pode selecionar a opção de criar uma conta e ser redirecionado para a página de criação de conta.
- Criação de conta: Na página de criação de conta, o usuário pode criar sua conta usando um e-mail qualquer.
- Cadastro de objetos: É nessa página que o usuário pode cadastrar e postar um objeto de aprendizagem de robótica educacional com descrição de metadados. Essa página só é disponibilizada para usuários logados no sistema.
- Lista de objetos: A listagem de objetos mostra para o usuário que esteja logado no sistema todos os objetos postados por ele mesmo e por outros usuários. É possível buscar um objeto na lista por busca textual, como

também é possível ordenar a listagem por objetos mais recentes ou mais antigos. Cada objeto de aprendizagem postado possui três interações disponíveis, sendo elas: mostrar detalhes, atualizar objeto e excluir objeto

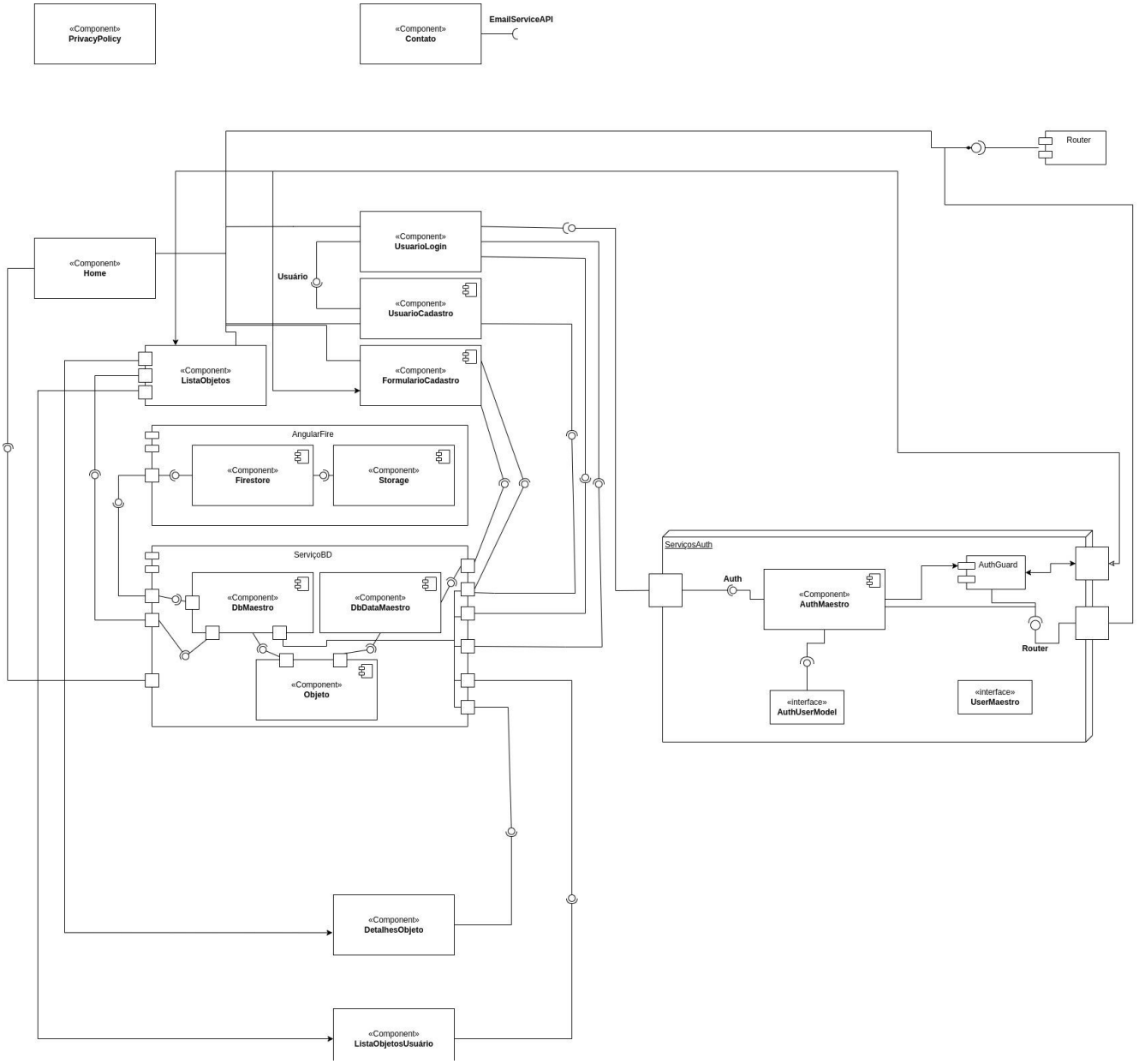
- Pesquisar objeto: Permite ao usuário usar a busca textual para pesquisar um objeto de robótica do seu interesse.
- Mostrar detalhes: Ao clicar em mostrar detalhes, o usuário pode visualizar o objeto com todos os metadados inseridos durante o cadastro do objeto.
- Atualizar objeto: Permite ao usuário a edição de um objeto já existente postado. Ao clicar em “atualizar objeto”, o usuário é redirecionado para a página de cadastro de objeto, porém agora com os campos a serem editados.
- Deletar objeto: O usuário pode usar essa funcionalidade para excluir um objeto da listagem.
- Curtir ou descurtir objeto: O usuário pode deixar uma pequena avaliação em forma de curtir ou descurtir no objeto de seu interesse.
- Contate-nos: Nesta página, o usuário pode enviar um e-mail diretamente para os desenvolvedores com uma mensagem personalizada através de um formulário. O formulário contém um campo para o usuário colocar o seu nome, um para o e-mail e outro campo para a mensagem que ele escreverá para os desenvolvedores. Ao clicar em “enviar mensagem” é enviado um e-mail para o e-mail do GTEC, com o que foi digitado nos campos.
- Lista de objetos relacionados: O usuário pode visualizar os objetos relacionados com o objeto no qual se está vendo os detalhes.
- Lista de objetos do usuário: O usuário pode visualizar todos os componentes postados e editados por ele mesmo nessa página. Esta página é exibida dentro da página de Login, caso o usuário não esteja logado.

2.1 DIAGRAMA DE COMPONENTES

Na **figura 1** está descrito e demonstrado o diagrama de componentes do sistema.

Dentre as partes mais importantes estão os núcleos de sistema ServiçoAuth (autenticação e controle de sessões de usuários) e ServiçoBD (conexão com o banco de dados), que são os blocos onde ocorrem as transações com os serviços do AngularFire, que é um pacote de integração do Angular com o Google Firebase (ANGULARFIRE, 2022). Também temos a interface Router que provê navegação no plano de fundo. O restante dos blocos representa as outras páginas acessíveis do sistema, sendo elas: página principal, página de login, página de cadastro de usuário, página de objetos do usuário, página de cadastro de Objetos de Aprendizagem, página de listagem de objetos, página de detalhe dos objetos e página de política de privacidade. A página de contato conecta-se com uma API integrada com o serviço de envio de e-mails do Sendgrid.

