MÉRCIO ANTÔNIO OLIVEIRA DE ANDRADE FILHO

PROJETO E DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA A PROMOÇÃO DO REUSO DE LIVROS DIDÁTICOS UNIVERSITÁRIOS

Recife
2019

MÉRCIO ANTÔNIO OLIVEIRA DE ANDRADE FILHO

PROJETO E DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA A PROMOÇÃO DO REUSO DE LIVROS DIDÁTICOS UNIVERSITÁRIOS

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Bacharelado em Ciências da Computação

Orientador: Prof. Vanilson Burégio

Recife,

2019

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

Ficha de Aprovação do Trabalho de Conclusão de Curso

Trabalho defendido por **MÉRCIO ANTONIO OLIVEIRA DE ANDRADE FILHO** às 15:00 do dia 12 de dezembro de 2019, no Auditório do Departamento de Computação - DC – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado "**Projeto e Desenvolvimento de uma Aplicação Web para a Promoção do Reuso de Livros Didáticos Universitários** ", orientado por Vanilson Andre de Arruda Buregio e aprovado pela seguinte banca examinadora:

Vanilson André de Arruda Buregio
DC/UFRPE

Leandro Marques do Nascimento
DC/UFRPE

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

A553p de Andrade Filho, Mércio Antônio Oliveira
PROJETO E DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA A PROMOÇÃO DO REUSO DE LIVROS DIDÁTICOS UNIVERSITÁRIOS / Mércio Antônio Oliveira de Andrade Filho. - 2019.
90 f. : il.

Orientador: Vanilson Buregio.
Inclui referências e apêndice(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2019.

1. SPA. 2. RESTful API. 3. Grails. 4. Angular. 5. Reuso de Livros. I. Buregio, Vanilson, orient. II. Título

CDD 004

Resumo

É comum na vida do estudante universitário o acúmulo de livros didáticos. Muitos desses livros acabam subutilizados e parados em estantes. Nesse contexto, as principais ferramentas utilizadas pelos discentes para o compartilhamento online de livros didáticos apresentam limitações importantes. Essas limitações foram identificadas em um questionário aplicado a 75 estudantes universitários. Com o intuito de fomentar a prática do compartilhamento de livros didáticos entre discentes da UFRPE foi desenvolvido um sistema web específico para esse fim. Este trabalho descreve o projeto e a implementação dessa aplicação. Ela se trata de uma aplicação de página única que é alimentada através de uma a API REST. Desse modo, a API desenvolvida poderá futuramente ser consumida por outros softwares. Foram utilizados os frameworks Angular e Grails, além do SGBD PostgreSQL para a persistência de dados. Para avaliar a aplicação foi realizado um segundo questionário no qual 33 pessoas foram entrevistadas. Os participantes usaram recursos da aplicação e indicaram o seu grau de satisfação. A aplicação obteve uma avaliação positiva. O sistema conta, 30 dias após a sua implantação em nuvem, com 140 usuários cadastrados e 39 livros no seu acervo.

Palavras chave: SPA, RESTful API, Angular, Grails, Groovy, MVC, Reuso de Livros.

Abstract

It is common in the college student's life the accumulation of textbooks. Many of these books end up underutilized and standing on bookshelves. In this context, the main tools used by students to share textbooks online are limited. These limitations were identified in a form applied to 75 college students. In order to foster the practice of sharing textbooks among UFRPE students, a specific system was developed for this purpose. This paper describes the design and implementation of this application. It deals with a single page application that is fed through a REST API. Thus, the developed API may be consumed by other software in the future. We used the Angular and Grails frameworks, as well as the PostgreSQL DBMS for data persistence. To evaluate the application was conducted a second form. It was answered by 33 people. The participants used application's features and indicated their degree of satisfaction. The application got a positive rating. The system has, 30 days after its deployment in the cloud, 140 registered users and 39 books in its collection.

Keywords: SPA, RESTful API, Angular, Grails, Groovy, MVC, Book's Reuse.

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E SÌMBOLOS

GORM - Grails Object Relational Mapping

HTTP - Hyper Text Transfer Protocol

JSON - JavaScript Object Notation

JVM - Java Virtual Machine

REST - Representational State Transfer

CRUD - Create Read Update Delete

API - Application Programming Interface

SPA - Single Page Application

UFRPE - Universidade Federal Rural de Pernambuco

MVC - Model View Controller

JAVAEE - Java Enterprise Edition

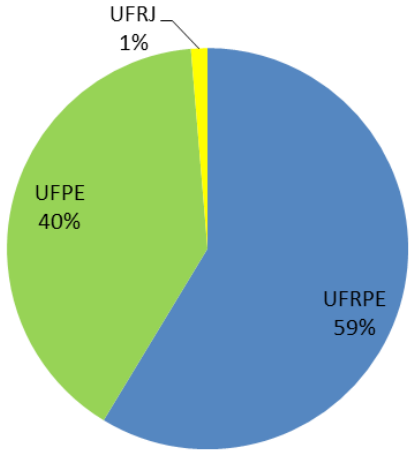
ÍNDICE

1. Introdução

O livro impresso é uma das principais razões para a confecção de papel e está intimamente relacionado ao universo acadêmico. As grades curriculares de cursos de graduação são compostas por uma ampla gama de disciplinas obrigatórias e optativas. Assim, durante a jornada acadêmica, é usual a compra de livros didáticos, muitos dos quais são utilizados somente ao longo das disciplinas. Eles têm um custo elevado para o discente e, geralmente, as bibliotecas não contam em seus acervos com todas as indicações dos professores, e quando possuem, é em quantidade inferior ao número de alunos.

Nesse contexto, com o intuito de amparar a ideia deste trabalho, foi aplicado um questionário a estudantes universitários. O questionário teve sua divulgação realizada através de redes sociais. Foram obtidas 75 respostas. As instituições dos participantes estão apresentadas na figura 1 a seguir:

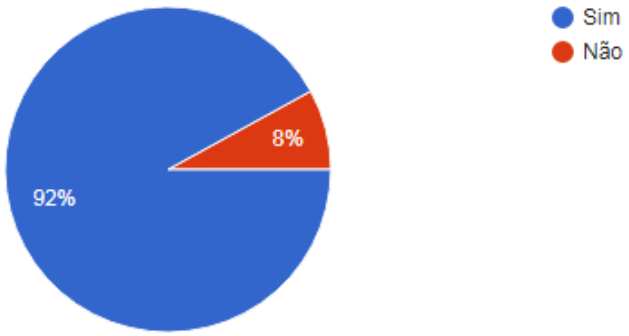
Figura 1: Instituição de ensino superior dos participantes do 1º questionário



Fonte: Elaborado pelo autor.

Seguidamente, foi perguntado se o participante possuía interesse no compartilhamento de livros didáticos, havendo 92% de respostas positivas. (Figura 2).

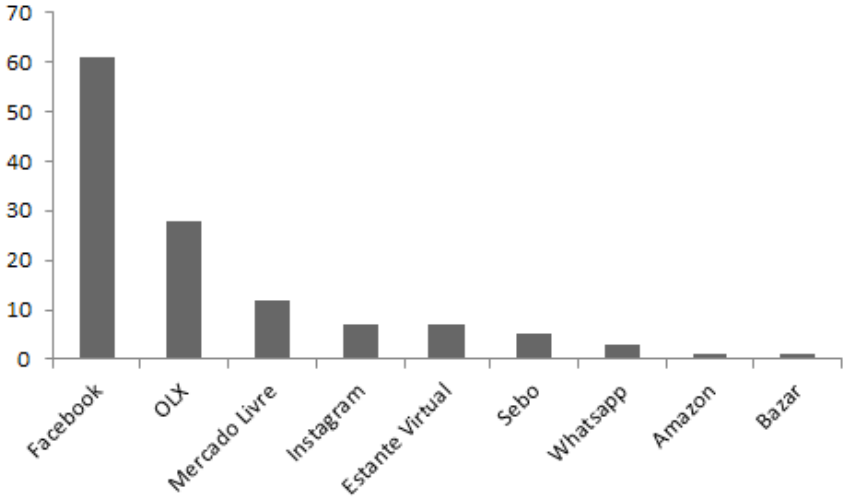
Figura 2: Porcentagem dos participantes do 1º questionário que possuem interesse no compartilhamento de livros didáticos.



Fonte: Elaborado pelo autor.

Objetivando conhecer as principais ferramentas utilizadas por estes discentes no contexto de compartilhamento de livros didáticos, o questionário fez a seguinte indagação aos participantes: “Qual ferramenta(s) você utiliza ou utilizaria para esse fim (compra, venda, troca ou doação de livros universitários)?” (Figura 3).

Figura 3: Ferramentas utilizadas para o compartilhamento de livros didáticos

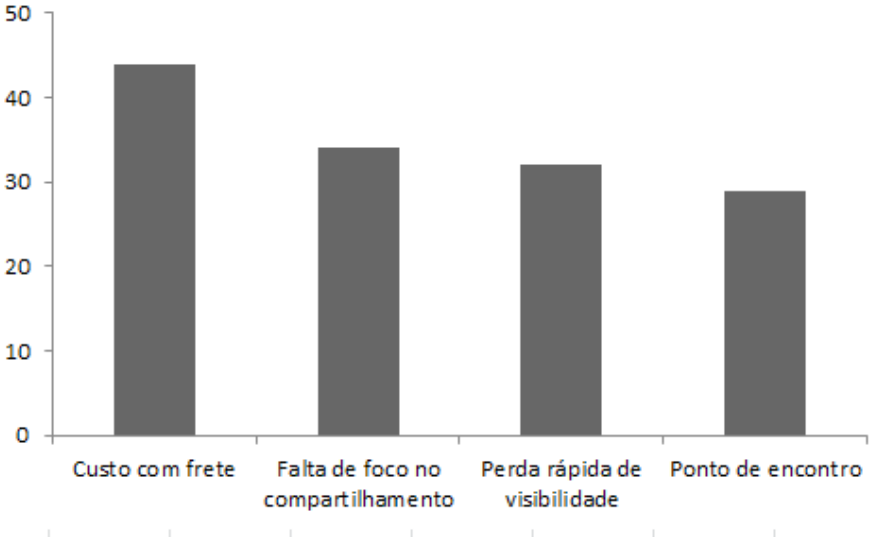


Fonte: Elaborado pelo autor.

De acordo com a figura 3, ferramentas muito populares como Facebook, OLX, Mercado Livre e Instagram foram as mais citadas pelos participantes. Estas ferramentas estão entre os sites mais acessados do Brasil (SIMILARWEB, 2019), e são utilizadas para o compartilhamento de livros entre discentes, no entanto, elas não foram feitas para esse propósito e acabam deixando lacunas importantes.

A próxima pergunta do questionário está relacionada as principais dificuldades ou limitações encontradas no uso das ferramentas já existentes para o compartilhamento de livros entre discentes universitários. (Figura 4).

Figura 4: Principais limitações encontradas nas ferramentas atuais



Fonte: Elaborado pelo autor.

De acordo com a figura 4, a principal dificuldade apontada pelos entrevistados foi o curso com frete. Essa dado está consoante com pesquisa realizada em 2017 pelo Serviço de Proteção ao Crédito (SPC Brasil) e a Confederação Nacional de Dirigentes Lojistas (CNDL) em escala nacional, na qual a principal desvantagem em compras online apontada pelos entrevistados foi o custo com frete.

A segunda limitação apontada, “Falta de foco no compartilhamento”, está relacionada a grande quantidade de funções providas por aplicações como Facebook ou Instagram e a grande variedade de tipos de produtos comercializados em sites como Mercado Livre ou OLX. Já a “perda rápida de visibilidade” é inerente a linha do tempo do Facebook. Essa linha do tempo exibe as postagens em ordem cronológica, dando destaque as postagens mais recentes. Em grupos como o da UFRPE, por exemplo, os anúncios perdem visibilidade rapidamente pois concorrem com outras inúmeras postagens de diversos temas.

A quarta limitação mais indicada pelos participantes foi a ausência de um ponto de encontro habitual, ou seja, um local de convívio comum, para a realização da troca em mãos do produto, como por exemplo, a mesma

universidade frequentada pelo anunciante e pelo interessado. Assim, não haveria custo com frete ou com deslocamento para um local alheio a rotina do interessado.

Diante disso, se faz propício o desenvolvimento de uma aplicação específica para o intermédio de informações entre estudantes universitários interessados na troca ou revenda de livros didáticos usados. Dessa maneira, poderá haver um acréscimo no reuso de livros entre os alunos, pois os mesmos poderão interagir diretamente entre si e marcar um ponto para troca dentro própria universidade, poupando assim gasto com frete ou deslocamento para outro ponto da cidade. Através de troca, ou compra por preços abaixo do mercado, eles poderão ter acesso a um material que de outro modo estaria subtilizado.

1.1 PROBLEMA DE PESQUISA

O acúmulo de material didático é uma constante na vida da maioria dos estudantes, e apesar do compartilhamento desse tipo de produto trazer benefícios pra todos, essa prática ainda está bastante aquém de sua plenitude. Nesse cenário, ferramentas online populares usadas para esse fim apresentam diversas limitações, dentre elas está ausência de um catálogo estruturado de acordo com os cursos e disciplinas de uma determinada universidade, por exemplo, ou a inexistência de um foco geográfico mais específico.

Diante disso, surge a questão: **A implementação de uma aplicação web específica para a comunicação entre discentes universitários interessados na compra, venda, troca ou doação de livros usados será bem recebida por esses estudantes?**

1.2 Objetivo Geral

O objetivo principal deste trabalho é o projeto e o desenvolvimento de uma aplicação para a web com o intuito de tornar mais eficiente o comércio informal de livros didáticos entre discentes universitários da UFRPE.

1.3 Objetivos Específicos

- Estudar o contexto do comércio informal de livros didáticos entre discentes universitários da UFRPE, através da aplicação de um questionário sobre o tema.

- Analisar as aplicações preexistentes de propósito similar ao deste projeto e identificar suas principais limitações.

- Projetar e desenvolver a aplicação proposta com base nas informações coletadas.

- Implantar em nuvem a aplicação desenvolvida.

- Avaliar a aplicação através de questionários a usuários.

1.4 Estrutura do documento

Na próxima seção, Revisão da Literatura, serão apresentados diversos trabalhos acadêmicos que empregaram tecnologias web modernas para o desenvolvimento de aplicações. Também serão descritas as principais aplicações presentes no mercado usadas por estudantes para o compartilhamento de livros didáticos.

Na seção 3, Fundamentação Teórica, serão apresentadas as tecnologias e ferramentas utilizadas para o desenvolvimento da aplicação proposta neste trabalho. Na seção 4, Projeto da Aplicação Proposta, serão expostos diagramas referentes ao projeto da aplicação. Na seção 5, Implementação da Aplicação Proposta, serão apresentados detalhes de implementação do software bem como as suas telas. Na seção 6, Avaliação da Aplicação Proposta, será apresentada a avaliação da aplicação que foi realizada através de entrevista com usuários.

2. Aplicações Relacionadas

Nesta seção 2.2, serão apresentadas aplicações presentes no mercado e utilizadas para o compartilhamento de livros entre estudantes. Elas foram indicadas por discentes entrevistados em questionário como as principais ferramentas utilizadas por eles para o compartilhamento de livros didáticos.

2.1 SOLUÇÕES DE MERCADO

2.1.1 FACEBOOK

O Facebook[1] é uma rede social extremamente popular. Ele Foi fundada em 2004 por Mark Zuckerberg. Inicialmente era restrito a universidade de Harvard, entretanto, se expandiu em escala global e se tornou uma das maiores redes sociais do mundo.

O Facebook teve em 2016 mais de 1.79 bilhões de usuários ativos (LEE, 2018; SHANE-SIMPSON et al., 2018). Em julho de 2019 ele possuía 2,375 bilhões de usuários ativos (STATISTA, 2019).

No mês de outubro de 2019, 5,63% de um total de 24,60 bilhões de visitas recebidas pelo Facebook foram provenientes do Brasil. Isso implica em aproximadamente 1 bilhão e 380 milhões de visitas realizadas por brasileiros em apenas 1 mês. (SIMILARWEB, 2019).

2.2.1.1 Grupo de compra e venda do Facebook

O Facebook possui uma funcionalidade denominada de “Grupo”. Ela consiste em um agrupamento de usuários em torno de um tema de interesse comum. Os grupos do Facebook podem ser tipificados como: “Geral”, “Compra e Venda”, “Jogos”, “Aprendizado Social”, “Empregos” e “Trabalho”.

[1]https://www.facebook.com

Serão apresentadas, em seguida, algumas características do tipo de grupo “Compra e Venda”. Essa configuração traz algumas funcionalidades inerentes ao contexto de compra e venda de produtos.

As principais funcionalidades de um grupo de compra e vendas são as seguintes (Figura 5):

Figura 5: Menu de um grupo de compra e venda do Facebook

Fonte: Facebook

- **Sobre**: Exibe uma descrição sobre o grupo. Mostra os seus administradores e membros.

- **Discussão:** O item discussão é inicialmente selecionado por padrão. Ele exibe as postagens ou anúncios dos membros em ordem cronológica.

- **Seus itens**: Lista os anúncios feitos pelo próprio usuário. Classificando-os em “À Venda”, “Vendido” ou “Arquivado”.

- **Membro**s: Lista os membros do grupo.

- **Vídeos**: Exibe todos os vídeos postados no grupo por membros do mesmo.

- **Fotos**: Exibe todas as fotos postadas no grupo por membros do mesmo.

- Eventos: Mostra eventos relacionados ao contexto do grupo.

2.2.1.2 Anúncio em um grupo de compra e venda do Facebook

Na figura 6 a seguir, é exposto como é estruturado um anúncio em um grupo de compra e venda do Facebook.

Figura 6: Anúncio em Grupo de Compra e Venda do Facebook

Fonte: Facebook

Eles são constituídos por um título, o preço pretendido para venda, a cidade do anunciante, uma foto ou fotos do produto e uma descrição textual feita livremente pelo autor do anúncio. O interessado no produto pode enviar uma mensagem diretamente ao anunciante ou fazer um comentário abaixo do anúncio.

Os grupos do Facebook são focados no relacionamento entre um conjunto de usuários, ou seja, um membro que realize uma postagem dentro de um grupo, exporá esse conteúdo para todos os membros desse grupo. Esse

recurso se tornou um meio popular para o comércio informal entre diversos grupos sociais.

