

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Vieira Mendes

**Desenvolvimento de uma ferramenta para
organização e gerenciamento de atividades de
docentes**

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rafael Vieira Mendes

**Desenvolvimento de uma ferramenta para organização e
gerenciamento de atividades de docentes**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau
de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. André Ricardo Backes

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2018

Rafael Vieira Mendes

Desenvolvimento de uma ferramenta para organização e gerenciamento de atividades de docentes

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Trabalho aprovado. Uberlândia, Brasil, 11 de julho de 2018:

Prof. Dr. André Ricardo Backes
Orientador

Prof. Dr. Bruno Augusto Nassif
Travencolo

Prof. Dr. Mauricio Cunha Escarpinati

Uberlândia, Brasil
2018

Resumo

Esse documento descreve o processo de desenvolvimento de um sistema para gestão das atividades exercidas pelos docentes, que tem o objetivo de facilitar o controle e avaliação da Progressão ou Promoção de Carreira determinada pelo Conselho Diretor (CONDIR) da Universidade Federal de Uberlândia. O sistema foi desenvolvido com a utilização do *framework* de *front-end* Vue.js (linguagem JavaScript), enquanto que o *back-end* foi feito utilizando o *framework* Laravel (linguagem PHP). Como banco de dados, foi escolhido utilizar o banco de dados relacional MySQL .

Palavras-chave: Aplicação Web. PHP. Vue.js. Universidade. Progressão de Carreira.

Lista de ilustrações

Figura 1 – Modelo do relatório de atividades	9
Figura 2 – Exemplo de renderização declarativa do Vue.js	11
Figura 3 – Página inicial do CoPilot	13
Figura 4 – Estrutura de um projeto em Laravel 5 e arquivo de configuração <code>.env</code> .	15
Figura 5 – Modelo Entidade-Relacionamento-Atributo (ERA) da aplicação	19
Figura 6 – Diagrama de casos de uso do sistema	21
Figura 7 – Tela de cadastro de tipos de atividades descritas na resolução do conselho	22
Figura 8 – Janela <i>modal</i> para cadastro ou edição de um tipo de atividade	22
Figura 9 – Tela de cadastro das atividades realizadas pelo docente	23
Figura 10 – Janela <i>modal</i> para cadastro ou edição de uma atividade	23
Figura 11 – Tela para geração de relatórios	24
Figura 12 – Tela de gerenciamento de usuários	24
Figura 13 – Janela <i>modal</i> para edição de um usuário, incluindo atribuição de papéis	25
Figura 14 – Janela <i>modal</i> para importação “massiva” de atividades	28

Sumário

1	INTRODUÇÃO	6
1.1	Objetivos	6
2	DETALHAMENTO DO PROBLEMA	8
2.1	Requisitos Funcionais	8
2.2	Requisitos Não Funcionais	9
3	DESENVOLVIMENTO	11
3.1	Tecnologias Utilizadas	11
3.1.1	Vue.js	11
3.1.2	AdminLTE	12
3.1.3	Bootstrap	12
3.1.4	CoPilot	12
3.1.5	Laravel Framework	13
3.1.6	MySQL	17
3.2	Atividades Desenvolvidas	18
3.2.1	Análise de dados e diagramas da aplicação	18
3.2.2	Explicação das telas	18
4	RESULTADOS OBTIDOS	26
4.1	Desafios Encontrados	26
4.2	Estado Atual do Desenvolvimento	26
4.3	Trabalhos Futuros	27
	Conclusão	29
	REFERÊNCIAS	30

1 Introdução

Atualmente grande parte dos procedimentos realizados nas universidades é feito de forma digital, como por exemplo as solicitações de matrícula dos alunos, os cadastros na biblioteca e até mesmo o diário de classe. No entanto, ainda existem alguns processos que são feitos de forma manual, sem a assistência de computadores.