Figura 1 - Diagrama de Componentes



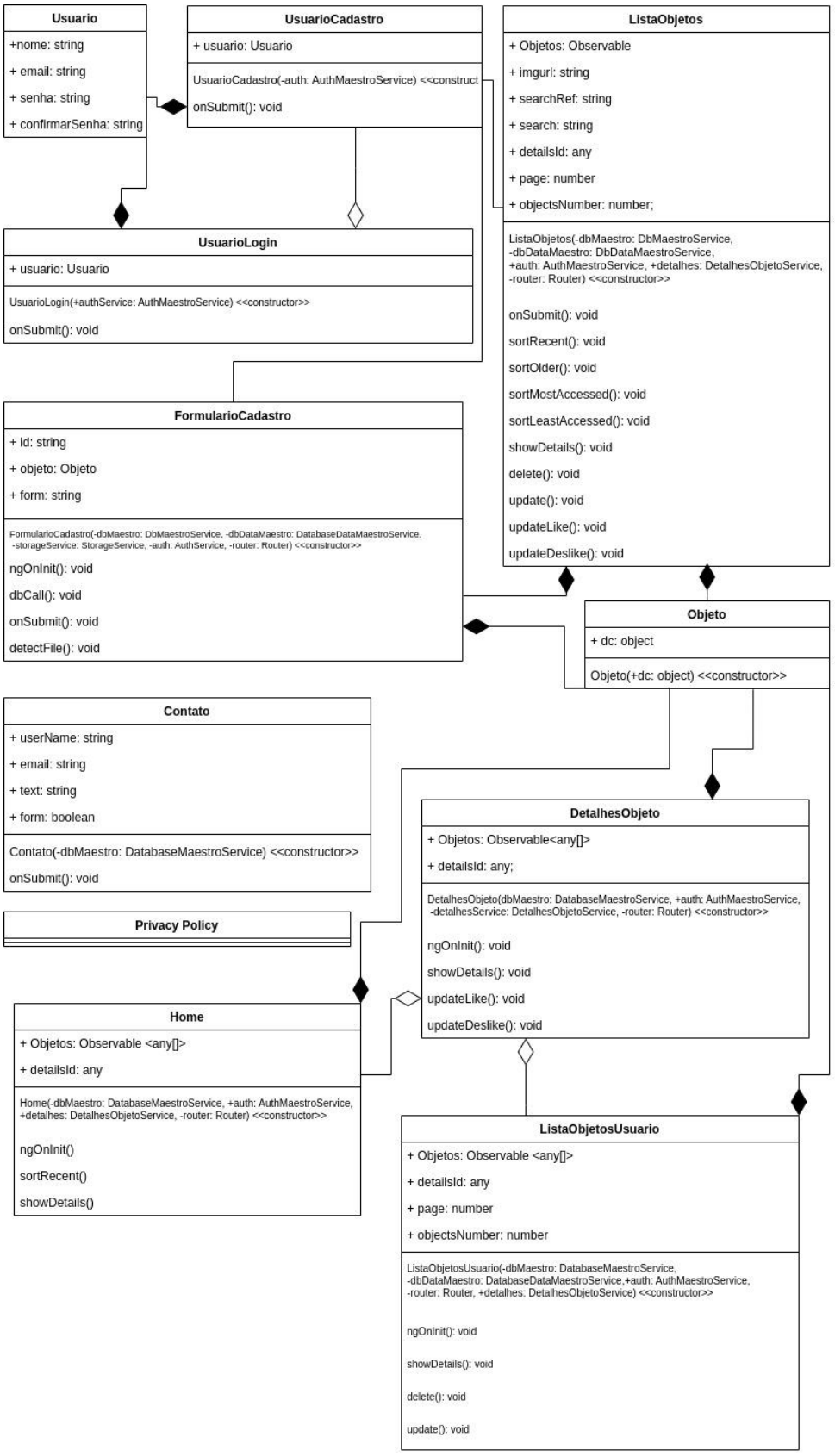
Fonte: Autoria Própria

2.2 DIAGRAMA DE CLASSES

Na **figura 2** é possível visualizar o diagrama de classes do sistema. O diagrama de classes mostra quais são as principais classes do sistema. Classes do sistema: Usuario, UsuarioCadastro, UsuarioLogin, ListaObjetos, FormularioCadastro, Objeto, DetalhesObjeto, ListaObjetosUsuario, Home, PrivacyPolicy e Contato. Cada classe possui um conjunto de atributos e métodos.

- Objeto: É uma classe que define todos os campos dos objetos de aprendizagem a serem postados no sistema. Os campos dessa classe são atrelados ao padrão de metadados Dublin Core.
- ListaObjetos: É a classe responsável por mostrar os objetos cadastrados no banco.
- FormularioCadastro: Essa é a classe responsável pela criação e tratamento dos dados de um Objeto de Aprendizagem, e por colocá-lo no banco de dados.
- DetalhesObjeto: É responsável por gerar a página de detalhes de cada objeto.
- ListaObjetosUsuario: Classe semelhante à ListaObjetos, mas específica em exibir somente os objetos criados pelo próprio usuário.
- Home: É a página inicial do sistema, que mostra os cinco últimos objetos cadastrados.
- Contato: Classe geradora da página de contato, que permite um portal de fácil acesso com a gestão do sistema por meio do envio de e-mails.
- UsuarioCadastro: Página de cadastro de conta dos usuários com email e senha.
- UsuarioLogin: É a página na qual o usuário pode autenticar-se no sistema através de uma conta com email criada na plataforma, ou alguma Conta Google existente.
- Usuario: Classe que define o modelo de usuário. Esta classe é utilizada para o cadastro e login de usuários.
- PrivacyPolicy: Uma classe que gera uma página simples com a política de privacidade.