O Facebook, notoriamente, não foi uma plataforma originalmente desenhada para o comércio eletrônico de produtos. Por fornecer uma plataforma aberta para cadastro e reunir milhões de usuários, esse tipo de atividade foi naturalmente acontecendo. O Facebook também permite a criação de páginas específicas para empresas, na qual a mesma pode fazer divulgação de seus produtos ou serviços, estreitando a relação entre ela e seus clientes.

Existem grupos muito populares no Facebook, como por exemplo, o “Tudo no Precinho Ò (Recife)”, que reúne cerca de 400 mil usuários, e se destina a venda e troca de qualquer tipo de produto. Há também o “LIVROS VENDAS E TROCAS RECIFE”. Ele reúne 10 mil pessoas e é focado no comércio informal de qualquer tipo de livro.

2.1.2 OLX

A OLX[2] (Online Exchange) é uma empresa multinacional de classificados on-line. Ela possui um dos 15 sites mais acessados do Brasil. O site “olx.com.br” recebeu cerca de 90 milhões de visitas no mês de setembro de 2019 (SIMILARWEB, 2019).

O sistema de classificados on-line da OLX permite ao seus usuários anunciar gratuitamente diversos produtos que vão desde eletrônicos a automóveis. Esses anúncios são categorizados de acordo com a natureza do produto. Algumas das categorias disponíveis, por exemplo, são: “Imóveis”, “Autos e Peças”, “Eletrônicos e celulares”, “Artigos Infantis” e “Música e Hobbies”.

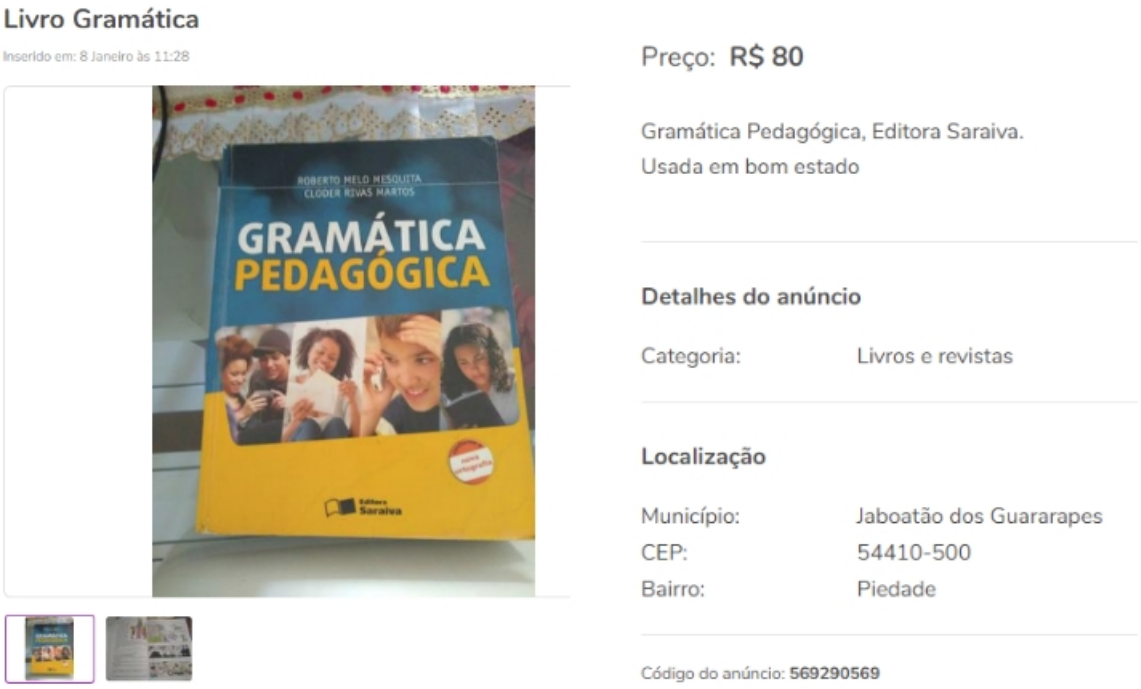
Dentro da categoria “Música e Hobbies” há uma subcategoria chamada “Livros e Revistas”. Ela engloba todos os tipos de conteúdo, dentre eles estão livros de preparatório para concurso até livros religiosos.

Na OLX, os anúncios criados pelos usuários estão geograficamente espalhados por todo o país. Um livro procurado, por exemplo,

[2]https://www.olx.com.br

pode ser encontrado, mas ter sido anunciado por outro usuário de algum ponto distante da cidade ou mesmo do país.

Figura 7: Anúncio de produto na OLX.



Fonte: Adaptado do site https://olx.com.br

Como pode-se observar na figura 7, um anúncio da OLX traz informações como: o título do anúncio, o preço do produto, uma breve descrição do mesmo, a localização geográfica do anunciante e a categoria do item anunciado. Por não ser uma plataforma especificamente focada em livros e que, portanto, aceita inúmeros tipos de produtos, como eletrônicos, veículos, móveis, imóveis, dentre muitos outros, a categorização de seus itens é muito generalista. No caso de material didático, por exemplo, seja ele de ensino fundamental ou voltado a concursos públicos na área de Direito, todos fazem parte da categoria “Livros e Revistas”.

Caso queira buscar um livro em específico, o usuário tem a sua disposição apenas uma busca textual, que é limitada, pois muitas vezes o

anunciante não insere informações como o nome do autor ou mesmo o título do livro, as quais ficam contidas apenas nas fotos, como é o caso da figura 7.

2.1.3 MERCADO LIVRE

O Mercado Livre[3] está dentre os dez sites mais acessados do Brasil. Ele contou com mais de 550 milhões de acessos durante o mês de julho de 2019 (SIMILARWEB, 2019). Nessa plataforma, os anúncios podem ser feitos diretamente por pessoas físicas ou empresas. Ele possui um sistema de reputação no qual os compradores podem avaliar positivamente ou negativamente os anunciantes ao fim de uma venda. Além de um meio de pagamento especial, no qual o dinheiro só é liberado para o vendedor após o cliente confirmar o recebimento do produto. Sendo esse recurso um dos responsáveis pelo grande sucesso do site, pois um dos principais inibidores para o comércio eletrônico é a desconfiança e o medo de fraude (EXAME, 2019).

Figura 8: Anúncio do Mercado Livre

Fonte: Mercado Livre.

[3]https://www.mercadolivre.com.br

Um anúncio no Mercado Livre (Figura 8) traz as seguintes informações: o título do livro ou nome do produto, o preço, opções de pagamento, informações sobre o envio e botões para compra ou adição ao carrinho de compras virtual. Diferentemente dos anúncios nas outras plataformas analisadas, como o Facebook e a OLX, o Mercado Livre possui uma categorização mais detalhada. O anúncio da figura 4, por exemplo, está dentro destas subcategorias, em ordem decrescente de abrangência, “Livros Universitários”, “Informática e Internet”.

No entanto, essas categorias poderiam ser ainda mais específicas, como por exemplo, agrupar os livros relacionados a um determinado curso de uma determinada universidade. Sendo essa uma das principais funcionalidades da aplicação proposta neste trabalho.

2.1.4 COMPARATIVO DE FUNCIONALIDADES

Na tabela 1 a seguir, é apresentado um comparativo entre as funcionalidades das aplicações presentes no mercado analisadas e a aplicação proposta neste trabalho.

Tabela 1: Comparativo entre as aplicações preexistentes analisadas e a aplicação proposta

	Limitação geográfica de anúncios a UFRPE	Anúncios gratuitos	Custo com deslocamento ou frete	Categorização de Livros Universitários por Cursos da UFRPE
Facebook	Não	Sim	Sim	Não
OLX	Não	Sim	Sim	Não
Mercado Livre	Não	Sim	Sim	Não
Aplicação proposta	Sim	Sim	Não	Sim

3. Fundamentação Técnica

Nesta seção serão apresentadas tecnologias e conceitos utilizados para o desenvolvimento da aplicação proposta. Todas as tecnologias utilizadas são de código aberto ou fornecem planos gratuitos por prazo indeterminado, como é o caso do Heroku e da Google Firebase. Dessa forma, nenhum valor foi gasto para o desenvolvimento e implantação da aplicação.

3.1 TECNOLOGIAS

3.1.1 BOOTSTRAP

Originalmente criado pelo Twitter[4], o Bootstrap[5] se tornou um dos mais populares frameworks front-end e projetos de código aberto do mundo (BOOTSTRAP, 2019).

Ele fornece uma biblioteca de componentes comumente utilizados em interfaces de sistemas web, como por exemplo, botões, alertas, barras de navegação, formulários, dentre outros.

Um recurso bastante poderoso é o sistema de grade, ou grid system, que permite facilmente a construção de páginas responsivas, as quais se adéquam às telas de diferentes dispositivos, sejam eles smartphones, tablets ou PCs.

Em 2019, o repositório oficial do Bootstrap no Github[6] contava com 137 mil estrelas. Sendo um dos mais populares.

3.1.2 FIREBASE

[4]https://twitter.com
[5]https://getbootstrap.com
[6]https://github.com

A Firebase[7] foi criada em 2009 e posteriormente foi comprado pela Google em 2014. Ela fornece diversos serviços de infraestrutura para aplicações móveis e web, dentre eles, base de dados não relacional, hospedagem para sites estáticos, funções remotas e espaço em nuvem para o armazenamento e recuperação de arquivos.

Uma base de dados é uma coleção organizada de dados. Ela pode ser armazenada localmente ou na nuvem. Todas as aplicações, sejam android, iOS ou web, precisam de uma base de dados. A ideia básica por trás da criação de uma base de dados é armazenar dados sistematicamente e recuperá-los quando requeridos.

O Firebase fornece uma API que permite a criação e a recuperação de dados em tempo real com apenas algumas linhas de código. Os dados são armazenados como JSON e acessíveis de qualquer plataforma (KHEDKAR et al., 2017). É utilizada a infraestrutura da Google, e em caso de crescimento da demanda pelo serviço, ele é automaticamente escalável.

Esse tipo de ferramenta é conhecida como BaaS ou Backend as a Service, e traz inúmeras vantagens, dentre elas a ausência de preocupação do cliente quanto a segurança dos dados, pois essa atribuição fica a cargo da Google. Outras vantagens são a alta disponibilidade e a total isenção de responsabilidade e custos com infraestrutura, além do ganho de tempo com o uso das APIs disponibilizadas (LUMMERTZ et al., 2018).

O seu diferencial está em poupar o desenvolvedor de ter trabalho com servidores, instalação de sistemas operacionais e atualizações de software. É disponibilizada uma API (Aplication Programming Interface) que permite a inserção e recuperação de dados em diversas linguagens de programação, como JavaScript, sem qualquer contato com o código que está sendo executado no servidor.

Na Firebase, são disponibilizados planos gratuitos com limites razoáveis, o que a tornou um atrativo para o seu uso nesse projeto.

[7]https://firebase.google.com

3.1.3 HEROKU

A Heroku[8] é uma empresa fundada em 2007. Ela force um modelo de computação em nuvem conhecido como PaaS (Platform as a Service). A Heroku suporta diversas linguagens de programação como Java, PHP, Scala, Clojure, Python e Ruby (BIH-HWANG et al., 2018).

A implantação de uma aplicação para a nuvem da Heroku ocorre de forma relativamente simples se comparada a outras plataformas. Ela fornece uma interface de linha de comando ou CLI (Comand Line Interface), a qual pode ser baixada diretamente no site da empresa, e permite o upload de um projeto Git[9]. Ao ser enviado para o nuvem, o projeto é automaticamente construído, implantado no servidor e acessível a partir de um determinado URL. Desse modo, o programador não precisará preocupar-se com instalação e manutenção de sistemas operacionais, o que é comum com máquinas virtuais, e além disso, deixará a responsabilidade de manutenção de toda a infraestrutura com a empresa.

A plataforma também fornece outra forma de implantação, na qual um projeto Git hospedado no Github pode ser associado a sua aplicação na Heroku. Dessa modo, a implantação pode ser feita através do web site da empresa, sem a necessidade de realizar operações através de linha de comando.

3.1.4 GROVVY

O Apache Groovy é uma linguagem de código aberto, orientada a objetos, dinamicamente tipada e totalmente compatível com Java. Ela foi criada com o intuito de suprir deficiências e adicionar novos recursos ao Java, dando suporte a funcionalidades que só posteriormente seriam implementadas no JDK. O Groovy é a linguagem utilizada no framework Grails (WEISSMANN, 2014).

[8] https://www.heroku.com
[9] Sistema de versionamento de software: https://git-scm.com

Malone et al., (2012) realizou um estudo comparativo entre projetos desenvolvidos em Java e projetos desenvolvidos com o framework Grails. Os resultados presentes na tabela 2 são referentes ao desenvolvimento de um CRUD para relatórios financeiros. Esse projeto envolveu três desenvolvedores e teve duração de cinco meses. De acordo com a tabela 2, é possível perceber uma grande diferença no total de linhas de código para o mesmo software escrito em Java e em Groovy/Grails. Houve uma redução de 14920 linhas de código em Java para 3661 linhas de código em Groovy/Grails.

Tabela 2: Linhas de código: Groovy/Grails vs Java

	Groovy/Grails	**Java**
Iteração 1	523	2663
Iteração 2	1539	6233
Iteração 3	1599	6024
Total	**3661**	**14920**

Fonte: (MALONE et al., 2012)

Como pode ser visto na figura 10, uma classe em Groovy, do ponto de vista do desenvolvedor, possui significativamente menos linhas de código em comparação a uma classe similar em Java. Para as classes escritas em Groovy, são gerados em tempo de compilação e inseridos no bytecode correspondente, os métodos, os modificadores de acesso e os construtores inerentes ao padrão Java Bean. Os modificadores de acesso de classes e métodos são por padrão públicos, enquanto os modificadores de acesso dos atributos são por padrão privados. Portanto, o desenvolvedor fica incumbido de definir apenas os atributos e os seus respectivos tipos em um determinada classe.

Na figura 10 abaixo, pode-se observar a classe Book escrita em Groovy. Os modificadores de acesso da classe e dos atributos ficam subentendidos, além dos métodos getters e setters e dos construtores.

Figura 9: Exemplo de classe em Groovy

```
1  package sebo.rural.deploy
2
3  class Book {
4
5      String titulo
6      String autor
7      String descricao
8      String urlFoto
9      Course curso
10     String periodo
11     String disciplina
12     String preco
13     Date dataCriacaoAnuncio
14     Student student
15 }
```

Fonte: Elaborado pelo autor.

Este tipo de convenção sobre configuração, poupa o programador de um trabalho repetitivo e torna o desenvolvimento mais ágil. A mesma classe acima, que em Groovy pôde ser implementada com 14 linhas, escrita em Java (Figura 11), possuirá 97 linhas. Isso representa uma redução, para essa classe específica, de aproximadamente 85% de linhas de código.

Pode-se verificar na tabela 2, que a redução de linhas para o projeto analisado por Malone et al. foi de aproximadamente 75%. Isso representa, para um conjunto grande de classes, uma redução de 3/4 do total de linhas de código para o mesmo software caso fosse escrito diretamente em Java, ou seja, há uma redução muito significativa no tempo gasto para escrita de código.

Figura 10: Exemplo de classe similar escrita em java

```java
package javaBean;

import java.util.Date;

public class Book {

    private String titulo;
    private String autor;
    private String descricao;
    private String urlFoto;
    private Course curso;
    private String periodo;
    private String disciplina;
    private String preco;
    private Date dataCriacaoAnuncio;
    private Student student;

    public Book() {

    }

    public Book(String titulo, String autor, String descricao, String urlFoto, Course curso, String periodo,
            String disciplina, String preco, Date dataCriacaoAnuncio, Student student) {
        super();
        this.titulo = titulo;
        this.autor = autor;
        this.descricao = descricao;
        this.urlFoto = urlFoto;
        this.curso = curso;
        this.periodo = periodo;
        this.disciplina = disciplina;
        this.preco = preco;
        this.dataCriacaoAnuncio = dataCriacaoAnuncio;
        this.student = student;
    }

    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
```

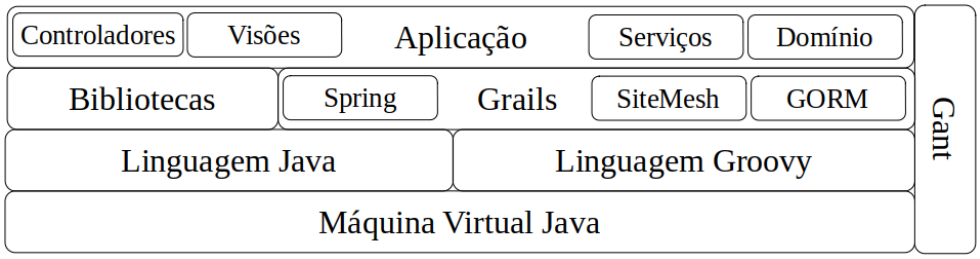
Fonte: Elaborado pelo autor.

3.1.5 GRAILS

O Grails é um meta-framework fullstack que tem como base a plataforma Java EE e emprega os já consagrados frameworks Spring e Hibernate. Sendo lançado em 2008, sob forte influência do Ruby on Rays, o Grails alia a versatilidade e simplicidade da linguagem de programação Groovy, que possui tipagem dinâmica e é menos verborrágica do que Java, às convenções sobre configuração do Rails. Fornecendo assim uma poderosa ferramenta para o desenvolvimento de aplicações para a JVM.

O banck-end da aplicação projetada e desenvolvida neste trabalho, que se trata de uma Restful API, foi totalmente desenvolvido com o Grails e implantado na PaaS (Platform as a Service) Heroku.

Figura 11: Arquitetura do Grails



Fonte: Adaptado de (JUDD et al., 2012)

Como pode ser visto na figura 12, uma aplicação Grails roda sobre a JVM, permitindo assim o trabalho conjunto de código Java e Groovy. Assim, funções e bibliotecas legadas escritas em Java podem facilmente ser acopladas a um projeto Grails.

Por padrão, o esqueleto do projeto trás pacotes para as classes de domínio, controladores, visão e serviços. Induzindo assim o programador a adotar o padrão MVC.

O GORM (Grails Object Relational Mapping) tem por base o Hibernate, possuindo uma camada de código Groovy por cima, e faz o mapeamento objeto relacional entre as classes de domínio e o banco de dados. Para cada classe criada no pacote de domínio, durante a compilação do projeto, serão criadas tabelas no banco de dados, sendo os atributos da classe mapeados em campos da tabela.