Periodicamente, o Conselho Diretor (CONDIR) da Universidade Federal de Uberlândia emite uma resolução que regulamenta a avaliação docente no que se refere à Progressão ou Promoção de Carreira. Um dos pré-requisitos para a Progressão ou Promoção é a obtenção da pontuação de referência da classe e nível para a qual o docente pode avançar. Essa pontuação é calculada com base em todas as atividades realizadas pelo docente no período avaliado, sendo que cada atividade tem o seu critério de pontuação, também detalhado na resolução (CONDIR, 2017). Assim, cada docente da universidade precisa submeter um relatório que descreve e comprova essas atividades. A data para submissão é diferente para cada docente e varia de acordo com a data de contratação.

A universidade não possui um padrão bem definido para criação do relatório discriminando a pontuação obtida e nem um sistema computacional que os docentes possam utilizar para realizar a gestão dessas informações. Por causa disso esse relatório é criado de forma manual pelos docentes, que precisam sempre buscar em seus papéis impressos e/ou computadores documentos que descrevam e comprovem como e quando cada uma das atividades foram realizadas. Além disso, como cada docente pode fazer um relatório com formato diferente, a leitura e interpretação dos mesmos pela comissão que o avaliará a quem se destinam pode ser dificultada, já que a organização do conteúdo entre cada um pode ser diferente, e, muitas vezes, até confusa.

1.1 Objetivos

O objetivo geral deste trabalho é implementar uma solução que possibilite a gestão das atividades dos docentes e a geração dos relatórios a serem submetidos. Os objetivos específicos do trabalho proposto são:

1. Permitir que o docente registre suas atividades conforme elas forem sendo executadas;
2. Permitir que o docente visualize a sua pontuação atual e o quão próximo ele está do próximo nível na sua progressão ou promoção de carreira;

3. Capacidade de criação automática do relatório de atividades, utilizando as informações submetidas pelo docente no período (descrição das atividades, comprovantes, etc.).

2 Detalhamento do Problema

2.1 Requisitos Funcionais

Requisitos funcionais são afirmações de serviços que o sistema deve fornecer, como o sistema deve reagir à cada comando, e como o sistema deve se comportar em situações particulares. Em alguns casos, os requisitos funcionais também devem dizer o que o sistema não deve fazer (SOMMERVILLE, 2007, p. 81). Os requisitos funcionais levantados antes e durante o desenvolvimento da aplicação foram:

- Todos os usuários do sistema devem possuir um ou mais papéis, que determinam as ações que eles serão permitidos executar no sistema.
- Um usuário com papel de administrador deve ser capaz de cadastrar, visualizar, editar e apagar outros usuários, além de associar papéis.
- Um usuário com papel de secretário deve ser capaz de cadastrar, visualizar, editar e apagar novos tipos de atividades no sistema. As seguintes informações devem ser armazenadas em relação a um tipo de atividade
 - **Resumo:** breve descrição do que se trata a atividade;
 - **Observação:** alguma consideração em relação à atividade. Por exemplo, a forma como será pontuada;
 - **Descrição da pontuação:** descreve como a pontuação da atividade deve ser calculada;
 - **Categoria:** a qual categoria a atividade pertence, por exemplo *Atividades de Ensino* ou *Atividades de Orientação*.
- Um usuário com papel de docente deve ser capaz de cadastrar, visualizar, editar e apagar suas atividades no sistema. As seguintes informações devem ser armazenadas em relação a uma atividade desenvolvida
 - **Identificador:** número da atividade de acordo com uma resolução do Conselho Diretor da Universidade;
 - **Observação:** alguma consideração que o docente deseja fazer em relação à atividade desenvolvida;
 - **Data de início e data de fim:** período em que a atividade foi realizada;