Figura 2 - Diagrama de Classes



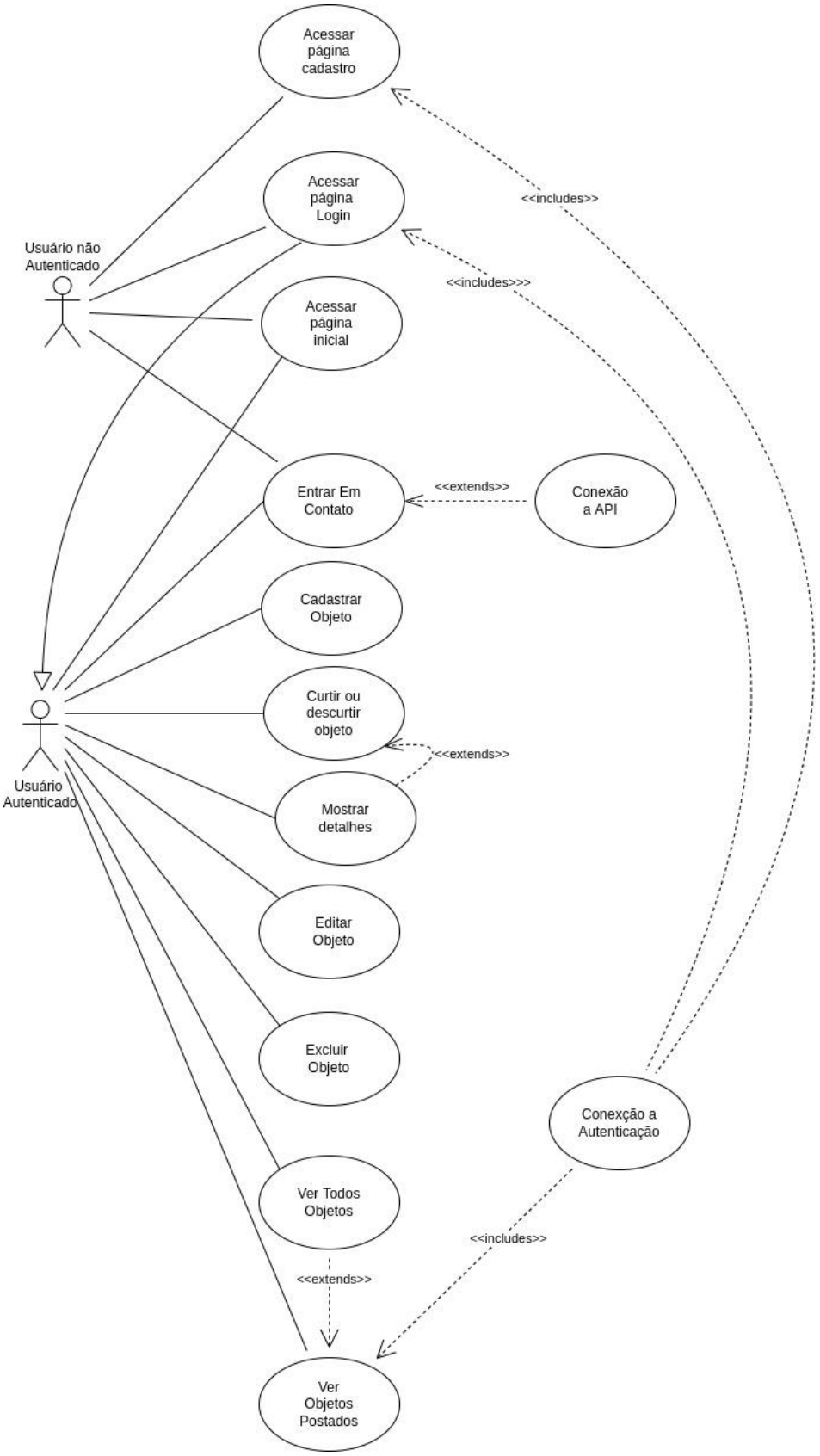
Fonte: Autoria própria

2.3 DIAGRAMA DE CASOS DE USO

A **figura 3** mostra o diagrama de casos de uso do sistema, levando em consideração os usuários autenticados e não autenticados. O Diagrama mostra quais funcionalidades do sistema são disponibilizadas para cada tipo de usuário, sendo mostrado o ponto de vista do usuário autenticado e do usuário não autenticado.

O usuário não autenticado possui quatro opções: acessar a página inicial, entrar em contato com desenvolvedores, acessar página de cadastro de conta e acessar página de login. O usuário autenticado possui as seguintes opções: acessar a página inicial, acessar página de contato com desenvolvedores, cadastrar objetos, ver lista de objetos, ver detalhes de objetos, excluir objetos, curtir ou descurtir objetos, editar objetos e ver objetos postados por si mesmo. O usuário autenticado não pode acessar a página de cadastro e login.

Figura 3 - Diagrama de de Casos de Uso



Fonte: Autoria própria

2.4 DESCRIÇÃO DOS PRINCIPAIS REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os principais requisitos funcionais do sistema são:

- O sistema deve prover o cadastro de objetos de robótica educacional.
- O sistema deve permitir a atualização de objetos.
- O sistema deve exibir objetos armazenados no banco de dados.
- O sistema deve permitir a busca textual por objetos.
- O sistema deve prover um método de exclusão de objetos.
- O sistema deve exibir os detalhes de cada objeto.
- O sistema deve prover um meio de os usuários avaliarem positivamente ou negativamente um objeto.
- O sistema deve prover um método de envio de e-mails para os desenvolvedores.
- O sistema deve dispor de um meio de autenticação com e-mails.
- O sistema deve permitir a autenticação com Conta Google.
- O sistema deve permitir o encerramento de uma sessão autenticada.
- O sistema deve conter um método de criação de contas com e-mail.
- O sistema deve prover uma filtragem de objetos por data de postagem ou número de acessos.

Os principais requisitos não funcionais foram definidos utilizando como base o padrão ISO/IEC 25010 demonstrado no trabalho de Vazquez e Simões (2016), os quais são são:

- Confiabilidade: O software deve apresentar consistência e estabilidade, estando sempre disponível para os usuários.
- Eficiência no desempenho: O software deve tratar os dados com a maior eficiência e velocidade o possível.
- Usabilidade: O software deve ser intuitivo, isto é, fácil de utilizar, não deixando brechas para mal uso ou dificuldade de uso.
- Compatibilidade: O software deve interagir com outros sistemas, se necessário

- Segurança: O software deve prover uma proteção sólida aos dados dos usuários, mantendo os mesmos seguros mesmo em caso de ataques.
- Capacidade de manutenção: O sistema deve ser facilmente administrado e mantido.

3. IMPLEMENTAÇÃO

O software possui um total de oito diferentes páginas que são exibidas ao usuário representando o *frontend* da plataforma, sendo que três dessas páginas são exibidas apenas para usuários autenticados. Visto que o software foi desenvolvido utilizando o *framework frontend* Angular, que segue o modelo *single page application*, em que todas as páginas são carregadas dentro de um único arquivo, sob a demanda em que o usuário requer acessar as páginas (ANGULAR, 2022).

Cada página é formada por um conjunto de arquivos, esse conjunto de arquivos é chamado de componente. A aplicação possui nove componentes.

- HomeComponent: Este componente gera a página principal da aplicação, sendo acessível por todos os usuários.
- ListaObjetosComponent: Este componente gera a página de lista de objetos, sendo acessível somente pelos usuários autenticados.
- DetalhesObjetoComponent: Este componente gera uma página com todos os campos de um objeto específico que foi postado, sendo acessado somente por usuários autenticados.
- PrivacyPolicyComponent: Este componente gera a página de políticas de privacidade da plataforma, sendo exibida para qualquer usuário.
- FormularioCadastroComponent: Este componente gera a página de formulário de cadastro de objetos, sendo exibido somente para usuários autenticados.
- ContatoComponent: Esta página gera o formulário de contato com os desenvolvedores, sendo exibida para qualquer usuário.
- UsuárioLoginComponent: Este componente gera a página em que o usuário usa para iniciar sua sessão autenticada no sistema. Caso o usuário esteja autenticado, será renderizado o componente ListaObjetosUsuarioComponent.

- ListaObjetosUsuarioComponent: Este componente é exibido dentro do componente de login, caso o usuário esteja autenticado.
- UsuarioCadastroComponent: Este componente gera a página que o usuário utiliza para criar uma conta utilizando um e-mail de sua escolha.

O padrão de metadados Dublin Core revisado foi utilizado para descrever os campos dos objetos de aprendizagem no sistema, porém havendo pequenas mudanças na forma em que são armazenados alguns metadados. As mudanças ocorreram nos seguintes campos:

- Autoridade (dc.creator.authority): este campo de metadado é preenchido automaticamente, sendo que seu valor sempre será o RepositORE.
- Buscador de informação (dc.creator.search_information): este campo de metadados é preenchido automaticamente, sendo que seu valor sempre será a URL do RepositORE.
- Editor (dc.publisher.editor): este campo de metadados é preenchido automaticamente, sendo que seu valor sempre será o usuário que fez a postagem do objeto de aprendizagem.
- Contribuinte (dc.contributor): este campo de metadados é preenchido automaticamente, sendo que seu valor sempre será uma lista que referencia os usuários que editaram o objeto de aprendizagem.
- Data de disponibilização (dc.date.availability): este campo de metadados é preenchido automaticamente, sendo que seu valor sempre será a data em que objeto de aprendizagem é postado.

O padrão Dublin Core revisado está aplicado de duas formas: embutido no HTML do sistema e o armazenamento por meio de coleções no Firebase também utiliza um grande objeto com campos Dublin Core.