A arquitetura do Grails é composta por uma pilha de frameworks. Na sua versão 3, uma aplicação Grails tem por base o Spring Boot. Judd et al. (2008) desrevem frameworks utilizados pelo Grails, são eles:

- **SiteMesh**: responsável pelo mecanismo de layout, implementa o padrão decorator para a renderização de HTML, com componentes constantes nas páginas, como cabeçalho, rodapé e navegação.

- **Ajax:** o Grails usa três frameworks populares para utilização de Ajax que atendem as necessidades da Web 2.0. São eles: o script.aculo.us, o Rico, e o Prototype.

- **Jetty**: é um servidor de aplicação popular e rápido, que serve para dar ao Grails um ambiente de desenvolvimento completo.

- **HSQLDB**: é um banco de dados 100% Java. É possível configurá-lo para ser executado na memória ou ser mantido em disco. De forma padrão, o Grails o executa em memória.

- **JUnit**: framework usado para testes unitários. Para cada classe de controle criada a partir da CLI do Grails é criada uma classe de testes correspondente.

- **GORM**: Este framework tem por base o Hibernate, e é responsável pelo mapeamento objeto relacional no Grails.

3.1.6 ANGULAR

O Angular[10] é um framework muito popular, aberto para o público em 2010, e usado para o desenvolvimento do front-rnd de plicações web e mobile. Ele conta com mais de 50.000 estrelas no Github (GITHUB, 2019). O Angular foi originalmente escrito em JavaScript e intitulado de AngularJS, sendo posteriormente totalmente reescrito em TypeScript e renomeado para Angular.

A nova versão do framework traz inúmeros recursos, como uma poderosa interface de linha de comando, que permite a geração de esqueletos de código dos componentes. Além disso, ele é toda baseada em Web Componentes, os quais permitem o fracionamento dos elementos de uma aplicação, facilitando assim o seu reuso em outras telas ou mesmo em outras aplicações.

Devido a adoção do TypeScript, fica mais transparente para o programador o conceito de orientação a objetos, tendo cada componente uma classe correspondente, o que melhora a organização, manutenibilidade e legibilidade do código.

[10]https://angular.io

3.1.7 SPRING SECURITY

O spring Security é um framework poderoso e altamente customizável. Ele é comumente utilizado para a segurança de aplicações baseadas em Spring. Segundo o DEVMEDIA (2019) ele fornece as seguintes funcionalidades:

- Segmentação dos usuários do sistema em classes de usuários.
- Atribuir níveis de autorização à papéis de usuário.
- Aplicar regras de autenticação globais para recursos da aplicação.
- Aplicar regras de autorização a níveis da aplicação.
- Prevenir tipos de ataques comuns que manipulas ou roubam sessões de usuários.

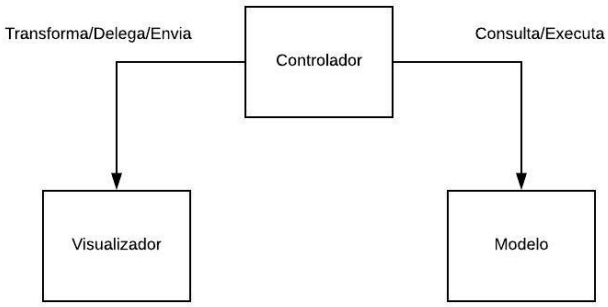
O Grails possui um plugin chamado de Grails Spring Security. Esse plugin permite facilmente a adoção do Spring Security por meio da adição de uma dependência em seu arquivo de configuração. Esse plugin foi utilizado para o emprego do Spring Security neste trabalho.

3.2 CONCEITOS

3.2.1 MVC

O MVC (Model View Controller) é um padrão de projeto muito popular em framewoks para desenvolvimento web. O desenvolvimento de software com os frameworks Spring, Django, Ruby on Rails e Grails é fortemente baseados nesse padrão. Ele foi inicialmente proposto na década de 70 por Trygve Reenskaug, durante visita ao Xerox PARC (Palo Alto Research Center). A princípio usado para o desenvolvimento de aplicações para desktop com interface gráfica, algo inovador para a época, foi posteriormente adotado em outros ambientes, como os frameworks para a construção de aplicações para a web (SINGH et al., 2018).

Figura 12: Padrão de projeto MCV



Fonte: Adaptado de (WEISSMANN, 2014)

No padrão MVC, como podemos ver na figura 9 acima, uma aplicação é dividida em três tipos de classes: modelo, visão ou visualizador e controlador. As classes de modelo representam entidades inerentes a determinada lógica de negócio, como por exemplo, no contexto deste trabalho: livro, estudante, anúncio, mensagem, dentre outros. As classes de visão são responsáveis pela exibição de informações das entidades de negócio para o usuário e o controlador faz o intermédio entre o modelo e a visão. É o controlador quem define como o sistema irá reagir a uma determinada entrada

do usuário. Dessa forma, há uma redução significativa no acoplamento, permitindo que uma camada do programa possa ser substituída com poucas alterações nas outras. Há também um ganho na manutenibilidade e compreensão do código, pois as funções de cada camada ficam bem definidas (GAMMA et al., 1995).

Em uma aplicação que implemente o padrão de projeto MVC, os usuários irão interagir com a camada de visão. Essa camada exibe informações, botões e espaços para entradas de dados. Ao clicar em um botão de submissão, por exemplo, uma determinada função em uma classe de controle é acionada, podendo receber dados provenientes da visão como parâmetro. Geralmente, o controlador, então, trabalha com a instância de um modelo e faz, indiretamente, a consulta ou a inserção de dados no banco de dados.

Singh et al. (2018) descrevem três vantagens do padrão MVC. São elas:

- **Separação de conceitos**: Todos os componentes no MVC são modularizados o que facilita a reusabilidade da lógica de negócio.

- **Especialização e foco do desenvolvedor**: Um projetista de interfaces pode trabalhar sem se preocupar com o modelo ou a lógica de negócio. Do mesmo modo, um administrador de dados pode trabalhar no modelo sem se preocupar com a interface gráfica.

- **Desenvolvimento paralelo por times separados**: O desenvolvimento pode ser feito de forma simultânea, resultando em menor interdependência e melhor utilização do tempo.

3.2.2 FRAMEWORK

Um Framework é um arcabouço de ferramentas, e no contexto de software consiste em um conjunto de funcionalidades pré implementadas que vão além de uma simples biblioteca, pois ele induz ao desenvolvedor o uso de uma determinada estruturação para o seu código.

Exemplos de frameworks utilizados neste trabalho são o Spring e o Angular. O Spring Framework, por exemplo, lida com tarefas complexas como inversão de controle, injeção de dependências e gerenciamento do ciclo de vida de objetos, permitindo ao programador apenas configurar o comportamento do mesmo.

Ao longo dos anos, através da popularização de conceitos como Convenção Sobre Configuração e Não Repita a Si Mesmo (Don't Repeat Yourself), a necessidade do preenchimento de arquivos de configuração tem sido reduzida e o programador pode dedicar-se mais a lógica de negócio, ou seja, ele pode investir mais tempo no desenvolvimento efetivo do software e menos em arquivos XML.

3.2.3 APLICAÇÃO DE PÁGINA ÚNICA

É uma aplicação que se assemelha com a experiência de uso de uma aplicação desktop. Ela carrega totalmente os arquivos necessários para sua execução na requisição inicial, assim, os componentes da página podem ser rapidamente substituídos por outros a partir da interação com o usuário.

Quando comparamos aplicações de página única com aplicações web tradicionais podemos ver que há uma analogia entre estados da aplicação de página única e páginas web tradicionais. Sendo a navegação em páginas web de aplicações tracionais análoga à navegação entre estados de uma aplicação de página única.

Uma aplicação de página única é composta por componentes individuais que podem ser reposicionados ou atualizados independentemente uns dois outros, sem atualizar ou recarregar toda a página, assim, uma página inteira não precisa ser recarregada em cada ação de usuário (JADHAV et al., 2015).

3.2.4 RESTful API

REST (Representational State Transfer) é um padrão de arquitetura de comunicação baseado na web. Sendo comumente aplicado para o desenvolvimento de serviços. Ele usa HTTP como o protocolo para comunicação de dados. Este padrão foi introduzido por Fielding (2000).

Os métodos do HTTP usados em arquiteturas baseadas no REST são: GET, POST, PUT e DELETE. (NUGROHO et al., 2017).

Vantagens da arquitetura RESTful decorrem de sua característica stateless. Dessa forma, ele é escalável, interoperável e simples (MORÓN_RODRIGUEZ et al., 2017).

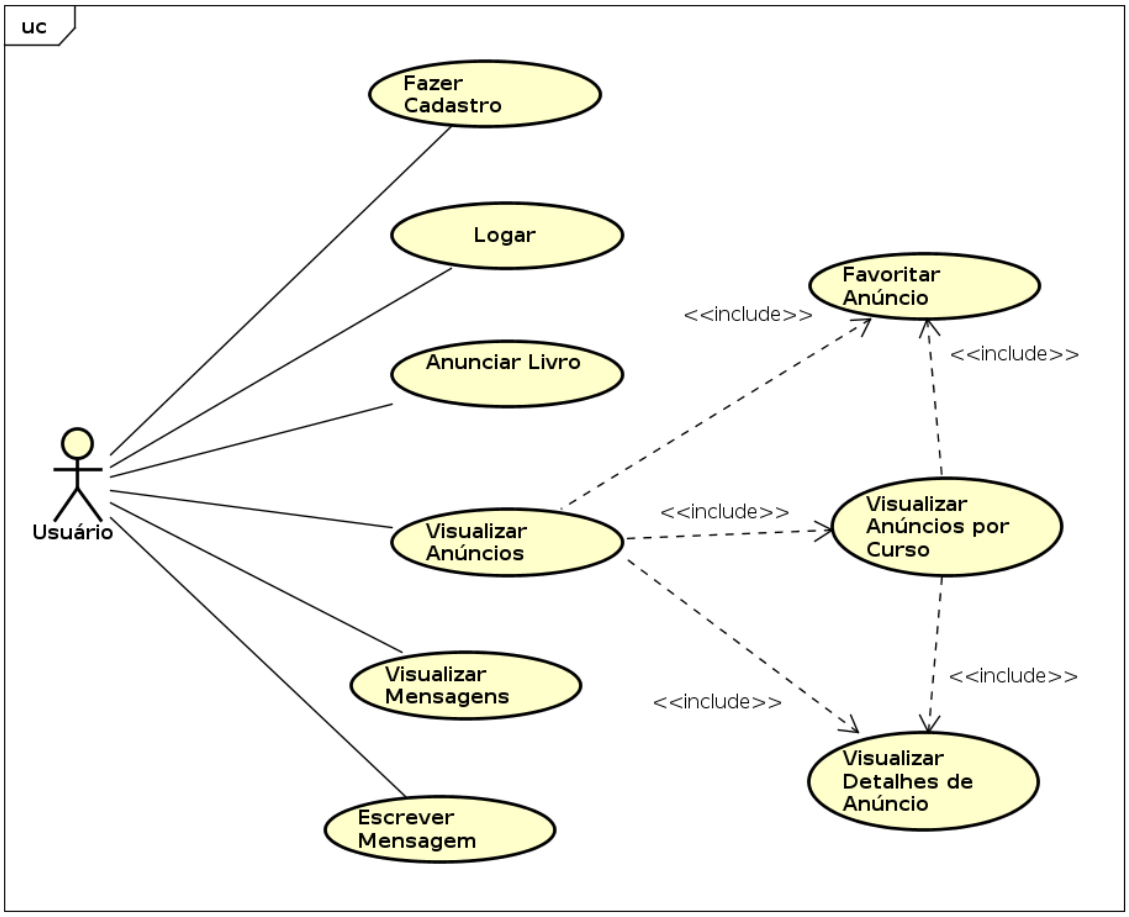
4. Projeto da Aplicação Proposta

Nesta seção será apresentado o projeto da aplicação proposta. Serão expostas as funcionalidades projetadas bem como a sua arquitetura.

4.1 Casos de Uso

Na figura abaixo (Figura 13) estão expostos os casos de uso projetados para a aplicação. São eles: Fazer Cadastro, Logar, Anunciar Livro, Visualizar Anúncios, Visualizar Mensagem e Escrever mensagem. No caso de uso Visualizar Anúncio estão inclusos: Favoritar Anúncio, Visualizar Anúncios por Curso e Visualizar Detalhes de Anúncio.

Figura 13: Diagrama de casos de uso da aplicação proposta



Fonte: Elaborado pelo autor.

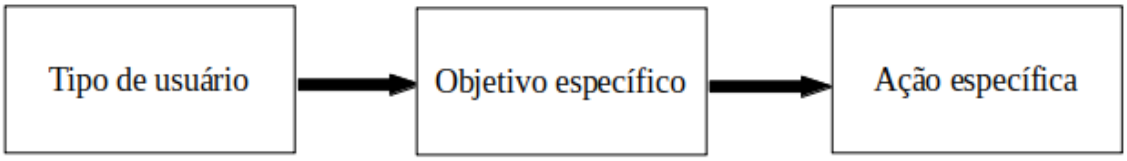
Dentre as principais funcionalidades propostas estão a criação de anúncios de livros pelos usuários e a visualização dos anúncios existentes. Essa visualização deve ser possível a partir da escolha um curso específico, ou seja, por meio de um determinado curso selecionado pelo usuário serão filtrados e exibidos na tela apenas os anúncios relacionados a esse curso.

4.2 ESTÓRIAS DE ÚSUÁRIO

Segundo Chopade et al. (2017), o processo ágil de desenvolvimento de software tem início com a escrita de estórias de usuário. Elas são pequenas afirmações formadas a partir de requisições de clientes do sistema a ser desenvolvido. As estórias de usuário têm o seguinte formato: Como um <tipo de usuário> eu quero <algum objetivo> então <alguma ação>. Por exemplo: Como um administrador eu quero logar no sistema então eu poderei usar a aplicação.

Na figura abaixo (figura 14) podemos ver uma típica estória de usuário. Ela é composta por um tipo de usuário, um objetivo específico e opcionalmente uma ação específica.

Figura 14: Típica estória de usuário



Fonte: (CHOPADE et al., 2017).

Segundo Dalpiaz et al. (2018), estórias de usuário têm um formato pré-definido com o intuito de capturar três elementos de uma requisição de funcionalidade:

1. Quem deseja a funcionalidade.

2. Qual funcionalidade a parte interessada quer para o sistema.
3. Por que a parte interessada quer esta funcionalidade. (opcional).

Desta forma, é possível documentar as requisições de funcionalidades de forma simples e rápida. Num segundo momento, a equipe de desenvolvedores poderá definir pesos para cada uma das estórias de usuários criadas. As mais complexas terão um peso maior e as menos complexas terão um peso menor. Assim, a equipe poderá definir, a partir da viabilidade, quais estórias de usuários deverão ser implementadas primeiro.

Na tabela a seguir são exibidas as estórias de usuário para os casos de uso apresentados na figura 13.

Tabela 3: Estórias de usuário da aplicação proposta

Caso de Uso	Estória de Usuário
Fazer Cadastro	Como um estudante quero me cadastrar no sistema, assim poderei acessar a aplicação.
Logar	Como um estudante quero logar no sistema,
Anunciar Livro	Como um estudante quero anunciar um livro.
Visualizar Mensagens	Como um estudante quero visualizar mensagens enviadas por outros usuários para mim.
Escrever Mensagem	Como um estudante quero escrever mensagens para outros usuários.
Visualizar Anúncios	Como um estudante quero visualizar anúncios meus e de outros usuários.
Favoritar Anúncio	Como estudante quero favoritar um anúncio.
Visualizar Anúncios por Curso	Como um estudante quero visualizar anúncios relacionados a um curso específico.
Visualizar Detalhes de Anúncio	Como estudante quero visualizar um anúncio em detalhes.

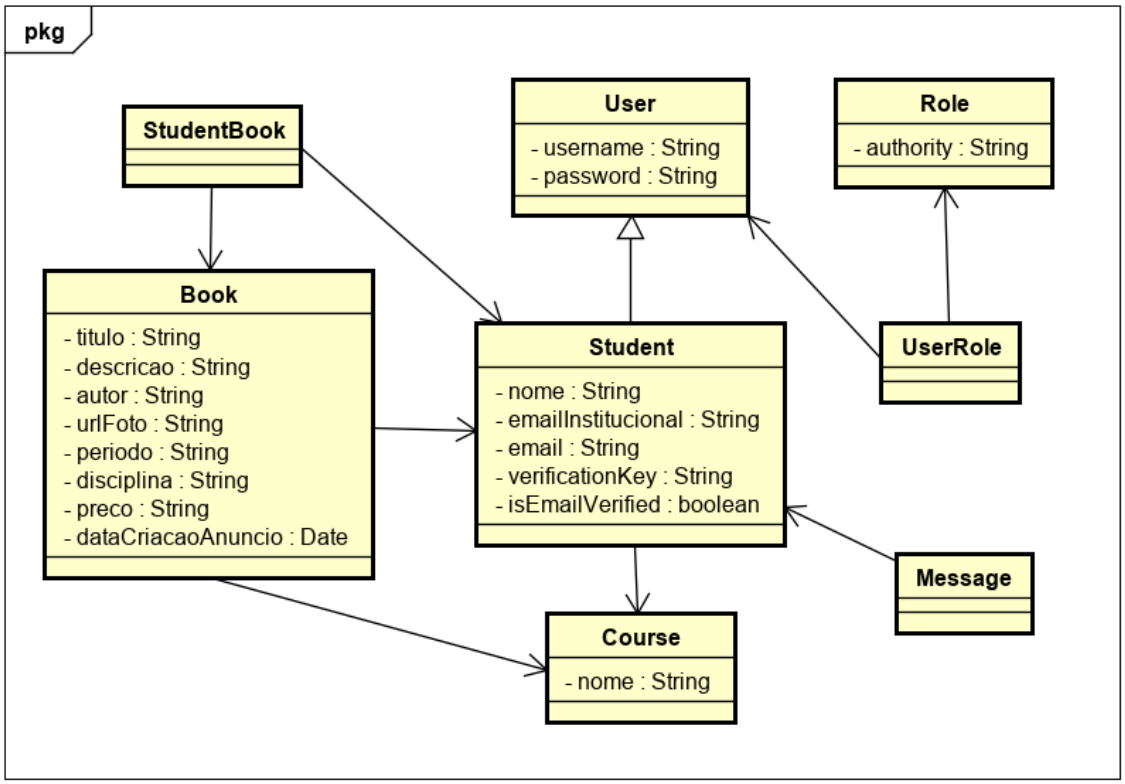
Fonte.: Elaborado pelo autor.