1 TABELA A1.2 - ATIVIDADES DE ORIENTAÇÃO

Tabela 1: ATIVIDADES DE ORIENTAÇÃO -
12/03/2016 a 05/12/2017

Item	Descrição	Data Início	Data Fim	Ptos Item	Unidade Ptos	Qte	Ptos	Págs.	Observação
14	Trabalho de Conclusão de Curso 2 Aluno: Pedro Eduardo de Paula Naves obs: o comprovante não apresenta a data início e fim de orientação. Assim, estabeleci essas datas como sendo o início e fim do semestre da graduação	08/08/2016	17/12/2016	1	Por aluno/ Mês completo (teto de 160 pontos)	4	4	2	Correto: ☐ Correção: ☐
14	Trabalho de Conclusão de Curso 2 Aluno: Silvestre Silverio da Silva obs: o comprovante não apresenta a data início e fim de orientação. Assim, estabeleci essas datas como sendo o início e fim do semestre da graduação	08/08/2016	17/12/2016	1	Por aluno/ Mês completo (teto de 160 pontos)	4	4	3	Correto: ☐ Correção: ☐
14	Trabalho de Conclusão de Curso 2 Aluno: André Silva de Oliveira obs: o comprovante não apresenta a data início e fim de orientação. Assim, estabeleci essas datas como sendo o início e fim do semestre da graduação	03/04/2017	03/08/2017	1	Por aluno/ Mês completo (teto de 160 pontos)	4	4	4	Correto: ☐ Correção: ☐

2

Figura 1 – Modelo do relatório de atividades

- **Comprovante(s)**: um ou mais arquivos no formato PDF que comprovam a realização da atividade, como um diário de classe ou declaração da Coordenação de Curso;
- **Nomes de alunos ou co-autores**: uma lista de nomes de alunos ou docentes que participaram de alguma forma da realização da atividade.
- Um usuário com papel de docente deve ser capaz de gerar um relatório no formato PDF contendo todas as atividades desenvolvidas, e seus comprovantes devem ser incluídos ao final. Um modelo para tal relatório é mostrado na Figura 1.

2.2 Requisitos Não Funcionais

Requisitos não funcionais são restrições nas funções ou serviços oferecidos pelo sistema. Eles podem estar relacionados à alguma propriedade emergente do sistema como por exemplo o tempo de resposta ou o espaço de armazenagem, ou então podem definir restrições como a capacidade de dispositivos de entrada/saída e as representações de dados do sistema. Esses tipos de requisitos muitas vezes se aplicam ao sistema como um todo,

ao invés de a uma funcionalidade ou serviço em específico (SOMMERVILLE, 2007, p. 82). Os requisitos funcionais levantados antes e durante o desenvolvimento da aplicação foram

- **Usabilidade:** os usuários do sistema devem ser capaz de utilizar o sistema sem dificuldades após o período de aprendizagem;
- **Implementação:** o sistema deve ser desenvolvido utilizando tecnologias *web* para que possa ser acessado a partir de dispositivos com acesso à internet, como celulares, tablets e computadores.

3 Desenvolvimento

Este capítulo apresenta as etapas desenvolvidas para a execução deste projeto. O desenvolvimento compreende as seguintes etapas: escolha das tecnologias utilizadas, modelagem de dados e desenvolvimento da aplicação em si.

3.1 Tecnologias Utilizadas

Este projeto foi desenvolvido em duas frentes: um desenvolvimento *front-end* e outro *back-end*. O *front-end* foi baseado em um modelo, ou *template*, chamado CoPilot, uma implementação em Vue.js do tema AdminLTE (desenvolvido com Bootstrap). Já o *back-end* foi feito utilizando o *framework* Laravel, na linguagem PHP .

3.1.1 Vue.js

O Vue.js (pronunciado *view*) é um *framework* JavaScript progressivo, de código-aberto, para a criação de interfaces de usuário. Iniciado em 2014 por Evan You, a ideia por trás do *framework* era extrair tudo que ele gostou enquanto trabalhava com o AngularJS e construir algo leve e sem os conceitos extras (CROMWELL, 2016). Assim, ao contrário de outros *frameworks* monolíticos, o Vue.js foi projetado para ser adotável de forma incremental e pode ser facilmente integrado com outras bibliotecas ou projetos existentes. A Figura 2 mostra uma aplicação exemplo “Hello, world” criada utilizando o Vue.js.