O RepositORE teve seu *backend* estruturado através da utilização do pacote AngularFire, que ajuda na integração de funcionalidades do Firebase com aplicações construídas em Angular, e também através do uso de Node.js e Express.js na criação de uma *API REST* para conexão com o serviço de envio de e-mails disponibilizado pelo *Sendgrid*. Foi utilizado o banco de dados NoSQL em nuvem Cloud Firestore, do Firebase. No Cloud Firestore, os dados são estruturados e guardados em documentos, sendo que esses documentos são armazenados

dentro de uma coleção. O *backend* da plataforma é formado primariamente por quatro arquivos, sendo eles: database-maestro.service.ts, storage.service.ts auth-maestro.service.ts e index.js.

- DatabaseMaestroService: Este arquivo é o responsável por fazer a conexão entre componentes e banco de dados. Através deste arquivo, é possível: inserir novos documentos em uma coleção já existente por meio do método “insert”; receber todos os documentos de uma coleção utilizando o método “getAll”; deletar um documento da coleção utilizando o método “delete”; atualizar um documento de uma coleção através do método “update”; incrementar a quantidade de acessos no campo “access” dentro de um documento específico na coleção, por meio do método “updateAccess”; incrementar a quantidade de curtidas no campo “like” dentro de um documento específico na coleção, por meio do método “updateLike”; incrementar a quantidade de descurtidas no campo “deslike” dentro de um documento específico na coleção, por meio do método “updateDeslike”; fazer a conexão com a *Api* index.js, para o envio de e-mails de contato através do método “uploadContact”.
- StorageService: É o arquivo responsável por fazer o envio de arquivos de imagem/vídeo para o banco de dados no Firebase. O método “uploadFile” faz o envio do arquivo, criando uma identificação aleatória para cada imagem.
- AuthMaestroService: Este arquivo é responsável pela autenticação de contas de usuário e controle de sessões de usuário. Através deste arquivo, é possível: autenticar o usuário com uma Conta Google existente ao utilizar o método “loginGoogle”; retornar valor do usuário atual, caso ele não seja nulo por meio do método “getUser”; criar uma conta com e-mail e senha com o método “createUserByEmail”; autenticar o usuário com uma conta de e-mail e senha existente, via utilização do método “loginEmail”; encerrar a sessão do usuário com o método “logout”
- Index.js: Este arquivo é responsável por fazer o envio de e-mails de contato com os desenvolvedores, utilizando do serviço de envio de e-mails proveniente do Sendgrid. Index.js utiliza uma chave de conexão com o Sendgrid que está encapsulada em uma variável de ambiente. Através do método “get”, os dados escritos no formulário de contato são recebidos. O

método “post” envia os dados recebidos para o serviço do Sendgrid, que por sua vez envia uma mensagem em forma de e-mail.

3.1 METODOLOGIA E DESENVOLVIMENTO

Para o desenvolvimento do código, foi utilizada a metodologia ágil Scrum. Esta metodologia foi escolhida devido aos seus benefícios e fluxo de trabalho se adequarem bem ao estilo de desenvolvimento e relacionamento da equipe. Os principais benefícios que se mostraram de suma importância durante o desenvolvimento foram: agilidade, adaptabilidade e colaboração. O projeto foi dividido em *sprints*, que é dividir o processo de construção do software em pedaços, criando uma melhor visualização da evolução do projeto, e tornando o processo de desenvolvimento mais claro (SCRUM, 2022).

As funcionalidades a serem implementadas, divisão de tarefas e erros a serem resolvidos foram definidos através de *issues* no Github. Uma *issue* é uma marcação de funcionalidade a ser implementada, ou seja, o desenvolvedor utiliza *issues* para monitorar uma funcionalidade que ele precisa implementar. A equipe era formada por dois desenvolvedores, e, apesar de ambos terem desenvolvido em conjunto algumas páginas da plataforma, este trabalho teve um foco maior no desenvolvimento do *backend* da aplicação.

Foram realizadas reuniões semanais, geralmente contendo no mínimo 40 minutos de duração. Durante as reuniões, eram discutidas as funcionalidades implementadas, a forma na qual foram implementadas, melhorias que poderiam ser feitas nessas funcionalidades, quais funcionalidades seriam desenvolvidas e estipulação de prazos.

3.2 FERRAMENTAS E BIBLIOTECAS UTILIZADAS

Foram utilizadas diversas ferramentas e tecnologias, sendo as principais: Google Firebase, Angular, Bootstrap, Github, Visual Studio Code, NodeJS, SendGrid, AngularFire e Heroku.

3.2.1 GOOGLE FIREBASE

O Firebase é uma plataforma gerenciada pela Google para a criação e desenvolvimento de aplicativos web e mobile, oferecendo uma série de serviços *backend as a service*, os quais são uma série de ferramentas e recursos que oferecem uma maior facilidade de integração com banco de dados, análise e gerenciamento dos dados(FIREBASE, 2022). Dentre os serviços oferecidos pelo Google Firebase estão:

- Cloud Firestore: Banco de dados NoSQL orientado a documentos, para o armazenamento de dados em nuvem. No Cloud Firestore, os dados são estruturados e guardados em documentos, sendo que esses documentos são armazenados dentro de coleções.
- Autenticação: Gerenciamento de usuários, senhas e sessões.
- *Hosting*: Armazenamento de sites para produção.
- Cloud Storage: Armazenamento de fotos e vídeos em nuvem.
- Crashlytics: Monitoramento da estabilidade de aplicações.
- Google Analytics: Geração de relatórios de eventos específicos.
- Monitoramento de desempenho: Resolução e monitoramento do desempenho de aplicações.

3.2.2 ANGULAR

O Angular é um *framework* front-end baseado em TypeScript para o desenvolvimento de aplicativos web mantido pelo Google e Microsoft, que facilita o processo de desenvolvimento e manutenção de aplicações web. (ANGULAR, 2022): “Angular é uma estrutura de design de aplicativos e plataforma de desenvolvimento para criar aplicativos de página única eficientes e sofisticados.”

3.2.3 BOOTSTRAP

O Bootstrap é um *framework* de desenvolvimento front-end de código aberto para desenvolvimento de interfaces em aplicativos web, permitindo aos desenvolvedores a criação de interfaces responsivas e personalizadas mais facilmente (BOOTSTRAP, 2022).

3.2.4 GITHUB

O Github é uma plataforma de hospedagem de código e versionamento de código, permitindo aos seus usuários terem total controle de seus projetos, e manterem um desenvolvimento colaborativo e seguro (GITHUB, 2022).

3.2.5 VISUAL STUDIO CODE

O Visual Studio Code é um software para edição de código, possuindo suporte ao versionamento de código Git. (VISUAL STUDIO CODE, 2022): “O Visual Studio Code é um editor de código-fonte leve, mas poderoso, que é executado em sua área de trabalho e está disponível para Windows, macOS e Linux. Ele possui JavaScript, TypeScript e Node, com suporte de C, integrado e outras linguagens (como vem C++) e outras linguagens (como vem com C++, Java, Python, PHP, Go e outras linguagens de execução).”

3.2.6 NODEJS

O NodeJS é um software de código aberto aberto que permite a execução de código Javascript fora de navegadores. É utilizado principalmente para o desenvolvimento e criação de código *backend.* O NodeJS foi desenvolvido para a construção de aplicações de rede com escalabilidade (NODEJS, 2022).

3.2.7 EXPRESSJS

O Express é um dos frameworks para NodeJs mais utilizados. (EXPRESSJS, 2022): “O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.”

3.2.8 SENDGRID

O SendGrid é uma plataforma que proporciona uma *api* para comunicação com usuários através de e-mails transacionais. O SendGrid utiliza uma arquitetura baseada em nuvem, permitindo a análise de dados, gerenciamento e entrega segura de e-mail (SENDGRID, 2022).