4.3 CAMADA DE MODELO

Uma das camadas do padrão de projeto MVC é a camada de modelo ou model. Nesta camada estão as entidades de um sistema, ou seja, as classes que representam e tipificam os elementos inerentes a um determinado modelo de negócio.

Na figura 15, está o diagrama de classes da camada de modelo da aplicação proposta. A camada será composta pelas seguintes classes: Student, Book, Course, User, Role, UserRole e StudentBook. A seguir é feita uma breve descrição sobre a função de cada uma delas e os seus respectivos atributos.

Figura 15: Camada de modelo da aplicação proposta



Fonte: Elaborado pelo autor.

- **Student**: Esta classe representará o estudante, o principal usuário da aplicação, e possuirá os atributos: nome, email, verificationKey e isEmailVerified. Os atributos nome e email guardarão o nome e o email

de um estudante, respectivamente. O atributo verificationKey armazenanará uma chave de verificação. Essa chave será um número aleatório gerado pelo servidor e enviado por email ao estudante, no momento do cadastro do mesmo, com o intuito de realizar a verificação do e-mail institucional. A propriedade isEmailVerified será utilizada para checar se o email do estudante foi ou não verificado. Sendo a mesma alterada para verdadeiro quando o código de verificação for confirmado.

- **Book**: É a entidade responsável por caracterizar um livro a ser anunciado no sistema. Ela possuirá os seguintes atributos: titulo, descricao, autor, urlFoto, periodo, disciplina, preco, dataCriacaoAnuncio. O atributos título, descrição, autor, preço, período e disciplina armazenarão, respectivamente, o título de um livro, uma descrição sobre ele, um nome de autor do mesmo, o seu preço e o período e disciplina aos quais ele está relacionado. O atributo urlFoto armazena a url de uma imagem, enviada pelo usuário, relacionada ao livro.

- **Course**: Classe responsável por modelar os cursos universitários. Possuirá o atributo nome. Esse atributo guardará o nome de um determinado curso.

- **User**: A classe user, caracterizará qualquer tipo de usuário do sistema, possuindo os atributos username e password. A propriedade username, armazenará um texto a ser usado pelo usuário como login e o atributo password armazenará a senha definida pelo usuário.

- **Role**: Esta classe identificará os papéis de usuários da aplicação. Esses papéis serão: Student e Admin, tendo cada um níveis de acesso diferentes.

- **UserRole**: Classe que relacionará um determinado usuário ao seu papel. Possuirá os atributos user e role. Sendo a propriedade user um objeto da classe User e a propriedade Role, um objeto da classe Role.

- **Message**: Esta Entidade será usada para denotar mensagens que podem ser trocadas pelos estudantes. Ela disporá de dois atributos: remetenteStudent e destinatarioStudent.
  .
- **StudentBook**: Classe responsável por relacionar um objeto da classe Student a outro da classe Book. O seu intuito será definir os livros favoritados por um determinado usuário do tipo Student.

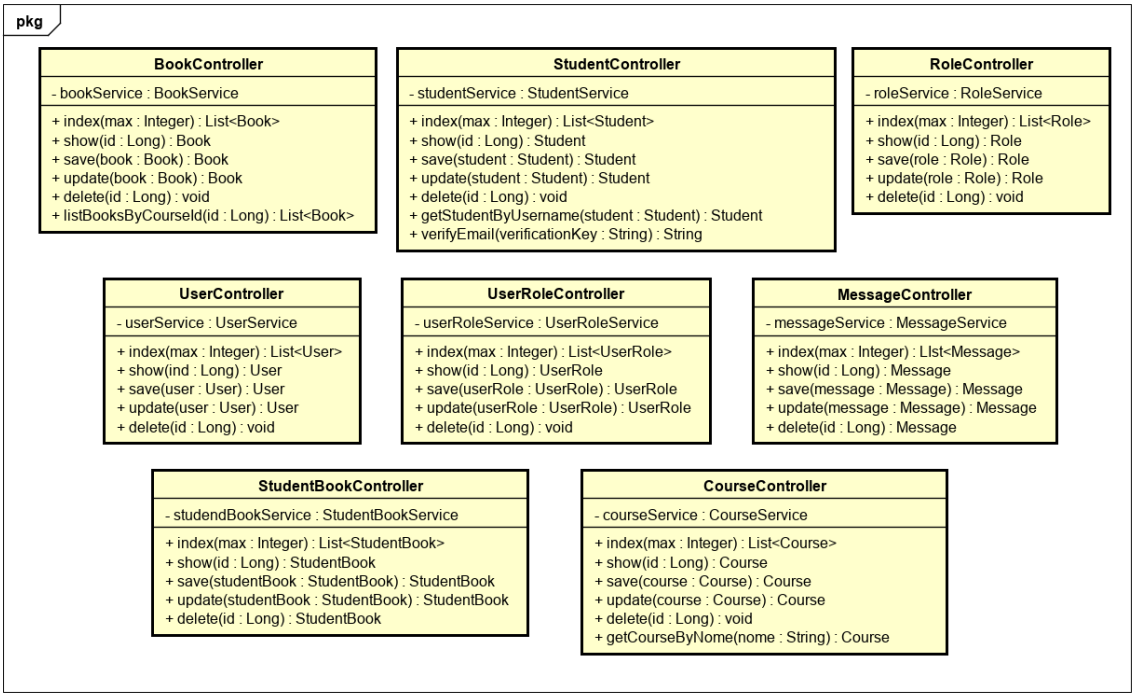
4.4 CAMADA DE CONTROLE

A camada de controle é uma das camadas definidas pelo padrão de projeto MVC. Esta camada será provocada a partir de interações do usuário com a camada de visão. Ela receberá dados inseridos como parâmetro pelo usuário por meio da camada de visão e, a partir de instâncias de classes de serviço, fará as operações CRUD.

Na aplicação proposta, as funções ou métodos de classes da camada de controle serão invocadas a partir de requisições HTTP. Serão utilizados os métodos GET, POST, PUT e DELETE.

Cada uma das classes de controle possuirá os métodos: index, show, save, update e delete. Essas funções serão responsáveis, respectivamente, por buscar uma lista de objetos na base de dados, buscar um objeto específico na base de dados a partir do seu identificador e salvar, atualizar ou deletar um determinado objeto na base de dados. Essas operações serão realizadas com o auxílio de instâncias das classes de serviço correspondentes.

Figura 16: Camada de controle da aplicação proposta



Fonte: Elaborado pelo autor.

Como mostrado na figura 18, a camada de controle será composta pelas seguintes classes: StudentController, BookController, CourseController, StudentBookController, MessageController, UserController, RoleController e UserRoleController. A seguir é feita uma descrição de cada uma delas:

- **StudentController**: Esta classe estará relacionada a entidade Student. Ela fará, em conjunto com a classe de serviço StudentService, operações CRUD para objetos do tipo Student.

- **BookController**: Esta classe de controle fará, em conjunto com a classe de serviço BookService, operações CRUD para a objetos do tipo Book.

- **CourseController**: A classe de controle CourseController realizará, por meio de instância da classe de serviço CourseService, operações CRUD para objetos do tipo Course.

- **StudentBookController**: Esta classe de controle será responsável por operações CRUD para objetos do tipo StudentBook. Ela instanciará a classe de serviço StudentBookService para realizar tais operações.

- **MessageController:** Esta classe de controle realizará, em conjunto com a classe de serviço MessageService, operações CRUD para objetos do tipo Message.

- **UserController:** Esta classe de controle fará, em conjunto com a classe de serviço UserService, operações CRUD para objetos do tipo User.

- **RoleController:** Esta classe de será responsável por executar, em conjunto com a classe de serviço RoleService, operações CRUD para objetos do tipo Role.

- **UserRoleCotroller:** Classe de controle que fará, em conjundo com a classe de serviço UserRoleService, operações CRUD para objetos do tipo UserRole.

4.5 CAMADA DE SERVIÇO

O framework Grails sugere ao desenvolvedor o uso de classes de serviço. Ele possui por padrão uma pasta chamada services, destinada a receber essas classes. A seguir serão exibidas as classes de serviço projetadas e as suas respectivas funcionalidades.

Na camada de serviço, encontram-se, naturalmente, as classes de serviço. Estas classes realizam operações com a base de dados. No Grails, métodos comumente usados para realizar operações com registros do banco de dados como: recuperar, deletar, salvar, listar e contar, podem ser gerados dinamicamente em tempo de compilação. Isso pode ser definido pelo

programador através da anotação *Service* que está presente na biblioteca *grails.gorm.services.Service. (Figura 16).*

Figura 17: Classe de serviço no Grails

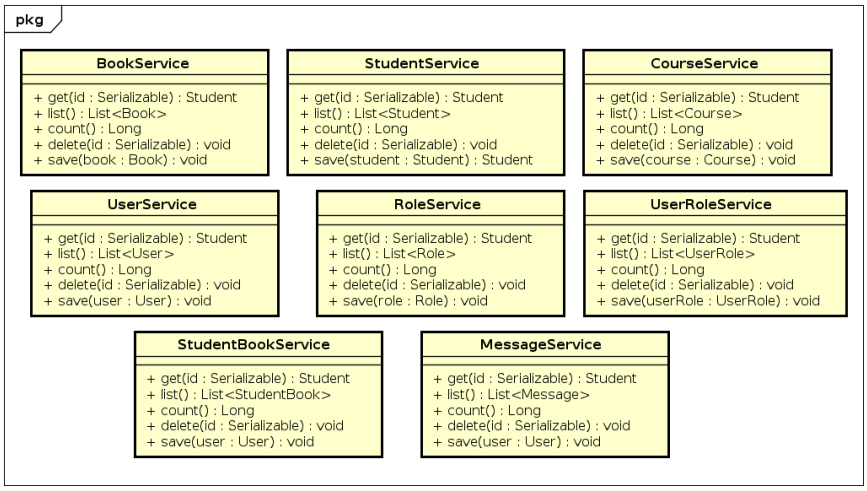
```
import grails.gorm.services.Service

@Service(Book)
interface BookService {

    Book get(Serializable id)

    List<Book> list(Map args)

    Long count()

    void delete(Serializable id)

    Book save(Book book)

}
```

Fonte: Elaborado pelo autor.

Na figura 16, vemos a declaração da interface BookService. Por meio desta interface e da anotação @Service(Book) será gerada em tempo de compilação a classe BookService. Esta classe possuirá os métodos implementados: get, list, count, delete e save para o tipo Book. A classe BookService poderá, então, ser instanciada em classes da camada de controle e realizará as operações de CRUD com a base de dados.

Para o projeto da aplicação proposta, foram definidas as seguintes classes de serviço: BookService, StudentService, CourseService, UserService, RoleService, UserRoleService, StudentBookService e MessageService. Essas classes podem ser vistas no diagrama de classes da figura 17.

Figura 18: Camada de serviço da aplicação proposta



Fonte: Elaborado pelo autor.

De acordo com a figura 17 acima, podemos observar que cada entidade da camada de modelo, figura 15, possuirá uma classe de serviço associada. As classes de serviço serão responsáveis por realizar as operações CRUD para cada uma das entidades do sistema. Estas classes serão instanciadas pelos controladores correspondentes, desse modo, os seus métodos serão invocados a partir da camada de controle.

5. Implementação da Aplicação Proposta

5.1 COMPONENTES DO FRONT-END

O front-end da aplicação proposta foi construído com base no framework Angular. Por se tratar de uma SPA (Single Page Application), todo o front-end é baseado em componentes. Esses componentes são invocado a partir de rotas pré-configuradas por meio do Angular. Cada um dos componentes possui o próprio escopo de variáveis. Eles são constituídos por um um arquivo HTML, um arquivo CSS e um arquivo TypeScript.

O componentes construídos para a aplicação proposta são: CreateBookComponent, CreateUserComponent, EditBookComponent, ListBookComponent, HomePageComponent, UserLoginComponent, OwnBookDetailsComponent, ShowBooksByStudentComponent, OwnBookDetailsComponent, NavbarComponent e FooterComponent.

Os componentes criados são descritos a seguir:

- **CreateBookComponent**: Componente responsável por desenhar o fomulário para a criação de um anúncio de livro e gerenciar as requisições e envios de dados para o back-end durante a execução deste formulário.

- **CreateUserComponent**: Este componente, quando invocado, fará o desenho na tela do formulário de criação de usuário. Ele também é responsável por realizar as requisições e envios de dados para o back-end durante a execução do formulário de criação de usuário.

- **EditBookComponent**: Componente encarregado de desenhar a tela de edição de anúncio e realizar as requisições e envio de dados relacionados a atividade de edição de anúncio.

- **ListBookComponent**: Este componente faz a listagem de livros da página inicial do acervo. Ele faz a requisição ao back-end e desenha uma lista de 10 livros na tela.

- **HomePageComponent**: Este componente é responsável pelo desenho da página inicial, ou seja, da primeira página carregada a partir da url raiz.

- **UserLoginComponent**: Componente incumbido de desenhar o formulário na tela e realizar o gerenciamento de requisições e envio de dados ao back-end.

- **OwnBookDetailsComponent**: Este componente é responsável pela tela de exibição de anúncios específicos de um determinado usuário.

- **NavbarComponent**: Componente responsável pelo desenho da barra de navegação e execução das ações invocadas por cliques do usuário.

- **FooterComponent**: Componente responsável pelo desenho do rodapé da página.

5.2 NAVEGAÇÃO ENTRE COMPONENTES DA APLICAÇÃO PROPOSTA

A navegação entre os componentes da aplicação proposta está exibida na figura 19 abaixo. Como característica de uma aplicação de página única, os componentes NavbarComponent e FooterComponent estarão sempre carregados, sendo o componente central substituído de acordo com o fluxo de

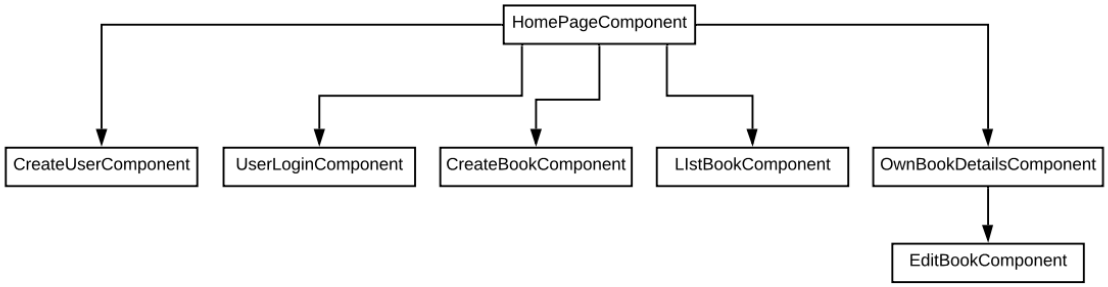
uso da aplicação. A partir do HomePageComponent, e em função de cliques em itens da barra de navegação, os demais componentes serão invocados.

Em uma SPA, todo o front-end da aplicação é baixado de uma vez pelo cliente. Isso implica que todos os componentes já estarão disponíveis para carregamento. Dessa forma, a navegação pelos componentes acontece de forma bastante fluída, não havendo a necessidade de recarregamento da página.

As principais vantagens da divisão do front-end da aplicação em componentes são a melhoria na manutenibilidade do código e a possibilidade de reuso dos componentes, pois eles possuem funcionalidades bem definidas e podem ser reutilizados com poucas modificações.

No Angular, a invocação dos componentes é configurada através da definição de rotas. O framework possui um arquivo de configuração (Figura 20) no qual os caminhos ou URLs e os respectivos componentes a serem invocados pelos mesmos podem ser definidos pelo programador.

Figura 19: Rotas da Aplicação proposta



Fonte: Elaborado pelo autor.

Figura 20: Configuração de rotas no Angular

```
const routes: Routes = [
  {
    path: 'createUser',
    component: CreateUserComponent
  },
  {
    path: 'userLogin',
    component: UserLoginComponent
  },
```

Fonte: Elaborado pelo autor.

5.3 COMUNICAÇÃO VIA API RESTFUL

Toda a comunicação com o back-end da aplicação proposta é feita via métodos HTTP: GET, POST, PUT, DELETE. Esse estilo arquitetural é denominado de REST.

Na tabela 4 abela a seguir, são exibidas as URIs utilizadas para a comunicação e conseguinte troca de informações entre cliente e servidor para a classe Book. Esse padrão é seguido para todas as classes de modelo e suas respectivas classes de controle associadas. A partir do nome da entidade, definido através da URI, e do método HTTP correspondente, os métodos index, show, save, update, delete das entidades são invocados. Em seguida, dados são retornados para o cliente via JSON (JavaScript Object Notation) ou salvos na base de dados.

Tabela 4: Mapeamento de urls para operações com objetos da classe Book

Requisição HTTP	Método Acionado	Descrição
GET /api*/book*	Index	Retorna uma lista de livros.
GET /api*/book/{id}*	show	Retorna um livro específico.
POST /api*/book/*	save	Cria um novo livro.
PUT /api*/book/{id}*	update	Atualiza um livro.
DELETE /api*/book/{id}*	delete	Deleta um livro.

Fonte: Elaborado pelo autor.

5.4 TOKEN DE ACESSO

Para ter acesso aos endpoints, o usuário precisará realizar login. O login pode ser feito através do URI /api/login, usando do método HTTP POST e passando no cabeçalho da requisição um objeto JSON com os respectivos

nome de usuário e senha. Na figura 21 a seguir a seguir, é possível observar o retorno do servidor para uma solicitação de login bem sucedida. O token de acesso gerado, então, deve ser inserido no cabeçalho das requisições para a obtenção de acesso aos endpoints. Note-se o papel ou role do usuário merciof29, o qual é ROLE_STUDENT, sendo este mais um recurso de segurança, além do token, para o controle de acesso a determinados recursos dos endpoints, como a deleção de conteúdo, por exemplo.

A chave de acesso se trata de um JWT (JSON Web Token), sendo gerado pelo servidor a partir de uma chave secreta. Esse chave é definida pelo programador em um dos arquivos de configuração do framework Grails.