Segundo a página principal do projeto (Vue.js, 2018), o Vue.js se caracteriza por ser

- **Acessível:** não são necessários conhecimentos além de HTML, CSS e JavaScript



Figura 2 – Exemplo de renderização declarativa do Vue.js

para começar a criar aplicações;

- **Versátil:** é um *framework* simples, capaz de lidar com aplicações pequeno ou grande porte;
- **Alta performance:** a biblioteca minificada do Vue.js tem tamanho de 20kb quando comprimida com GZIP e tem mínima necessidade de esforço para otimizações.

3.1.2 AdminLTE

O AdminLTE é um *template* para desenvolvimento de *dashboards* ou “painéis de controle”. Possui código-aberto, construído usando Bootstrap 3 e design responsivo. Entre as principais características do AdminLTE estão

- **Leve:** mesmo possuindo diversas funcionalidades, ainda assim é leve e rápido;
- **Alta compatibilidade:** possui suporte para a maioria dos principais navegadores, incluindo Safari, Internet Explorer 9+, Google Chrome, Firefox e Opera;
- **Comunidade:** por ser de código-aberto, dúvidas ou problemas encontrados podem ser postados no repositório do projeto no GitHub¹, onde qualquer membro da comunidade pode prestar ajuda e propor soluções.

3.1.3 Bootstrap

O Bootstrap é a biblioteca de componentes *front-end* mais popular do mundo (Bootstrap, 2018), e é voltado principalmente para o desenvolvimento de projetos *mobile*. Todos os componentes oferecidos, como alertas, cartões e formulários, são responsivos, ou seja, eles se redimensionam, se escondem, diminuem ou ficam maiores dependendo do tamanho do dispositivo em que a página é visualizada.

Atualmente a biblioteca está na versão 4, o que trouxe muitas melhorias em relação a versão 3, como a simplificação dos componentes de navegação. No entanto, a versão usada pelo AdminLTE ainda é a 3.3.7 e por isso esta é a versão utilizada neste projeto.

3.1.4 CoPilot

O CoPilot é uma implementação em Vue.js baseado no *template* responsivo AdminLTE, que por sua vez utiliza o *framework* Bootstrap. A Figura 3 mostra a página inicial do CoPilot sem nenhuma customização.

Utilizar projetos como o CoPilot como base para desenvolver outras aplicações reduz o tempo necessário para iniciar o desenvolvimento e ajuda a evitar possíveis problemas