3.2.9 HEROKU

O Heroku é uma plataforma que oferece serviço de hospedagem e publicação de projetos virtuais na nuvem. (HEROKU, 2022): “Heroku é uma plataforma em nuvem que permite que as empresas criem, entreguem, monitorem e dimensionem aplicativos — somos a maneira mais rápida de passar da ideia à URL, ignorando todas as dores de cabeça de infraestrutura.”

3.2.10 ANGULAR FIRE

O Angular Fire é a biblioteca oficial de integração do *framework* Angular com as bibliotecas do Firebase (ANGULARFIRE, 2022).

3.3 CAPTURAS DE TELA

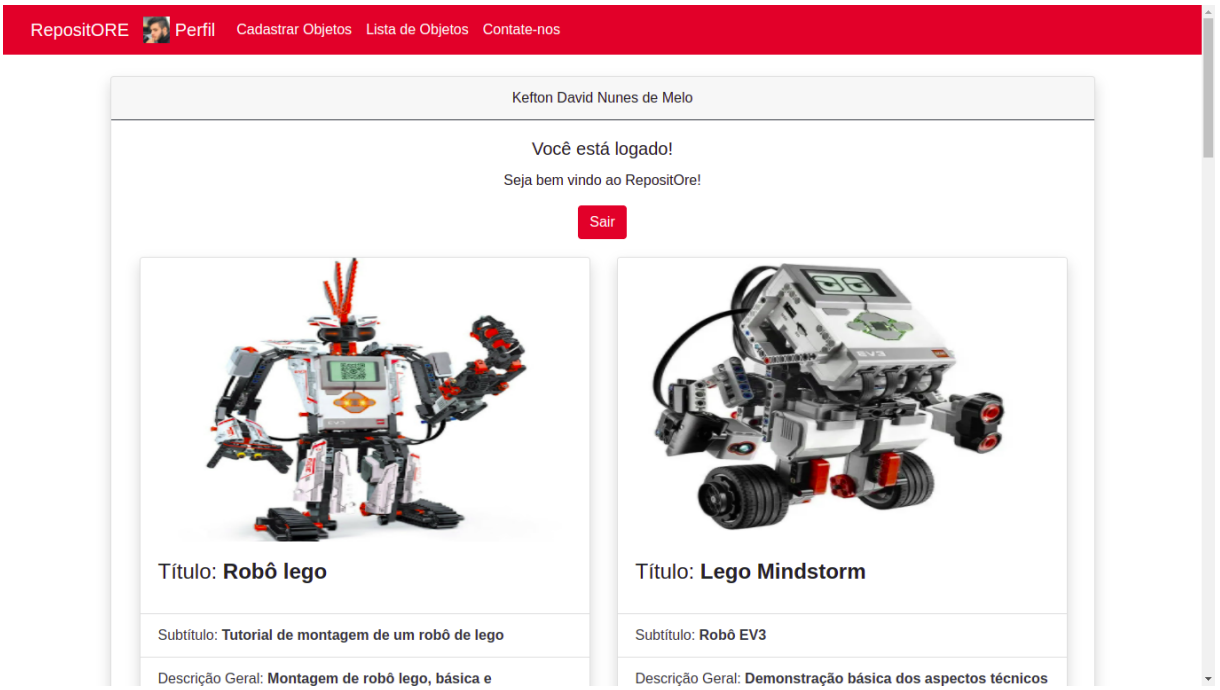
Na **figura 4** é vista a tela de página inicial da plataforma, onde o usuário pode visualizar as cinco últimas postagens feitas por usuários. Esta página pode ser vista por usuários autenticados e não autenticados.

Figura 4 - Página inicial

Fonte: Autoria Própria

Na **figura 5** é possível visualizar os objetos postados e editados pelo usuário.

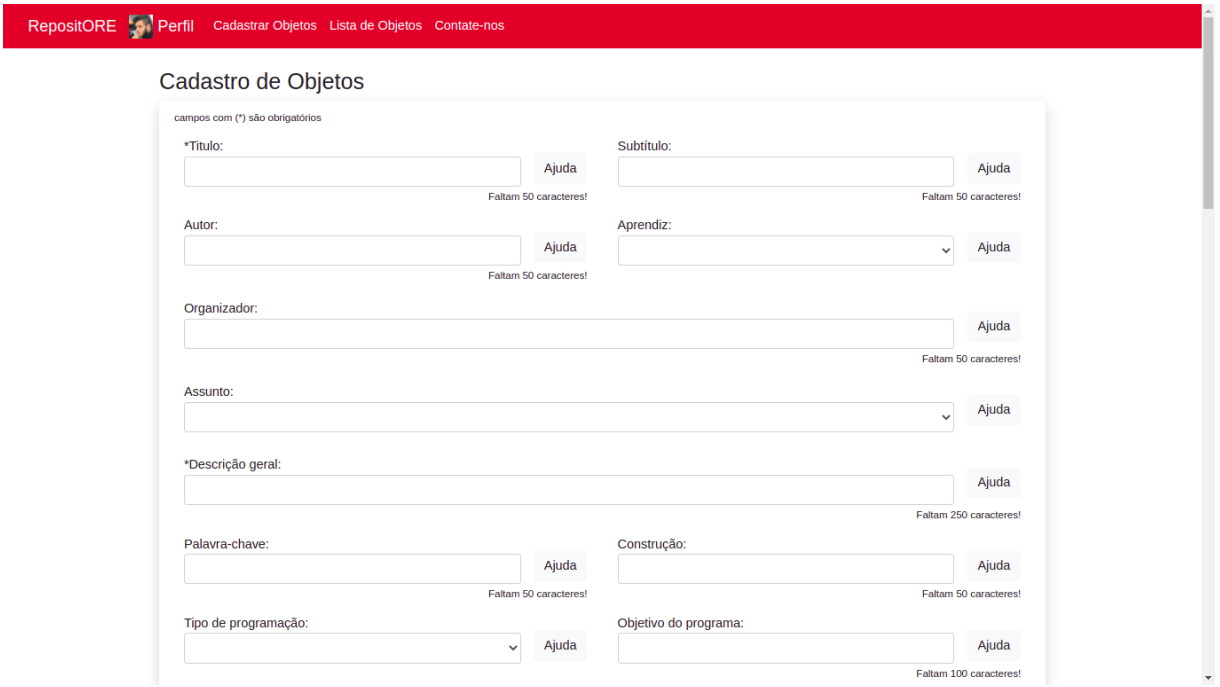
Figura 5 - Postagens do usuário



Fonte: Autoria Própria

O formulário de cadastro de objetos de aprendizagem pode ser visualizado através da **figura 6**. No cadastro, existem um conjunto de campos a serem preenchidos, sendo que apenas cinco campos são obrigatórios. Cada campo possui um botão de ajuda, que mostra um resumo sobre o campo específico.