Figura 21: Retorno de token de acesso para usuário


```
{
    "username": "merciof29",
    "roles": [
        "ROLE_STUDENT"
    ],
    "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJwcmluY2lwYWwiOiJINHNJQUFBQUFBQUFBSlZTdjA4VVFSUitleDVCQTFFdzBjUUNHN0V6ZXdsVzVpcUJrNFJzR
    HVQZE5aaG81bWJmclFPek04dk1MTncxNUNvc0tEQUNpWW10SmYrSk5Qd0JSQXRiYWx2ZkxCeDcyaENuMnJ6NTl2djE1dVFDSnF5QjU0bGhRdG93azNraVZHZ3pJMVJp
    a2VkR3VFR1lXelF4dWdLeFVnQTdOSUhMRTFRZ2lLQWlZZ2Yzb3cyMnpXcVNxYVMyMXQxQTd1cDlBd3ZhSkZlTVBjTlMzTkZtTTd6bTV0cmdYd0lsZGZDdEFwUHJNTXM
    0MTdseVRhMGFcL1V3WWpOZGhwcHhGbW1cLzYwUU5PTjZpY1lOS09ReWRSc2E3RU9JSXBscnNQbWxRRldnZjNMczNtVHNoYUMxMDlndHNaczViY1wvWk9rNWJ4MWYrOX
    RLa3F3QmJ0UTdXY0JIZXJ1cVllR25pZGMwbEpTYXFHVm5lK29WTWVpSjd3NDhRXC9uUHA4ZGZCMTJLZ0RVeWJPYlwveW5uanhaaCtQM2Q3OGRGMFFGMzhIRE1lZ21yO
    XpOeU0xc3l0dzE2NWZNdnJ3K1BMejYrdlVYS0h2SHFcL1wvY3hcL1wvS3F1Y0dTVGpObW1OTmpPeUxhbmFyXC9KdkxGbThsSFd4aUVMWkZtRXVsRktZZnh0VVJKVEhH
    clJzdFIzdzZtMzZ4RmpmZXRkbWU1MFd6N3daMFVEUmU2dFwvQ0N0TzhXMGYzT3dralR4dlpcL2ZUbzllUEtEZUZaaFlwdkpIS241bVJMVXpOTXVtcjJUNDdtcG81XC8
    3Ulk3Um1cLzREdjNcL1RjaGNEQUFBPSIsInN1YiI6Im1lcmNpb2YyOSIsInJvbGVzIjpbIlJPTEVfU1RVREVOVCJdLCJleHAiOjE1NzI0OTYzNzAsImlhdCI6MTU3Mj
    Q5Mjc3MH0.HvKTvZ63bwJDs06LAxWXefSY-FASqLnKapZX6VtJvyE"
}
```


Fonte: Elaborado pelo autor.

5.5 CONTROLE DE ACESSO ÀS URLS

O controle de acesso às funções do back-end, é feito através de um arquivo de configuração do framework Grails. Na figura 22, pode-se observar que através das URIs são definidos quais papeis e através de quais métodos HTTP determinados recursos poderão ser acessados. Desta forma, determinados serviços estarão disponíveis apenas para usuários logados enquanto outros estarão abertos.

Figura 22: Controle de acesso às URLs

```
grails.plugin.springsecurity.controllerAnnotations.staticRules = [
    [pattern: '/',                                access: ['permitAll']],
    [pattern: '/error',                           access: ['permitAll']],
    [pattern: '/index',                           access: ['permitAll']],
    [pattern: '/index.html',                      access: ['permitAll']],
    [pattern: '/newHomePage',                     access: ['permitAll']],
    [pattern: '/shutdown',                        access: ['permitAll']],
    [pattern: '/assets/**',                       access: ['permitAll']],
    [pattern: '/**/js/**',                        access: ['permitAll']],
    [pattern: '/**/css/**',                       access: ['permitAll']],
    [pattern: '/**/images/**',                    access: ['permitAll']],
    [pattern: '/**/favicon.ico',                  access: ['permitAll']],
    [pattern: '/book/**',                         access: ['ROLE_STUDENT']],
    [pattern: '/user/**',                         access: ['ROLE_STUDENT']],
    [pattern: '/role/**',                         access: ['ROLE_STUDENT']],
    [pattern: '/userRole/**',                     access: ['ROLE_STUDENT']],
    [pattern: '/student/**',                      access: ['ROLE_STUDENT']],
    [pattern: '/course/**',                       access: ['ROLE_STUDENT']],
    [pattern: '/student/**',                      access: ['permitAll'], httpMethod: 'POST'],
    [pattern: '/student/verifyEmail',             access: ['permitAll'], httpMethod: 'GET'],
    [pattern: '/student/getStudentByUsername',    access: ['permitAll'], httpMethod: 'POST'],
    [pattern: '/book/listBooksByCourseId',        access: ['permitAll'], httpMethod: 'GET'],
    [pattern: '/course/**',                       access: ['permitAll'], httpMethod: 'GET'],
    [pattern: '/course/getCourseByNome',          access: ['permitAll'], httpMethod: 'POST'],
    [pattern: '/userRole/**',                     access: ['permitAll'], httpMethod: 'POST']
]
```

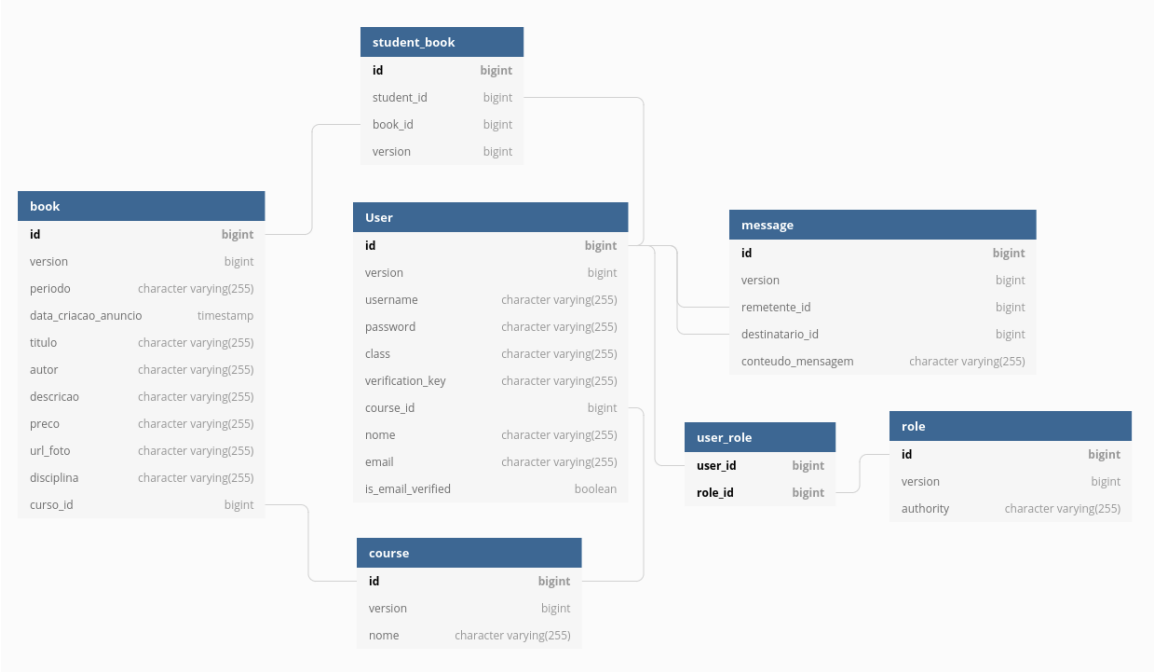
Fonte: Elaborado pelo autor.

5.6 PERSISTÊNCIA DE DADOS

O dicionário da base de dados implementada está descrito nas tabelas 5, 6, 7, 8, 9 e 10 a seguir. O SGBD (Sistema Gerenciados de Banco de Dados) utilizado foi o PostgreSQL. Foram criadas as tabelas: book, student_book, User, message, course, user_role e role. Estas tabelas estão diretamente associadas a classes, pois foram definidas pelo mapeamento objeto relacional do Hibernate. Desse modo, para cada uma das classes definidas na pasta domain do framework Grails, será criada uma tabela correspondente, sendo os atributos das mesmas transformados em campos de tabelas.

Diferentemente do uso comum do Hibernate, no qual o desenvolvedor deve inserir anotações no código para associá-lo a tabelas e campos, no Grails isso acontece de forma subentendida. Toda classe criada dentro da pasta domain terá por padrão uma tabela com o seu nome criada e os atributos desta classe serão os campos da mesma. Esta é uma aplicação do conceitos de Convenção Sobre Configuração, o qual permite que os programadores construam aplicações robustas com o mínimo de configuração possível (WORTH et al., 2012).

Figura 23: Diagrama físico da base dedados



Fonte: Elaborado pelo autor.

Como pode-se observar na figura 23, a tabela User possui a primeira letra maiúscula pois “user” é um termo reservado no PostgreSQL. Essa tabela é incumbida de persistir os dados dos usuários, e engloba as entidades User e Student em uma única tabela.

Tabela 5: Dicionário de dados da tabela User

Nome da tabela: **User**				
Nome do campo	**Tipo de dado**	**Tamanho do campo**	**Descrição**	**Tipo de dado**
id	bigint		Identificador	Chave primária
version	bigint			
username	character	255	Nome para login	
password	character	255	Senha para login	
class	character	255		
verification key	character	255	Chave de verificação para validação de e-mail	
course_id	bigint		Id do curso	Chave estrangeira

nome	character	255	Nome do usuário
email	character	255	E-mail do usuário
isEmailVerified	boolean		Campo utilizado para verificar se o e-mail foi ou não verificado

Fonte: Elaborado pelo autor.

Tabela 6: Dicionário de dados da tabela student_book

Nome da tabela: **student_book**				
Nome do campo	**Tipo de dado**	**Tamanho do campo**	**Descrição**	**Constraint**
student_id	bigint		Id do estudante	chave estrangeira
book_id	bigint		Id do livro	chave estrangeira
version	bigint			

Fonte: Elaborado pelo autor.

Tabela 7: Dicionário de dados da tabela book

Nome da tabela: **book**				
Nome do campo	**Tipo de dado**	**Tamanho do campo**	**Descrição**	**Constraint**
id	bigint		Identificador	Chave primária
periodo	character	255	Período do estudante dono do anúncio	
data_criacao_anuncio	timestamp		Data na qual o anúncio foi criado	
titulo	character	255	Título do livro anunciado	
autor	character	255	Autor do livro anúnciado	
descricao	character	255	Descrição sobre o livro anunciado	
preco	character	255	Preço do livro anunciado	
url_foto	character	255	Url da foto do livro anunciado	
disciplina	character	255	Disciplina na qual o livro anunciado é	

			usado	
curso_id	bigint		Id do curso ao qual o livro faz parte da bibliografia	Chave estrangeira

Fonte: Elaborado pelo autor.

Tabela 8: Dicionário de dados da tabela message

Nome da tabela: **message**				
Nome do campo	**Tipo de dado**	**Tamanho do campo**	**Descrição**	**Constraint**
id	bigint		Identificador	Chave primária
remetente_id	bigint		Id do usuário remetente	Chave estrangeira
destinatario_id	bigint		Id do usuário destinatário	Chave estrangeira
version	bigint			

Fonte: Elaborado pelo autor.

Tabela 9: Dicionário de dados da tabela course

Nome da tabela: **course**				
Nome do campo	**Tipo de dado**	**Tamanho do campo**	**Descrição**	**Constraint**
id	bigint		Identificador	Chave primária
nome	character	255	Nome do curso	
version	bigint			

Fonte: Elaborado pelo autor.

Tabela 10: Dicionário de dados da tabela role

Nome da tabela: **role**				
Nome do campo	**Tipo de dado**	**Tamanho do campo**	**Descrição**	**Constraint**
id	bigint	255	Identificador	Chave primária
authority			Nome do papel	
version	bigint			

Fonte: Elaborado pelo autor.

5.7 ARMAZENAMENTO E RECUPERAÇÃO DE IMAGENS

Uma das funcionalidades mais importantes do sistema é a criação de anúncios de livros. Esses anúncios podem ser acompanhados de uma foto do livro anunciado. No entanto, uma imagem pode exigir diversos megabytes para o seu armazenamento. Assim, ao longo do uso da aplicação, o armazenamento de algumas centenas de imagens se tornaria muito custoso. Para lidar com essa questão, poderia-se limitar o tamanho da imagem a ser enviada. Contudo, caso o limite fosse ultrapassado, o usuário teria de reduzir a resolução de sua câmera fotográfica ou fazer a compressão da foto.

Com o intuito tornar a aplicação mais simples, e tirar a responsabilidade sobre o manejamento de imagens do usuário, foi implementada uma função o redimensionamento de imagens no Google Cloud Functions.

Através da função implementada, o usuário pode fazer o upload de imagens com vários megabytes, sendo o sistema responsável por redimensioná-las, reduzindo assim o seu tamanho.

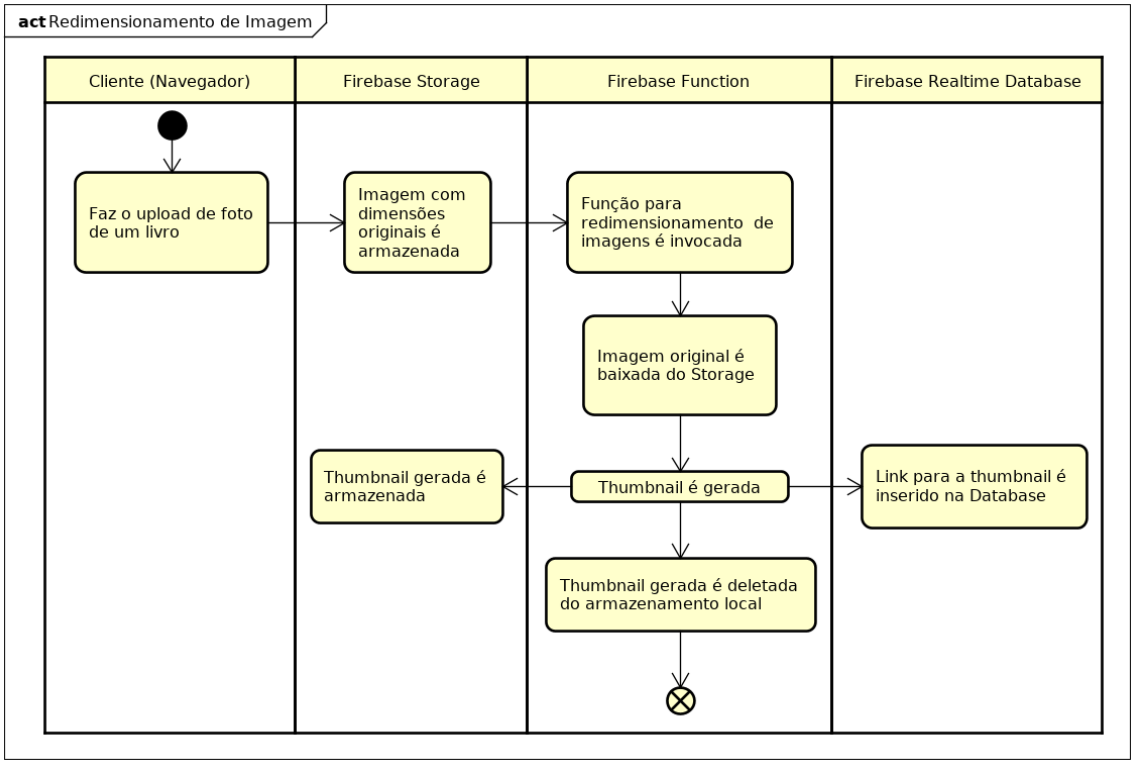
O redimensionamento e conseguinte redução de tamanho das imagens enviadas pelos usuários reduzirá significativamente o tempo de carregamento do site, pois em sua página inicial, por exemplo, serão exibidos diversos anúncios e suas respectivas imagens. Se, por exemplo, fossem exibidos 10 anúncios e cada imagem tivesse um megabyte, seriam dez megabytes para serem carregados, o que prejudicaria bastante a experiência de usuário.

A função para redimensionamento de imagens implementada no Google Cloud Functions foi escrita em JavaScript e será executada em nuvem por um servidor NodeJS. Essa função é invocada sempre que uma imagem for adicionada ao Firebase Storage, e em seguida, a imagem será baixada pelo servidor do Google Cloud Functions, e redimensionada. A foto com novas dimensões será enviada novamente para o Firebase Storage. Ao fim, as urls da

imagem original e da imagem redimensionada serão salvas no Firebase Realtime Database.

No figura abaixo (Figura 24), é apresentado um fluxograma do processo de armazenamento e redimensionamento de imagem implementado

Figura 24: Fluxograma do tratamento de imagens através da infraestrutura do Firebase



em nuvem.

Fonte: Elaborado pelo Autor

De acordo com a figura 24, uma imagem é enviada pelo usuário através de seu navegador web, e por meio do front-end, essa imagem é remetida para o serviço de Storage. Esse evento ativa a Função em Nuvem (Cloud Function), que por sua vez, faz o download da imagem do Storage, a redimensiona e salva a nova imagem de volta ao Storage. Em seguida a imagem é removida do armazenamento da Função em Nuvem e é inserido o link na Realtime Database para a imagem redimensionada salva no Storage.

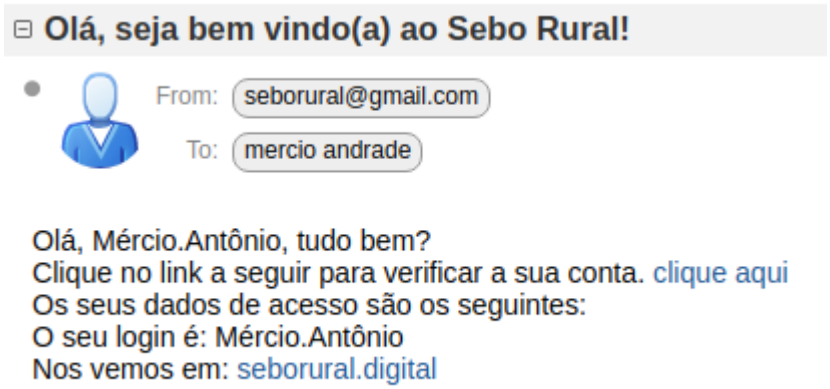
No momento da criação de um anúncio, será enviada para o back-end e persistido no base de dados relacional apenas o link da imagem redimensionada que foi salvo na Firebase Realtime Database.

5.8 VERIFICAÇÃO DE E-MAIL INSTITUCIONAL

Para ter acesso a determinados endpoinsts da API RESTful implementada é necessário possuir um cadastro no sistema e estar logado no mesmo. E para cadastrar-se, por sua vez, é preciso inserir um e-mail institucional da UFRPE na requisição de cadastro.

Em seguida a verificação da presença de “@ufrpe.br” no e-mail informado no momento do cadastro, também é realizada a confirmação de que usuário possui a acesso ao e-mail. Para isso é montado dinamicamente no servidor e enviado um e-mail para o usuário através da API do Sendgrid[11]. O formado deste e-mail é apresentado na figura 27 a seguir.

Figura 25: Verificação de email institucional da UFRPE



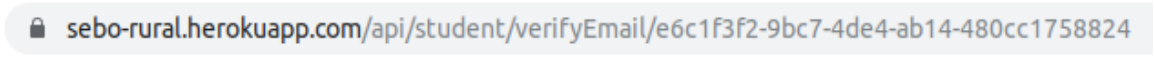
Fonte: Elaborado pelo autor.

O nome de usuário recebido pelo servidor por meio de uma requisição HTTP POST é utilizado para a geração do e-mail de verificação. A chave de verificação é então remetida para o servidor mediante uma requisição

[11] https://www.sendgrid.com

HTTP GET, que é executada pelo usuário ao clicar no link gerado e enviado para o mesmo por e-mail. O link gerado pelo servidor tem o seguinte formato:

Figura 26: Link gerado no back-end para verificação de email do usuário



Fonte: Elaborado pelo autor.

O link exibido na figura 28 acima aciona a API RESTful mediante uma requisição HTTP GET. Ela chama o método verifyEmail do controlador Student e passa como parâmetro a chave de verificação, a qual no exemplo acima trata-se de : e6c1f3f2-9bc7-4de4-ab14-480cc1758824. Para a validação da chave, o método verifyEmail faz uma busca na base de dados pela chave de verificação. Caso ela seja encontrada, o atributo isEmailVerified do objeto student correspondente é modificado para verdadeiro, possibilitando que ele possa logar-se e ter acesso a endpoints restritos. Ao fim da execução da função verifyEmail, é exposto na tela do usuário a mensagem: “email verificado” ou “código inválido”.

6. Telas da aplicação

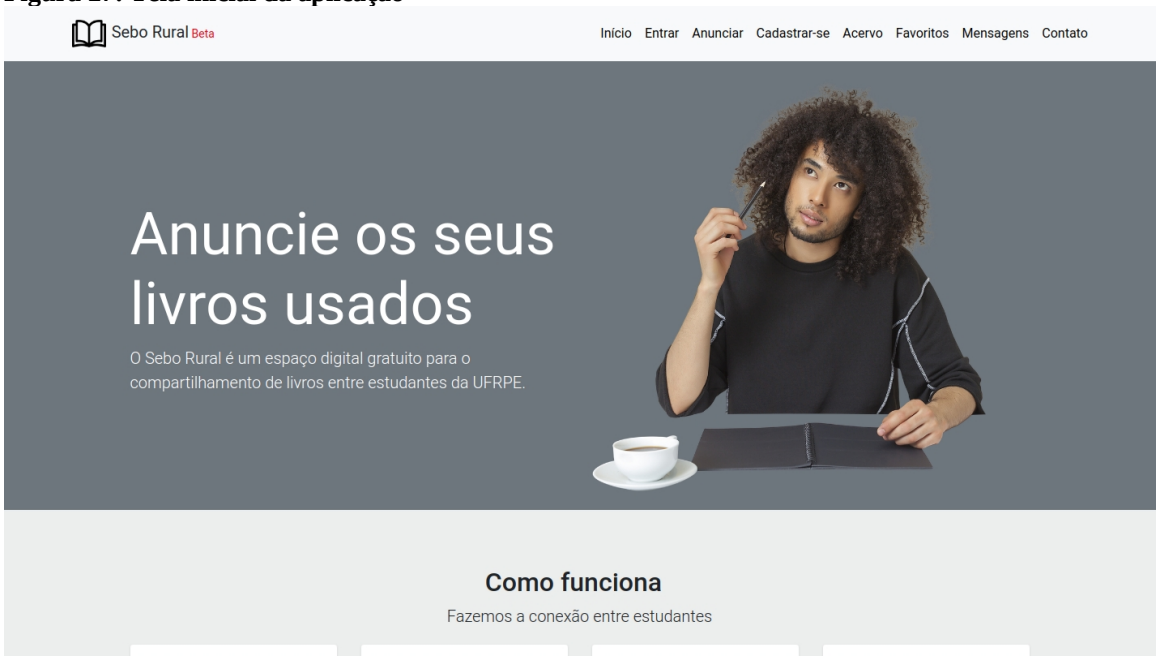
6.1 TELA INICIAL DA APLICAÇÃO

Na figura 25 abaixo pode-se observar a tela inicial da aplicação. Por ser um software em Angular, cada uma das telas é um elemento independente, e possui suas próprias variáveis.

As principais funcionalidades estão dispostas em um menu no canto superior direito da tela. São elas: Entrar, Cadastrar-se, Acervo, Favorito, Mensagens e Contato. Cada um desses itens, ao serem clicados, carregam um componente Angular correspondente. Os mesmos são carregados no centro da tela, sendo mantidos o mesmo cabeçalho e rodapé. Esta é uma característica inerente a uma aplicação de página única.

Os itens do menu Entrar e Cadastrar-se levam, respectivamente, as telas de login e de cadastro de usuário. O item Acervo exibe todos os anúncios de livros cadastrados e permite a navegação pelos mesmos de acordo com o curso universitário desejado. O elemento Favorito apresenta para o usuário os anúncios marcados como favoritos pelo mesmo e o item Mensagem exibe as mensagens enviadas para o usuário.

Figura 27: Tela inicial da aplicação



Fonte: Elaborado pelo autor.

6.2 TELA E PROCESSO DE LOGIN

Na figura 26 abaixo, está a tela de cadastro da aplicação, a mesma é carregada ao ser clicado o item Cadastro no menu da tela inicial. As informações solicitadas do usuários são: O nome, o e-mail institucional da UFRPE, o curso que o mesmo está matriculado e uma senha criada por ele.

Os dados informados pelo usuário passam por um tratamento no próprio software cliente, ou seja, no front-end. O nome e sobrenome inseridos pelo usuário, serão concatenados para montar o seu “username”, por exemplo, caso seja inserido no campo nome: Mércio Andrade, o primeiro e segundo nomes serão identificados, concatenados e será, então, enviada a string “Mércio.Andrade” para para o servidor. Esse dado será salvo na propriedade username. Como será enviado como texto no corpo de uma requisição, através do método HTTP POST, o mesmo permite o uso de acentos e caracteres especiais.

Para o campo e-mail, é feita uma verificação na qual exige-se que o mesmo termine com ufrpe.br. Caso falhe será exibida uma mensagem informando da necessidade de possuir um e-mail institucional da UFRPE. Já no campo de seleção do curso, são solicitados ao servidor os cursos cadastrados através de uma requisição HTTP GET, e os mesmos são exibidos numa tag HTML select para que o usuário possa selecionar o seu curso. No campo senha, poderá ser inserida uma senha qualquer criada pelo usuários.

Após a submissão dos dados, que é feita ao clique do botão Enviar, o front-end realizará uma requisição HTTP POST. Todos os dados informados estarão contidos no corpo da requisição como um objeto JSON. Ao receber os dados o back-end criará um chave de verificação e enviará um e-mail para o usuário informando os seus respectivos dados dados de login e senha, além de um link para a verificação do e-mail.

Figura 28: Tela de cadastro da aplicação proposta

Fonte: Elaborado pelo autor.

6.3 TELA E PROCESSO DE CRIAÇÃO DE ANÚNCIO

Para anunciar um livro, o usuário deve clicar no item de menu Anunciar e preencher o seguinte formulário (Figura 29). Neste formulário serão solicitados os seguintes dados relacionados ao livro a ser anunciado: o título, o nome do autor, o preço para revenda, o curso e o período ao qual no qual o mesmo é utilizado e uma breve descrição acerca do livro, além de uma foto do mesmo.

O upload da foto enviada pelo usuário é feito por meio do front-end, ou seja, por meio do framework Angular. A foto é enviada para o Firebase Storage. Um thumbnail da mesma é gerada por uma função no Google Cloud Functions. Essa função envia, então, o link da thumbnail para o Firebase Realtime Database e também de volta para o front-end. O front-end então recebe esse link e o envia para ser salvo no back-end.

Os dados fornecidos pelo usuário serão enviados para o servidor por meio de uma requisição HTTP POST. Esses dados estarão no cabeçalho da requisição como atributos de um objeto JSON. Eles estão associados a entidade Book e serão recebidos pelo back-end e persistidos na tabela book.

Figura 29: Tela de criação de anúncio

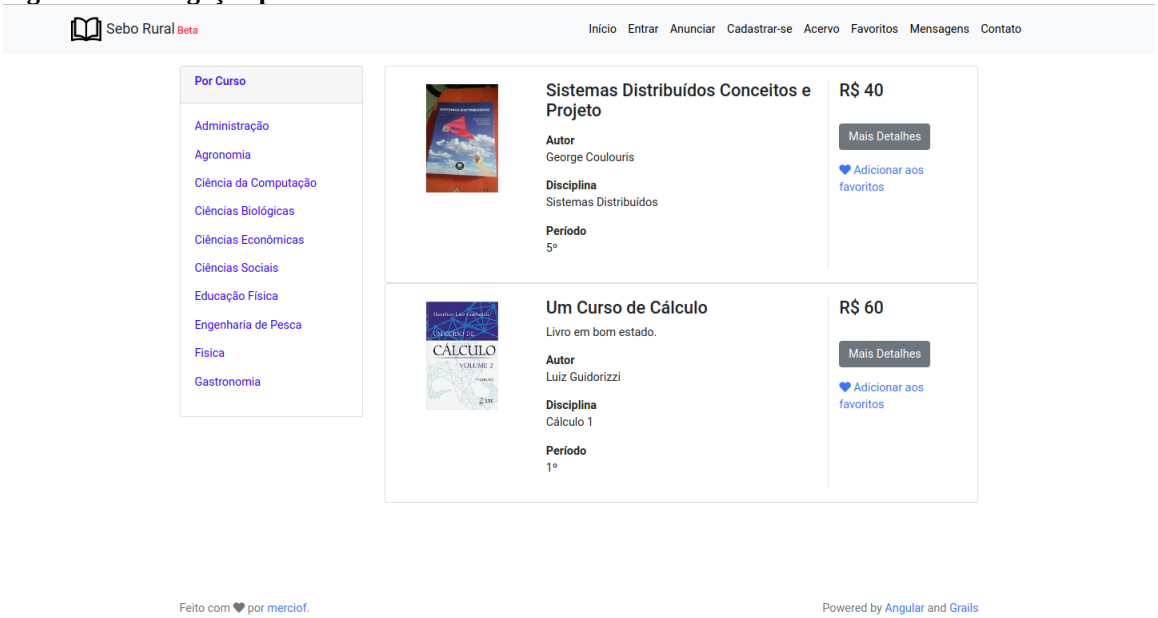
Fonte: Elaborado pelo autor.

6.4 TELA DE NAVEGAÇÃO PELO ACERVO

A navegação pelo acervo de livros anunciados poderá ser feita pela ordem cronológica na qual os anúncios foram criados. Após clicar no menu Acervo, o usuário terá exibido em sua tela, uma listagem com os anúncios mais recentes.

Um dos requisitos desta aplicação proposta, e um diferencial em relação as aplicações preexistentes analisadas, é a navegação pelos anúncios a partir de cursos da universidade, especificamente da UFRPE. Como pode ser visto na figura 30, há uma barra lateral, na qual os cursos cadastrados são trazidos do servidor, por meio da API RESTful e exibidos em ordem alfabética na tela. O usuário pode então clicar no curso desejado. Dessa forma, após uma nova consulta na base de dados, solicitada pelo front-end através da API, serão exibidos os cursos que possuam o id de um determinado curso como chave estrangeira.

Figura 30: Navegação pelo acervo de anúncios



Fonte: Elaborado pelo autor.

6.5 TELA DE EXIBIÇÃO DETALHADA DE UM ANÚNCIO

A visão detalhada de um anúncio, a qual será exibida a partir do clique no item Mais Detalhes da barra de navegação, trará os seguintes dados: Autor, Curso, Disciplina, Período e o nome do Estudante responsável pela criação do anúncio em questão (Figura 31). Desta forma, o usuário saberá qual o curso, a disciplina e o período aos quais determinado livro está relacionado. No futuro, poderão ser implementadas funcionalidades como navegação por disciplinas e períodos.

A foto exposta na tela de exibição detalhada de anúncio, possui dimensões um pouco maiores que a foto exibida na listagem geral de anúncios, no entanto, se trata da mesma thumbnail armazenada no Firebase Storage.

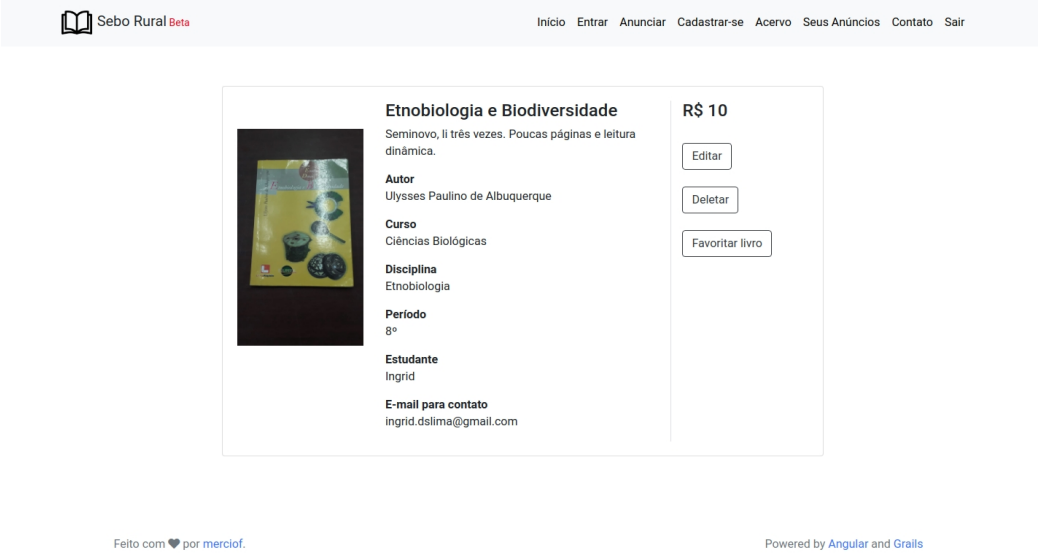
Figura 31: Exibição detalhada de um anúncio

Fonte: Elaborado pelo autor.

6.6 TELA DE LISTAGEM DE ANÚNCIOS DE UM USUÁRIO ESPECÍFICO

O usuário do sistema poderá ver os seus próprios anúncios de forma exclusiva, ou seja, a parte dos demais anúncios. De acordo com o usuário que estiver logado, o sistema exibirá os anúncios feitos por esse determinado usuário (Figura 32). Essa funcionalidade poderá ser acessada através do clique em Seus Anúncios na barra de navegação. Assim, o usuário terá acesso aos seus próprios anúncios de forma simples, podendo editá-los ou deletá-los.

Figura 32: Tela de listagem de anúncio de um usuário específico



Fonte: Elaborado pelo autor.

7. Telas da aplicação em dispositivos móveis

7.1 TELA DE EDIÇÃO DE UM ANÚNCIO

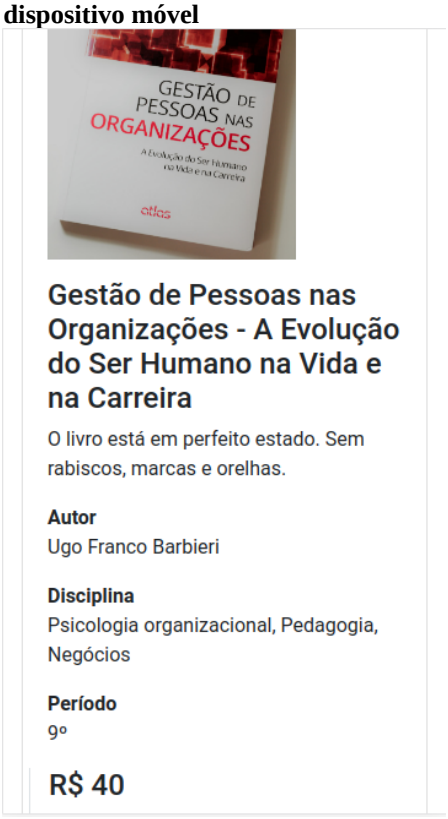
Na figura 33, está a tela de edição de um anúncio. Nela os usuários poderão editar os seus próprios anúncios. A edição ocorre através de uma requisição HTTP PUT para a URI api/book/update. Os dados são enviados no cabeçalho da requisição.

Figura 33: Tela de edição de um anúncio

7.2 TELA MOBILE DE NAVEGAÇÃO PELO ACERVO

Todas as telas da aplicação implementada se adaptam à dispositivos móveis. Na figura 34 abaixo, está um recorte de tela de um smartphone durante a navegação no acervo de anúncios. O front-end também identifica quando é acessado por um dispositivo móvel e exibe mensagens específicas aos usuários destes dispositivos para facilitar a sua navegação na aplicação.

Figura 34: Navegação pelo acervo em dispositivo móvel



Fonte: Elaborado pelo autor.

7.3 TELA PRINCIPAL EM DISPOSITIVOS MÓVEIS

A visualização da tela principal em um dispositivo móvel pode ser vista na figura 35. O conteúdo se adéqua as dimensões da tela, sendo carregado e empilhado verticalmente.

Ao clicar no ícone de “hamburguer”, no canto superior direito (Figura 36), o menu de navegação será carregado. Desse modo, o usuário poderá navegar pelo menu de forma apropriada ao seu dispositivo.