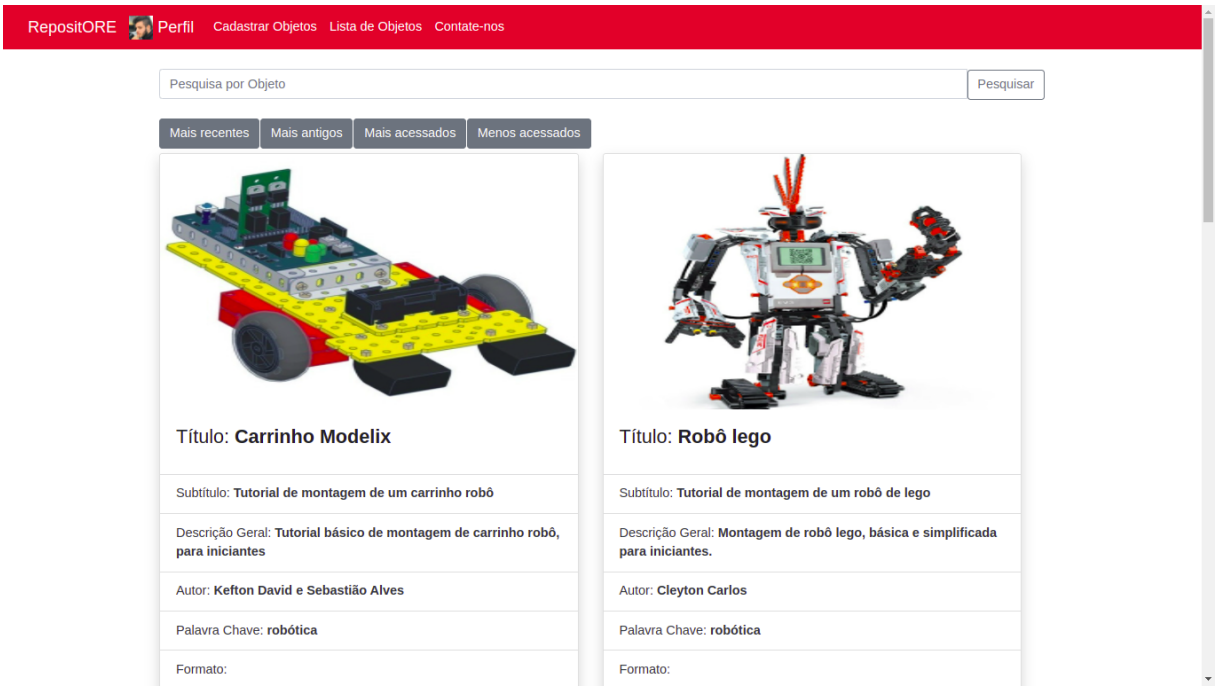
Figura 6 - Formulário de cadastro de objetos



Fonte: Autoria própria

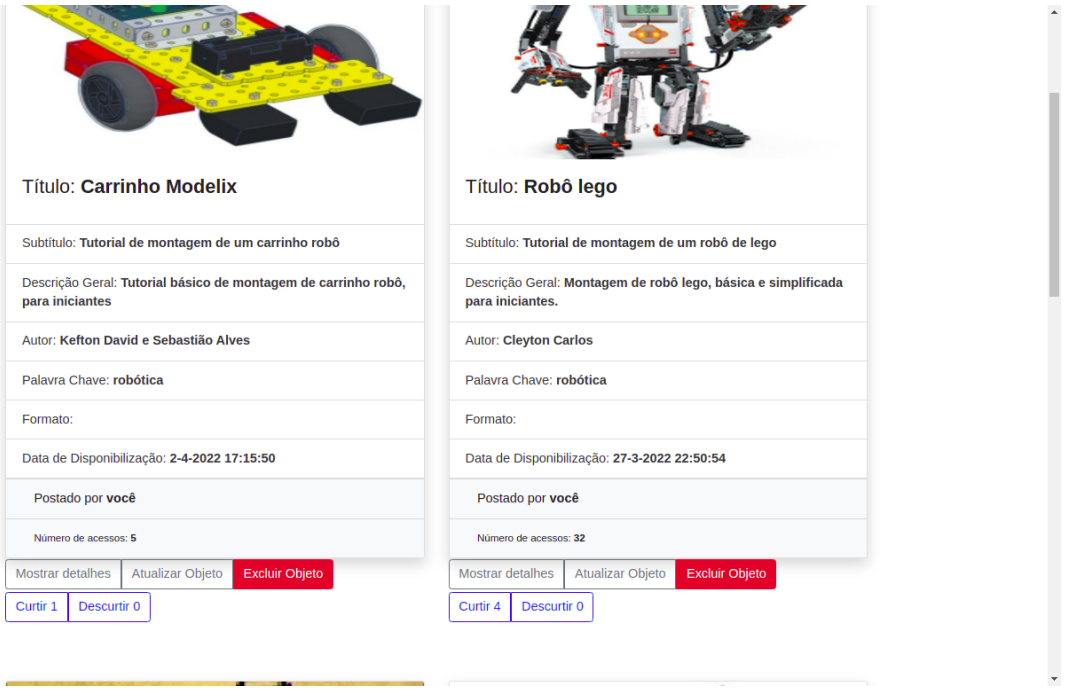
A **figura 7** mostra a tela de listagem de objetos de robótica educacional, postados pelos usuários autenticados. Na lista de objetos, é possível ver algumas funcionalidades, tais como: a barra de pesquisa textual por objetos; filtragem por postagens mais recentes, mais antigas, mais acessadas e menos acessadas; botões de curtir e descurtir um objeto de aprendizagem; atualizar, deletar e ver detalhes dos objetos.

Figura 7 - Lista de objetos



Fonte: Autoria própria

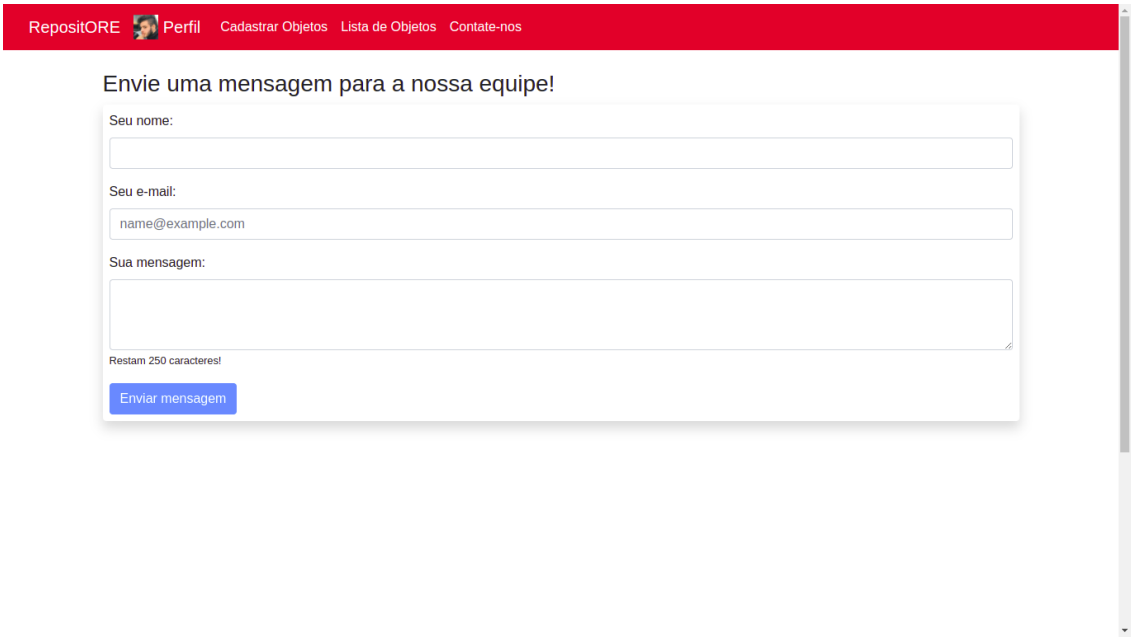
Figura 8 - Funcionalidades na lista de objetos



Fonte: Autoria própria

É possível que o usuário entre em contato com com os desenvolvedores através do formulário de contato. No click do botão do formulário, é enviado um e-mail para os desenvolvedores. A **figura 9** mostra o formulário de contato.