Figura 35: Tela principal em dispositivo móvel

Figura 36: Menu em dispositivo móvel

7.4 NAVEGAÇÃO PELO ACERVO EM DISPOSITIVO MÓVEL

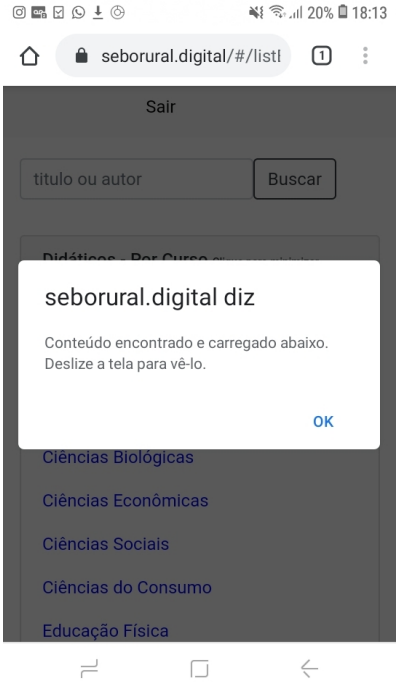
Na figura 37 abaixo, pode-se observar a barra de seleção de cursos em um dispositivo móvel. A listagem dos livros é movida para baixo da barra de seleção, adequando-se, assim, a tela do dispositivo.

A mensagem apresentada na figura 38 é exibida apenas durante a navegação por meio de dispositivos móveis. Ao ao clicar no curso desejado, caso hajam livros cadastrados, o conteúdo é carregado abaixo da barra lateral. Tendo o usuário que deslizar a tela para vê-lo. Assim, ele é informado caso haja ou não conteúdo.

Numa tela de dimensões maiores o conteúdo buscado é carregado ao lado da barra de seleção. Assim, o usuário percebe facilmente o carregamento ou não do conteúdo buscado, não precisando de mensagem para auxiliá-lo.

Figura 38: Barra lateral em dispositivo móvel

Figura 37: Mensagem específicas para navegação dispositivos móveis



8. Avaliação da Aplicação Proposta

Com o intuito de avaliar se a aplicação proposta atende ou não a necessidade de seus usuários, um questionário foi elaborado e aplicado através do Google Forms. O formulário foi divulgado em um grupo do Facebook que reúne alunos e ex-alunos da UFRPE, em pequenos cartazes colados em murais de divulgação da UFRPE e também através do aplicativo de mensagens Whatsapp. Foram obtidas ao total respostas de 33 pessoas. O modelo de avaliação utilizado na pesquisa foi baseado em questionário aplicado por LUPCHINSKI (2015).

O questionário solicita ao participante que se cadastre no sistema, valide o seu e-mail e logue na aplicação, faça o anúncio de um ou mais livros e posteriormente navegue pelo acervo de livros.

Inicialmente, com o propósito de conhecer um pouco sobre o perfil do participante, foi perguntado ao mesmo se ele teria interesse no compartilhamento de livros didáticos. Pode-se observar na figura 35 que 97% deles responderam que sim. Apenas um participante negou possuir interesse nesse tipo de compartilhamento.

Figura 39: Interesse no compartilhamento de livros didáticos

Fonte: Elaborado pelo autor.

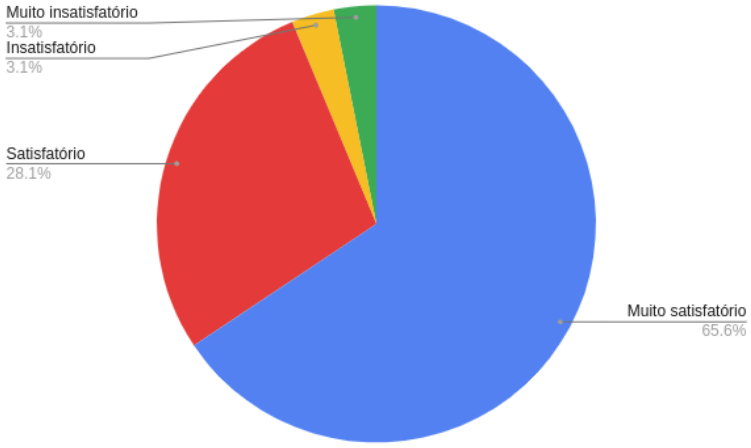
Em seguida, foi perguntado por qual dispositivo o participante estava acessando o formulário e a aplicação. Como pode-se observar na figura 35, aproximadamente 70% afirmaram que o dispositivo que estavam utilizando era um celular. Isso evidencia e fundamenta a importância da responsividade da aplicação, ou seja, que a mesma se adapte a diferentes tamanhos de telas. Todas as telas da aplicação proposta são responsivas. Por outro lado, nenhum dos participantes utilizou um desktop para acesso ao formulário.

Figura 40: Dispositivo utilizado para acesso

Fonte: Elaborado pelo autor.

A primeira tarefa solicitada no questionário, consistia na realização de um cadastro na aplicação. As respostas disponíveis para o usuário, após a criação de seu cadastro, buscavam conhecer o nível de satisfação do participante com o procedimento executado. (Figura 37).

Figura 41: Nível de satisfação com o procedimento de cadastro



Fonte: Elaborado pelo autor.

Após a avaliação do nível de satisfação com o procedimento de cadastro, foi indagado se, caso o usuário tivesse alguma dificuldade, ele a descrevesse textualmente.

Algumas dificuldades e sugestões relatadas foram:

- Quatro pessoas sugeriram que seria mais simples que o login fosse o próprio e-mail. Pois o sistema exige que o usuário insira o seu nome e sobrenome e para gerar o login no modelo “nome.sobrenome”.

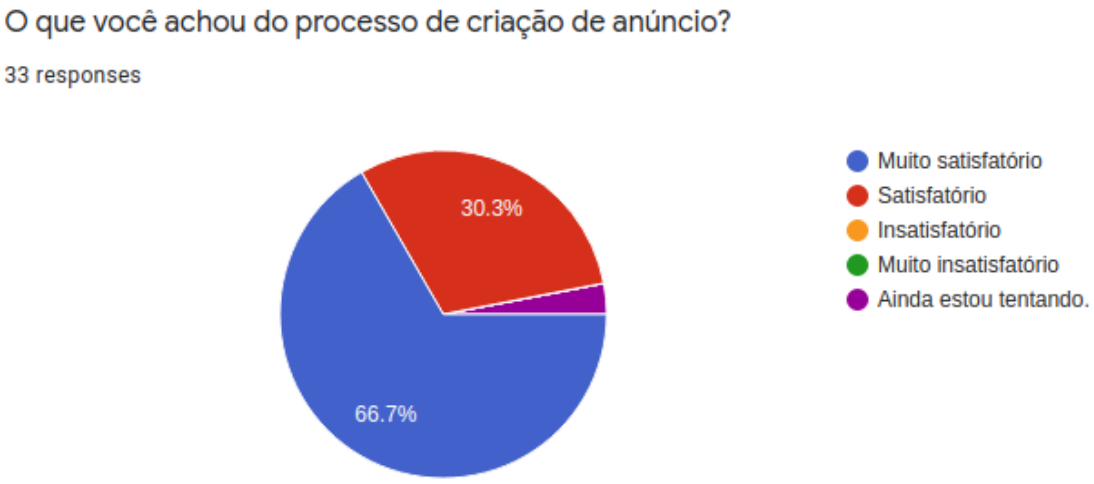
- Duas pessoas não encontraram o seu curso cadastrado no sistema.

- Uma pessoa demorou pra encontrar o e-mail de confirmação pois o mesmo estava na caixa de spam.

Em seguida foi perguntado sobre o nível de satisfação sobre o processo de criação de um anúncio (Figura 38), o qual só acontece após o login, e envolve o upload de uma foto e informações específicas sobre o livro a ser anunciado. De acordo com a figura 38, vinte e duas pessoas consideraram o

processo muito satisfatório, 10 consideraram o processo satisfatório e uma pessoa afirmou não ter conseguido realizar o anúncio.

Figura 42: Nível de satisfação em processo de criação de anúncio



Fonte: Elaborado pelo autor.

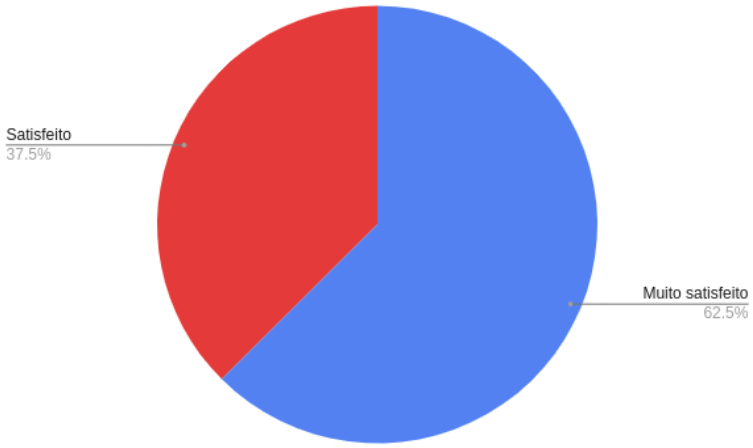
Logo após foi questionado sobre dificuldades encontradas e sugestões para o procedimento de criação de um anúncio. Uma das sugestões foi a substituição do texto “Anunciar” da barra de navegação por uma imagem, pois seria mais visível. Outras duas pessoas sugeriram uma funcionalidade que permita a edição da imagem a ser enviada para o sistema.

Seguidamente, foi perguntado a opinião dos participantes sobre o estilo dos anúncios e das informações que ele fornece. Nesta seção, foi sugerido por participantes a adição de informação sobre a edição e ano de publicação do livro. Houve uma sugestão sobre a adição de uma opção “todos” para disciplina e período. Assim, seria possível o cadastro de um determinado livro relacionado a qualquer período e disciplina como, por exemplo, livros de auto ajuda ou biografias.

A pergunta subsequente tratou do índice de satisfação dos participantes sobre a navegação pelo acervo anúncios (Figura 39). Vinte dos entrevistados afirmaram estarem muito satisfeitos e 12 entrevistados afirmaram

estarem satisfeitos. Para esta funcionalidade houve a sugestão de reduzir o tamanho de cada anúncio para que mais anúncios pudessem ser visualizados na tela. Também foi sugerida a adição de navegação por períodos de um determinado curso.

Figura 43: Opinião sobre a navegação pelo acervo de anúncios



Por fim, foi pedido dos participantes propostas gerais sobre a adição de novas funcionalidades para a aplicação. Algumas das propostas foram:

- Inclusão de mais de uma disciplina e mais de um período para o mesmo livro durante a criação do anúncio do mesmo, pois um determinado livro pode ser útil para várias disciplinas e períodos.

- Criação de uma via de comunicação entre usuários dentro da própria aplicação.

- Inclusão de informações sobre a forma correta de tirar a foto para o anúncio. Garantindo e qualidade e conseguinte legibilidade da mesma.

Abaixo está uma das avaliações:

“Bem, gostei muito da experiência que tive, além de ser uma forma nova de conseguir os livros que precisamos e ajudar também a quem precisa! Muito bom!”

A partir dessa avaliação exercida por possíveis usuários, foi possível incrementar a aplicação de diversas formas. Desde a melhoria de pequenas questões antes não percebidas pelo desenvolvedor até mesmo o planejamento de novas funcionalidades para o sistema.

9. Conclusão

Este trabalho tratou do projeto e do desenvolvimento de uma aplicação baseada na web desde a sua interface com o usuário até a persistência de dados e posterior implantação dela em nuvem. A aplicação desenvolvida tem o objetivo de suprir lacunas no contexto de compartilhamento de livros didáticos online entre estudantes universitários. Para ampliar o conhecimento sobre o tema e fundamentar tanto a proposta quanto a aplicação desenvolvida foram aplicados questionários. Em geral, o retorno foi positivo, cerca de quarenta livros foram cadastrados, e intenciona-se que a aplicação desenvolvida fique como legado para a comunidade.

Conjuntamente, foi intenção deste trabalho o emprego de tecnologias e serviços atuais, como o Angular 8, o Grails 3 e a Google Firebase. Por meio do Github Student Pack, foram usados planos gratuitos para a implantação e manutenção da aplicação em nuvem, bem como para a criação do domínio escolhido. Dessa forma, este trabalho também consistiu em uma oportunidade de participar do processo completo de projeto, desenvolvimento, implantação e avaliação de uma aplicação sem necessidade de nenhum aporte financeiro.

Foram utilizados conceitos e experiências aprendidos essencialmente nas disciplinas de Desenvolvimento de Aplicações para Web, Engenharia de Software, Metodologias Ágeis de Desenvolvimento de Software, Análise e Projeto de Sistemas Orientados a Objetos e Projeto de Desenvolvimento de Software. Essas disciplinas contribuíram como alicerce de conhecimento para o desenvolvimento deste projeto.

Uma das principais lições aprendidas foi a importância do feedback de usuários. A partir deles diversas pequenas questões foram aprimoradas. A maioria delas, fluxos que fugiam do “caminho feliz” e que acabam passando despercebidos do programador. Além disso, foi possível a percepção de inúmeras funcionalidades que poderiam ser adicionadas ao sistema.

Diante da possibilidade de manter o sistema em nuvem gratuitamente, pretende-se o incremento do mesmo com a implantação de funcionalidades

propostas por participantes do questionário de avaliação e a manutenção de sua divulgação por meio de redes sociais e pequenos cartazes.

Como trabalhos futuros poderá ser implementado “Full-text search”, o que enriquecerá muito os resultados de buscas a partir de texto. Também poderá ser adicionado um sistema de recomendação que sugira livros para o usuário com base na atividade dele no site e ser feita leitura de informações diretamente a partir de foto de livros, de modo que a partir da foto enviada pelo o usuário o sistema extraia informações automaticamente sobre o livro a ser anunciado.

10. Referências

BIH-HWANG, Lee; DEWI, Ervin Kusuma; WAJDI, Muhammad Farid, Data security in cloud computing using AES under HEROKU cloud, *in*: **2018 27th Wireless and Optical Communication Conference (WOCC)**, Hualien: IEEE, 2018, p. 1–5.

BOOTSTRAP. MARK, Otto; THORNTON, Jacob. **Bootstrap**. https://getbootstrap.com/. Acessado 17 de novembro de 2019.

CHOPADE, Mrs. Rupali M.; DHAVASE, Nikhil S. Agile software development: Positive and negative user stories. *In*: **2017 2nd International Conference for Convergence in Technology (I2CT)**. Mumbai: IEEE, 2017, p. 297–299. Disponível em: <http://ieeexplore.ieee.org/document/8226139/>. Acesso em: 2 nov. 2019.

COHN, Mike. **User Stories and User Story Examples by Mike Cohn**. Mountain Goat Software. Disponível em: <https://www.mountaingoatsoftware.com/agile/user-stories>. Acesso em: 2 nov. 2019.

COSTIM, André Luiz Donatti. **Sistema web para comercialização de peças automotivas usadas**. 2017. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

DALPIAZ, Fabiano; BRINKKEMPER, Sjaak. Agile Requirements Engineering with User Stories. *In*: **2018 IEEE 26th International Requirements Engineering Conference (RE)**. Banff, AB: IEEE, 2018, p. 506–507. Disponível em: <https://ieeexplore.ieee.org/document/8491182/>. Acesso em: 2 nov. 2019.

DEVMEDIA. Curso de Spring Security: Primeiros passos com Spring Security". **DevMedia**, https://www.devmedia.com.br/curso/curso-spring-security/2190. Acessado 17 de novembro de 2019.

DORNELLES, Thiago de Azevedo. **Reconstrução do software AvalWeb usando conceitos de SPA, REST e NoSQL**. 2016.

EXAME. O que esperar do e-commerce na América Latina em 2019. **Dino**. Fonte: https://exame.abril.com.br/negocios/dino/o-que-esperar-do-e-commerce-na-america-latina-em-2019/. Acessado 17 de novembro de 2019.

FIELDING, Roy T.; TAYLOR, Richard N. **Architectural styles and the design of network-based software architectures**. Doctoral dissertation: University of California, Irvine, 2000.

FRANCO, FERNANDO HENRIQUE ALVES. **Ferramenta Didática WEB Colaborativa para Criação e Simulação de Circuitos Lógicos Digitais.** 2016.

GAMMA, Erich (Org.). **Design patterns: elements of reusable object-oriented software**. Reading, Mass: Addison-Wesley, 1995. (Addison-Wesley professional computing series).

GITHUB. **Build Software Better, Together**. Fonte: https://github.com. Acessado 17 de novembro de 2019.

GOETZE, Márcia; THOMÉ, Gladis. Efeito alelopático de extratos de Nicotiana tabacum e Eucalyptus grandis sobre a germinação de três espécies de hortaliças. **Current Agricultural Science and Technology**, v. 10, n. 1, 2004.

IDEC. Pesquisa do Idec constata que escassez de livros nas bibliotecas e legislação dificultam formação do estudante. https://idec.org.br/em-acao/em-foco/pesquisa-do-idec-constata-que-escassez-de-livros-nas-bibliotecas-e-legislacao-dificultam-formacao-do-estudante. **IDEC**. Acessado em 12/07/2018 as 10:54.

JADHAV, Madhuri A.; SAWANT, Balkrishna R.; DESHMUKH, Anushree. Single page application using angularjs. **International Journal of Computer Science and Information Technologies**, v. 6, n. 3, p. 2876-2879, 2015.

JOSEPH, Renien John. Single page application and canvas drawing. **arXiv preprint arXiv:1502.03530**, 2015.

JUDD, Christopher M.; NUSAIRAT, Joseph F.; SHINGLER, James. **Beginning Groovy, Grails and Griffon.** New York: Apress, 2012.

KHEDKAR, Sonam et al. Real Time Databases for Applications. I**nternational Research Journal of Engineering and Technology (IRJET)**, v. 4, n. 06, p. 2078-2082, 2017.

LEE, In, Social media analytics for enterprises: Typology, methods, and processes, **Business Horizons**, v. 61, n. 2, p. 199–210, 2018.

LIMA, Priscila Batista et al. **Desenvolvimento de uma Aplicação Web para Colônia de Pescadores Utilizando os Frameworks AngularJS e Node**. js. 2016.

LIMA, Priscila Batista et al. **Desenvolvimento de uma Aplicação Web para Colônia de Pescadores Utilizando os Frameworks AngularJS e Node. js.** 2016.

LUMMERTZ, Ramon Santos; SGANZERLA, Antoni. Direto ao Ponto–App colaborativo do transporte coletivo usando o Firebase. **Conversas Interdisciplinares**, v. 14, n. 1, 2018.

LUPCHINSKI, Raphael de Leon Ferreira. **Desenvolvimento de uma aplicação de página-única e banco de dados não-relacional para organização e controle de eventos esportivos.** 2015.

MACHADO, Nilson José. Sobre livros didáticos: quatro pontos. **Em aberto**, v. 16, n. 69, 2008.

MACHADO, Rafael Silveira. **Análise e implementação de aplicação web para programação musical da rádio da universidade.** 2014.

MALONE, K., & GRIFFITH, J. (2012). A case study in the use of Groovy and Grails. Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12.

MARTINS, S. G. et al. Fator cobertura e manejo do solo e perdas de solo e água em cultivo de eucalipto e em Mata Atlântica nos Tabuleiros Costeiros do estado do Espírito Santo. **Scientia Forestalis**, v. 38, n. 87, p. 517-526, 2010.

MORON-RODRIGUEZ, Luis; LONGA-CHEVARRIA, Bryan; SHIGUIHARA-JUAREZ, Pedro, Analysis of image transfer mechanisms in a RESTful API client-server architecture and its application to lane detection, *in*: **2017 IEEE International Conference on Aerospace and Signals (INCAS)**, Lima: IEEE, 2017, p. 1–4.

NUGROHO, Lukito Edi *et al*, Development of RESTful API to support the oil palm plantation monitoring system, *in*: **2017 7th International Annual Engineering Seminar (InAES)**, Yogyakarta, Indonesia: IEEE, 2017, p. 1–5.

RASMO, Garcia et al. Fatores limitantes ao crescimento do capim-tanzânia em um sistema agrossilvipastoril com eucalipto, na região dos cerrados de Minas Gerais. **Revista Brasileira de Zootecnia**, v. 30, n. 4, p. 1178-1185, 2001.

SANTOS, Claudia Santana e SILVA, José Luís Caetano da. Os Impactos do Plantio de Eucalipto e da Produção de Celulose em Comunidades Tradicionais no Extremo Sul Baiano. ANPPAS. **Anais do II Encontro Associação Nacional de Pós Graduação**. Novembro de 2002.

SHANE-SIMPSON, Christina *et al*, Why do college students prefer Facebook, Twitter, or Instagram? Site affordances, tensions between privacy and self-expression, and implications for social capital, **Computers in Human Behavior**, v. 86, p. 276–288, 2018.

SILVA, Matheus Rodrigues Rosado da. **Projeto e desenvolvimento de um sistema para gerenciamento de trabalhos de conclusão de curso.** 2017.

SIMILARWEB. Similarweb.Com - Digital World Market Intelligence Platform. **SimilarWeb.Com**, https://similarweb.com/. Acessado 17 de novembro de 2019.

SINDICATO NACIONAL DAS EDITORAS DE LIVROS. Divulgado o resultado da Pesquisa Produção e Vendas do Setor Editorial Brasileiro ano-base 2017; confira a íntegra. Fonte: http://www.snel.org.br/apresentado-o-resultado-da-pesquisa-producao-e-vendas-do-setor-editorial-brasileiro-ano-base-2017/. **SNEL**. Acessado em 12/07/2018 às 07:39.

SINGH, Aditya et al. Formulating an MVC Framework for Web Development in JAVA. In: **2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)**. IEEE, 2018. p. 926-929.

ULGUIM, Yulle Pereira et al. **Rede social para pesquisa e criação de projetos colaborativos.** 2017.

VAN OEL, P. R.; HOEKSTRA, A. Y. **The green and blue water footprint of paper products**: methodological consideration and qualification. Delft, The Netherlands: UNESCO-IHE, 2010.

WEISSMANN, H. L. **Falando de Grails**: São Paulo: Casa do Código, 2014.

Apêndice A

A seguir está o questionário aplicado a estudantes universitários com o intuito de conhecer mais sobre o contexto e fundamentar a proposta do trabalho:

Título: Compartilhamento e Reuso de Livros Universitários

subtítulo: Participe da construção de uma aplicação gratuita que facilitará a vida do estudante universitário e atuará na conservação do meio ambiente.

Pergunta 1: Qual a sua instituição de ensino?

Pergunta 2: Qual o seu curso?

Pergunta 3: Qual o seu período?

Pergunta 4: Você tem interesse na compra, venda, troca ou doação de livros didáticos usados?

Alternativas da pergunta 4: Sim ou não.

Pergunta 5: Qual ferramenta(s) você utiliza ou utilizaria para esse fim (compra, venda, troca ou doação de livros universitários)?

Alternativas da pergunta 5: Facebook, Mercado Livre, OLX, Outro

Pergunta 6: Quais as principais limitações ou dificuldades encontradas nessas ferramentas?

Pergunta 7: Você usaria uma aplicação específica para o anúncio de livros usados entre estudantes de sua universidade?

Alternativas da pergunta 7: sim ou não.

Apêndice B

Para a divulgação do site, foi criado um anúncio patrocinado no Facebook. O anúncio pode ser visto na figura 41 a seguir. Ele foi direcionado especificamente para residentes da Região Metropolitana do Recife que sejam membros ou ex membros da UFRPE.

Figura 44: Publicidade no Facebook

Fonte: Elaborado pelo autor.

O resultado do anúncio está apresentado na figura 42 a seguir. Foram alcançadas 2.233 pessoas, das quais 54 clicaram no anúncio. O Facebook forneceu 30 reais de crédito de forma promocional.

Figura 45: Resultado de publicidade no Facebook

Fonte: Elaborado pelo autor.

Como uma outra forma de divulgação da aplicação, foram colados cerca de 20 pequenos cartazes em diversos murais espalhados pela Universidade. Os cartazes podem ser vistos nas figuras 43 e 44 abaixo. Nos pequenos papéis destacáveis estava escrito: “Anuncie seus livros usados! Acesse: https://seborural.digital”.

Figura 46: Cartaz 1

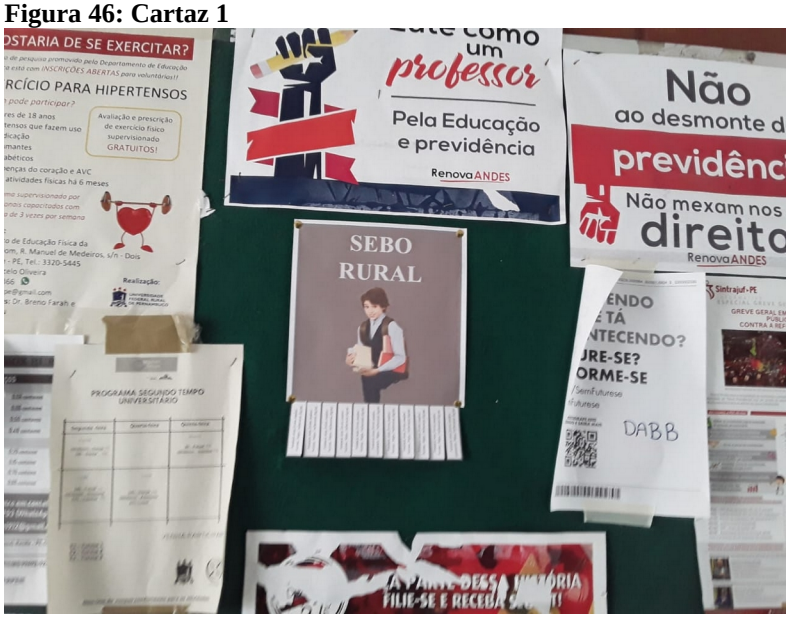


Figura 47: Cartaz 2

Como resultado da divulgação, foram realizados cerca de 90 cadastros no site do Sebo Rural. (Figura 45).

Figura 48: Cadastros realizados

Data Output | Explain | Messages | History

	id bigint	version bigint	password_expired boolean	username character varying(255)	account_locked boolean	password character varying(255)	account_expired boolean	enabled boolean	class character varying(255)
86	300	1	f	HEITOR.FEITOSA	f	$2a$10$1X7vojhCAKh3/4/2l2N15.wXpDFPLm0agzOZ026jbAcQofcy.yNTa	f	t	sebo.rural.deploy.Stu
87	413	1	f	Luan.Martins	f	$2a$10$uxWiSNSLuZtmPrSoeIrXle72KOBuZLcP6zVW7qBGnY9z4GFB8pz1m	f	t	sebo.rural.deploy.Stu
88	414	0	f	Bruno.luz	f	$2a$10$Vxbmagnppnzure48m9QdaeDsATTCM3pdjDMeqrrlzWNO99jqx1uzC	f	t	sebo.rural.deploy.Stu
89	415	2	f	Marcos.Cardoso	f	$2a$10$kzh7q7e3T99DYJmxh.NJU.I3CPRTO3cmUaRB1drmoiQBBoNAXOY.K	f	t	sebo.rural.deploy.Stu
90	416	1	f	Brenda.Oliveira	f	$2a$10$2mvQqw9SNHe9E6BudVJYde3jEphZJvEDqTR2Afn0dhzdqzO.YiS2G	f	t	sebo.rural.deploy.Stu
91	427	1	f	Gabriel.Santos	f	$2a$10$kydy1.FlUL8nMqUPmCBqnOds0GFy0hWEXKwi5h14uDI0UwRJd/PNS	f	t	sebo.rural.deploy.Stu
92	429	1	f	Rayanne.Thalita	f	$2a$10$GLdBE4nbRyRW/dRFRmuMZefdandojjjwZkBLTvA6ZFrBTKK/KaZ9S	f	t	sebo.rural.deploy.Stu
93	430	0	f	Helen.Lylyan	f	$2a$10$F35nHDFDyRRlfOAqpiZ9w.1MeLHAzZ1/55k3FEcGysikMbdj6VjRm	f	t	sebo.rural.deploy.Stu
94	428	2	f	Juliana.	f	$2a$10$sa.3e9zn7o/UeAlORe0GiuWyQf.CIIGP2IvS1b55N390x5LVTWRJu	f	t	sebo.rural.deploy.Stu
95	431	1	f	Rafaela.	f	$2a$10$CEXVksn61a2mgnorWNjMsOB06ivBcNlqQQq0ws0fOuhcPTeQ/A/nm	f	t	sebo.rural.deploy.Stu
96	443	1	f	Maria.silva	f	$2a$10$uyD9SZ1blexiA3Uoj9aesuaF3RhKptzLlI/sLcyI7gS2NjxKrLZlm	f	t	sebo.rural.deploy.Stu
97	462	1	f	Leonardo.ricardo	f	$2a$10$x0jUOSoCrmEfijBHX1zsyu70UAPzc4CBNuyS//.Jffkelzs2KiyFK	f	t	sebo.rural.deploy.Stu
98	463	0	f	José.Gouveia	f	$2a$10$RQhlfzsFByqSZSLWvhq7WOy/DBzFwgqXlp9GDwxnIPuHN/K5var72	f	t	sebo.rural.deploy.Stu
99	464	1	f	JOSÉ.GOUVEIA	f	$2a$10$9hA.QYbDDyE1m6G0dG5WXO0ucteFJ6amauZjbL60lMTzGhCErqozq	f	t	sebo.rural.deploy.Stu
100	465	1	f	Givanildo.Alves	f	$2a$10$ZZv4IoFNRHneML7bxXiuC.XX7IMv2frKkhyDFur17HzE7sxP0.PLW	f	t	sebo.rural.deploy.Stu
101	467	1	f	Gabriela.Regina	f	$2a$10$7FXnNFkVINnQxF5za019BOV9jCh82tiVfsp9kH63STFH2heyp4Nc2	f	t	sebo.rural.deploy.Stu
102	474	0	f	Mércio.Filho	f	$2a$10$z/RR9WIgLZ7UmQwJRK.CBuGm2wlfaxgRUfiG0VYb56NW6FANtNqdq	f	t	sebo.rural.deploy.Stu
103	466	1	f	Franklin.Ferreira	f	$2a$10$aYqwhRSZ/LSoPfKICMh5POzDfmhc9ucfysOH71SBbxnEIT/iNQUNu	f	t	sebo.rural.deploy.Stu
104	475	1	f	Douglas.Silva	f	$2a$10$PnUB8zqxQGE85TsUFl9D2.m66u8hQMlxVHAUs9u6OgLyI0osr6H3m	f	t	sebo.rural.deploy.Stu
105	476	1	f	Maria.Carolinne	f	$2a$10$K9l5vdj/0EKl49CAmH3/BuEAI39LrVd470EFIhXvMHfUE4Jj.rkUi	f	t	sebo.rural.deploy.Stu
106	478	0	f	Jennifer.Santos	f	$2a$10$Otu3jTbxJ7o3yYIK3HaHsus3jku.0sZaLAYJfUeaoxXeEVafZQKuy	f	t	sebo.rural.deploy.Stu
107	479	1	f	ANDREZA.RODRIGUES	f	$2a$10$MCwh8McfyLZJU/Ll7/3j7.YHBk3Ug/cSywkjoy2Dd1I1C2bTpkU5i	f	t	sebo.rural.deploy.Stu
108	480	1	f	Luiza.Lyra	f	$2a$10$fQ9Y3iXV/qvr3z3KR8gzeuoD6DBbk8XJsVpAgqy.jEDnio3ox2ZqS	f	t	sebo.rural.deploy.Stu
109	481	1	f	Pedro.Sena	f	$2a$10$OvwaOFzHrQCypUIVUkikgemAww58ab9RYa/bjWD6BGh6gF5DzDnqK	f	t	sebo.rural.deploy.Stu
110	483	1	f	Isabelle.Araujo	f	$2a$10$196xCkaoNF.Wv8ZTfWLjnOapXzc6URNcUMu73EP3rqH87l2f37DnW	f	t	sebo.rural.deploy.Stu
111	484	2	f	Vitória.Neves	f	$2a$10$jrIovBqlZO6dq7jcb/l8Nesgqyebl32U1zkiBQIRTymhpnjsWxQHS	f	t	sebo.rural.deploy.Stu
112	482	1	f	Thiago.Silva	f	$2a$10$d/r0AD34S5BCgp4NAuq51OnUHWd7ZD30g8q86JcoUmk8BP5fe8PJG	f	t	sebo.rural.deploy.Stu
113	485	1	f	Kerolayne.Emilly	f	$2a$10$Z90R/SPuADLjxxipypKv4.KE45mHi1uFGlDKJmy5rJSKKv0CNcuIC	f	t	sebo.rural.deploy.Stu
114	486	1	f	Kevin.kline	f	$2a$10$uWp6w.eSc6JMzhrcvSg85uLlbjMkjWeNKcUuzKDoEVLlMva8/xzYO	f	t	sebo.rural.deploy.Stu
115	488	1	f	jose.george	f	$2a$10$5EhXA7bKY3fHWnQBH3yQne1HzDQDi0sShTw6jhKCiO3IMJDilP1Ve	f	t	sebo.rural.deploy.Stu

Foram anunciados 28 livros. (Figura 46).

Figura 49: Livros anunciados na aplicação proposta

Data Output | Explain | Messages | History

	id bigint	version bigint	periodo character varying(255)	data_criacao_anuncio timestamp without time zone	titulo character varying(255)	autor character varying(255)
1	58	0	8º		Etnobiologia e Biodiversidade	Ulysses Paulino de Albuquer
2	103	0	2º		Estrutura de Dados em C	Aaron M. Tenenbaum
3	104	0	5º		Sistemas distribuídos	George Coulouris
4	145	0	6º		O golpe de 64 e a ditadura militar	Júlio José Chiavenato
5	146	0	2º		Teoria do Texto	Salvatore D'Onofrio
6	106	0	2º		A Língua Como Instrumento	ROSANA MORAIS WEG
7	107	0	2º		A Língua Como Expressão e Criação	Rosana Morais Weg
8	150	0	8º		Coleção Geografia Conexões - Parte I	Lygia Terra; Regina Araujo;
9	162	0	2º		Genética médica	Thompson & Thompson
10	164	0	1º		Manejo ecológico do solo : A agricultura em regiões tropicais	Ana Maria Primavesi
11	166	0	1º		Modernidade Líquida	Zygmunt Bauman
12	178	0	6º		Oracle SOA Suite 11g Developer's Cookbook	Wright Matt
13	182	0	1º		zoologia dos invertebrados	Ruppert/ barnes
14	183	0	3º		PRINCIPLES OF DEVELOPMENT	Lewis Wolpert
15	187	0	4º		python	Sergio luiz
16	189	0	1º		prisioneiros da mente 2018	augusto cury
17	194	0	5º		Princípios de Estatística em Ecologia	Nicholas J. Gotelli
18	204	0	5º	2019-11-19 23:38:03.373	Inteligência artificial	Faceli
19	265	0	9º	2019-11-22 12:48:36.865	Gestão de Pessoas nas Organizações - A Evolução do Ser Humano na Vida e na Carreira	Ugo Franco Barbieri
20	266	0	1º	2019-11-22 12:54:04.133	Introdução ao Direito Penal, criminologia, princípios e cidadania	Gianpaolo Poggio Smanio e H
21	267	0	1º	2019-11-22 12:58:19.123	Organizações Criminosas Aspectos Penais e Processuais	Eduardo Araújo da Silva
22	264	2	9º	2019-11-22 12:37:23.411	Administração Management, Construindo Vantagem Competitiva	Bateman e Snell

Apêndice C

A seguir serão apresentadas as perguntas utilizadas no questionário de avaliação da aplicação.

Pergunta 1: Você tem interesse no compartilhamento de livros didáticos?
Pergunta 2: Qual é o seu curso?
Pergunta 3: Qual é o seu período?
Pergunta 4: Você está acessando o Sebo Rural por qual dispositivo?
Pergunta 5: O que você achou do processo de cadastro do Sebo Rural?
Pergunta 6: Você teve alguma dificuldade com o seu cadastro? Qual?
Pergunta 7: O que você achou do processo de criação de anúncio?
Pergunta 9: Você teve alguma dificuldade no processo de criação de anúncio? Qual?
Pergunta 10: Quanto aos dados exibidos nos anúncios, sentiu falta de alguma informação?
Pergunta 11: O que você achou da navegação pelo acervo de anúncio?
Pergunta 12: Você teve algum problema com a navegação pelo acervo de anúncios? Qual?
Pergunta 13: Gostaria de sugerir alguma nova funcionalidade para o Sebo Rural?