Figura 9 - Formulário de contato

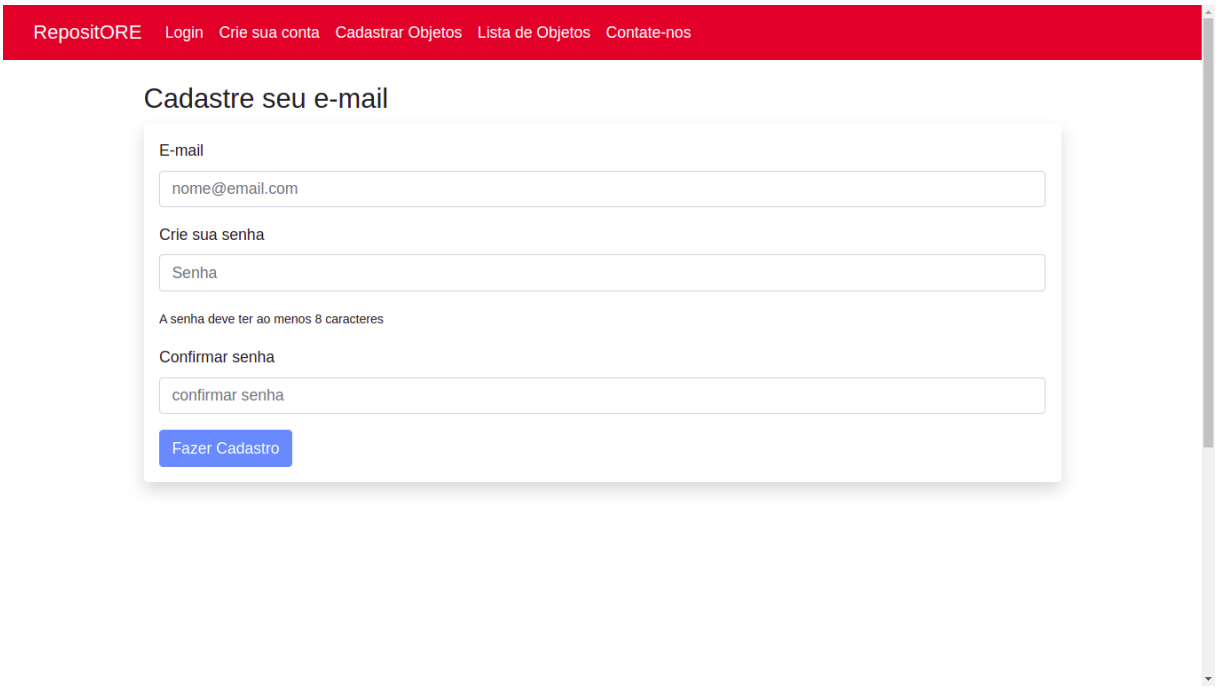


Fonte: Autoria própria

A **figura 10** possibilita a observação da tela de criação de conta, caso o usuário não esteja logado no sistema. É possível criar uma conta utilizando um e-mail preexistente.

A tela de criação de conta só pode ser acessada e visualizada enquanto o usuário não estiver logado no sistema.

Figura 10 - Criação de conta

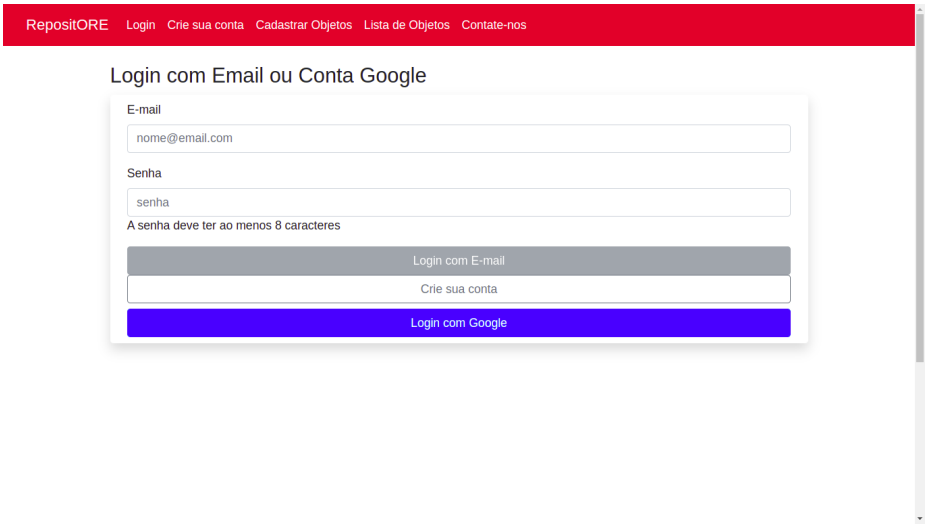


Fonte: Autoria própria

Caso o usuário não esteja logado no sistema, ele pode fazer login através da página de login, que pode ser observada na **figura 11.** O usuário pode preencher o formulário e fazer o login através da sua conta de e-mail previamente cadastrada, ou fazer o login direto através de uma Conta Google existente.

A página de login não é acessível após o usuário logar no sistema.

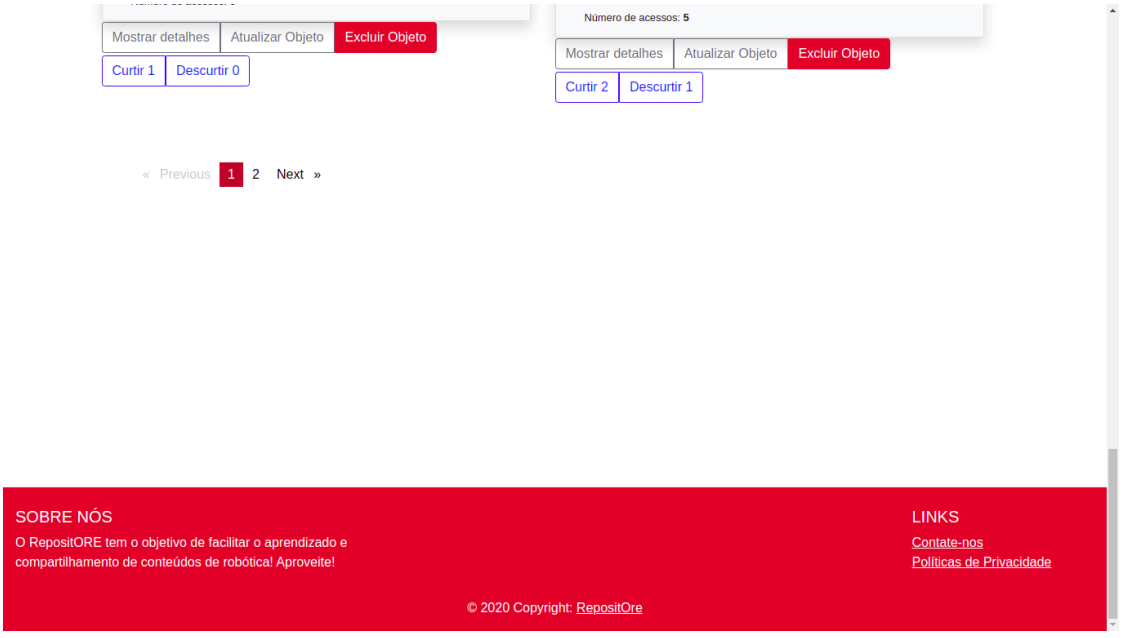
Figura 11 - Login



Fonte: Autoria própria

O rodapé da plataforma pode ser visto pela **figura 12.** São disponibilizadas três opções de link, sendo elas: página principal, formulário de contato e políticas de privacidade.

Figura 12 - Rodapé da página



Fonte: Autoria própria.

As políticas de privacidade da plataforma podem ser vistas na **figura 13.**

Figura 13 - Políticas de privacidade

Política De Privacidade

Sobre a privacidade dos dados

Todos os dados providos para a criação dos *usuários* são armazenados no Google Firebase, e só serão utilizados para identificar os usuários na plataforma. Todos os dados providos no cadastro de um objeto poderão ser disponibilizado de maneira pública.

Responsabilidades e Papeis

O *usuário* tem como responsabilidade de verificar tudo que o mesmo está injetando na plataforma por meio de *Cadastro de Objeto*, já que a veracidade do conteudo postado será de responsabilidade do *usuário*. O **RepositORE** tem como responsabilidade manter a integridade dos dados, além disso, o *administrador* da plataforma também está no direito de excluir conteudo da plataforma em casos de fraude, conteudo impróprio ou não relevante para a plataforma. Em casos de contíinua publicação de conteúdo improprio ou ilicito a conta do *usuário* poderá ser terminada permanentemente.

Sobre seus dados pessoais

Os dados pessoais disponibilizados no momento do cadastro serão amntidos em sigilo e serão utilizados somente para identificação do usuário na plataforma. Essas informações trafegam pelo servidor do Google Firebase apenas para armazenamento e identificação. aqui está a Termos de Uso e a Política de Privacidade da prestadora de serviços.

Sobre os dados públicos

Todos os dados disponibilizados para o *cadastro de objetos* no **RepositORE** serão tratados como publicos e podem ou não estar completamente disponiveis para qualquer usuário da plataforma ou utilizadores não cadastrados.

Armazenamento dos Dados

Todo o Armazenamento de dados e lidado pelo Google Firebase, e portanto também estão sobre efeito dos termos de uso da prestadora de serviços. Os dados, todavia, não estão sobre efeito da política de privacidade da mesma, ou seja, estão somente sujeitos aos termos desse contrato.

Direitos do Titular

Todos os usuários tem direito de manipular dados da própria plataforma nas formas de Adição, Subtração e Edição dos dados. Contanto, o **RepositORE** é considerado um repositório de contribuição, então alterações devem ser levadas com moderação. Em caso de comportamento malicioso proposital, o Administrador tem o direito de entrar em ação para proteger os dados e a integridade da plataforma. podendo limitar os

Fonte: Autoria própria

4. TESTES E VALIDAÇÃO

Foi realizado uma série de testes e etapas de teste de software, para garantir a usabilidade, e que a aplicação possua somente os comportamentos desejados.

Testes realizados:

- Testes unitários: Foram feitos diversos testes unitários durante o desenvolvimento do software, em todos os arquivos que foram utilizados para criação de código. Na data da criação desse documento, todos os componentes foram amplamente testados separadamente.
- Testes de integração: Todos os componentes da aplicação foram testados, e as interações entre os componentes que necessitam de interoperabilidade entre si se mostraram satisfatórias. Entretanto, como é utilizado o formato de cobrança gratuito do google firebase, a plataforma tem um limite operacional associado com os limites impostos pelo Google Firebase, isso inclui o limite de tráfego simultâneo, a capacidade máxima de armazenamento total do banco de dados, entre outros limites.
- Teste operacional: A funcionalidade da aplicação foi extensamente colocada a prova em seu estado hospedado em um provedor de

serviços web, como a aplicação foi desenhada para ser utilizada, e sua funcionalidade de todos os casos de uso foi avaliada como como funcional. Todos os módulos não apresentaram nenhum tipo de mal comportamento e as respostas da aplicação estão como foram planejadas. Não foi possível realizar um teste de estresse da plataforma, pois para isso é necessário de mais de 20 pessoas utilizando o programa ao mesmo tempo.

5. CONCLUSÕES E PERSPECTIVAS FUTURAS

O sistema RepositORE está atualmente implementado, apresentando um funcionamento desejável do backend, tanto a integração ao banco de dados e armazenamento de imagens, quanto a *API* de envio de e-mails de contato com desenvolvedores. O RepositORE está publicado e hospedado no Firebase, e pode ser acessado através do link: https://repositore-601ab.web.app/ .

A plataforma em sua condição atual, servirá como a base para a implementação de funcionalidades novas e melhorias nas funcionalidades atuais e de qualidade de vida e usabilidade do software. Algumas melhorias a serem implementadas são: melhorias no sistema de recomendação de conteúdos; criar uma espécie de moderação de conteúdos, a fim de manter um controle sobre o contexto das postagens; implementação de página de administrador; adicionar um CAPTCHA para aumentar a segurança do site; aplicar melhorias no perfil do usuário e entre outros.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ANGULAR. **Angular.** 2022. Disponível em: https://angular.io. Acesso em: 8 abr 2022.

ANGULARFIRE. **AngularFire**. 2022. Disponível em: https://www.npmjs.com/package/@angular/fire. Acesso em 10 abr 2022.

BOOTSTRAP. **Bootstrap.** 2022. Disponível em: https://getbootstrap.com. Acesso em: 9 abr 2022.

DANIELA, Pires. **USO DO DUBLIN CORE NA DESCRIÇÃO DE OBRAS RARAS NA WEB: A COLEÇÃO DA BIBLIOTECA BRASILIANA DIGITAL** , 2012.

EXPRESSJS. **Express.js.** 2022. Disponível em: https://expressjs.com/pt-br/. Acesso em 9 abr 2022.

FIREBASE. **Firebase.** 2022. Disponível em: https://firebase.google.com/docs. Acesso em: 9 abr 2022.

GITHUB. Disponível em: https://github.com/features. Acesso em: 9 abr 2022.

GONÇALVES, Paulo de Castro. et al. **Adequação do Dublin Core ao AACR2: o caso da Biblioteca Digital da Assembleia Legislativa do Estado de Minas Gerais.** XXV.

GURGEL, Thalia K. S. **RepositORE: Um Repositório de Objetos de Aprendizagem para Robótica Educacional,** Trabalho de Conclusão de Curso de Ciências da Computação, Faculdade de Ciências Naturais e Exatas, Universidade Estadual do Rio Grande do Norte, Mossoró, 2019.

HEROKU. **Heroku.** 2022. Disponível em: https://www.heroku.com/what. Acesso em: 9 abr 2022.

NODEJS. **Node.js.** 2022. Disponível em: https://nodejs.org/en/about/. Acesso em: 9 abr 2022.

ROSETTO, Marcia; NOGUEIRA, Adriana Hipólito. **Aplicação de elementos metadados DUBLIN CORE para descrição de dados bibliográficos on-line da biblioteca digital de teses da USP**. Anais.. Recife: SNBU, 2002.

SCRUM. **Scrum.** 2022. Disponível em: https://www.scrum.org/resources/what-is-scrum. Acesso em: 8 abr 2022.

SENDGRID. **Sendgrid.** 2022. Disponível em: https://sendgrid.com/solutions/email-api/. Acesso em: 9 abr 2022.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. **Engenharia de Requisitos: software orientado ao negócio.** 2016.

VSCODE. **Visual Studio Code.** 2022. Disponível em: https://code.visualstudio.com/docs. Acesso em: 9 abr 2022.