¹ <https://github.com/almasaeed2010/AdminLTE>

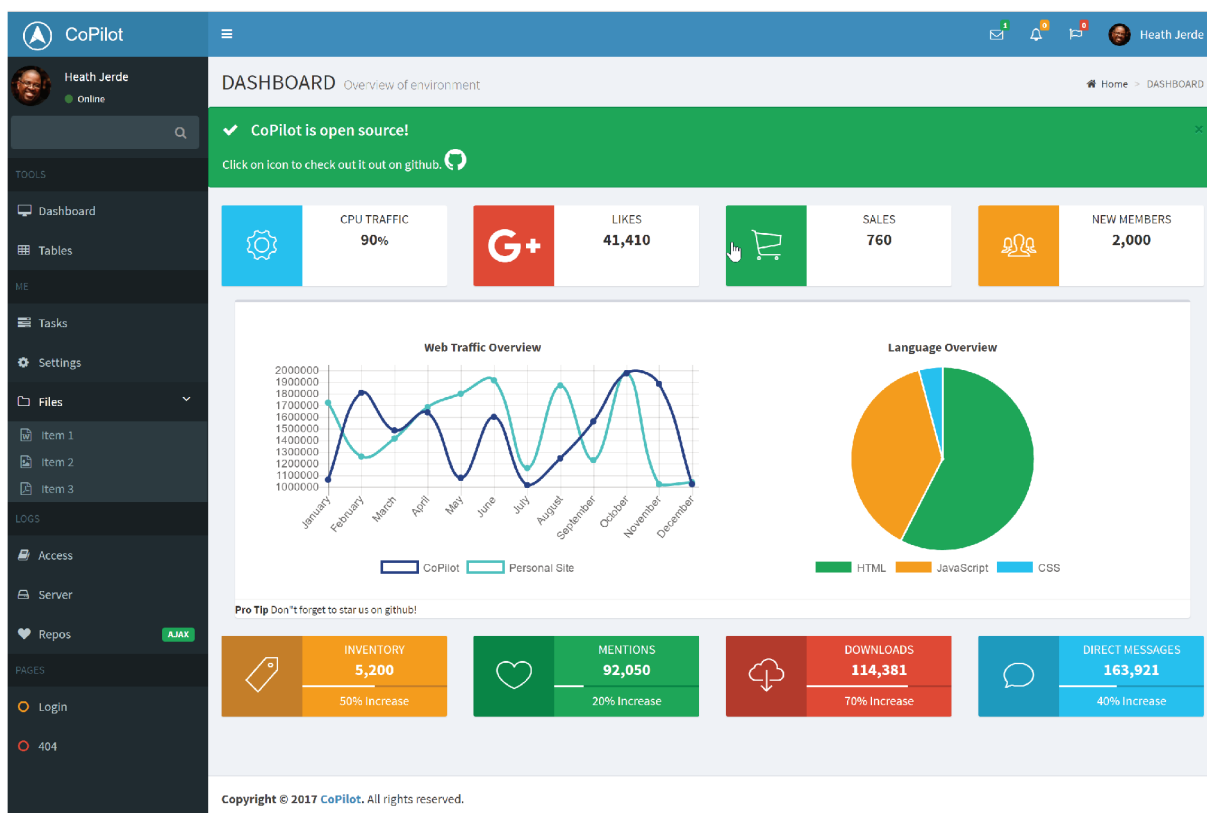


Figura 3 – Página inicial do CoPilot

de configuração que um desenvolvedor menos experiente pode encontrar. Por exemplo, a maioria das aplicações JavaScript modernas utiliza um *module bundler* como o Webpack² ou o Gulp.js³, cujo principal objetivo é permitir que o código seja escrito de forma modular. Além disso, o *module bundler* também pode automatizar processos como a ofuscação e minificação de código e outras operações de pré-processamento. O CoPilot vem com uma configuração pronta do Webpack, o que faz com que o desenvolvedor possa aproveitar todas as vantagens provindas do uso do *module bundler* sem ter que se preocupar com sua configuração inicial.

3.1.5 Laravel Framework

O Laravel é um *framework* PHP de código aberto baseado no Symfony⁴ e voltado para o desenvolvimento de aplicações web seguindo a arquitetura MVC (Model-View-Controller). Diferente de *frameworks* PHP anteriores, compostos de bibliotecas inflexíveis e nos quais desenvolvedores precisam trabalhar estritamente da forma definida pelo autor do framework, o Laravel oferece componentes que os desenvolvedores podem usar para criar aplicações da forma como quiserem (BEAN, 2015). Isso fez com que o Laravel se

² <<https://webpack.js.org/>>

³ <<https://gulpjs.com/>>

⁴ <<https://symfony.com/>>

tornasse o *framework* PHP mais popular do mundo, somando mais de 30 mil “estrelas” no GitHub ([GitHub, 2017](#)).

Segundo ([BEAN, 2015](#)), entre as principais vantagens do Laravel estão

- **Modularidade:** o Laravel é construído a partir de mais de 20 bibliotecas diferentes, e ele mesmo é dividido em módulos. Além disso, ele utiliza o Composer⁵ como gerenciador de dependências, o que faz com que todos esses componentes possam ser facilmente atualizados;
- **Testabilidade:** o Laravel dispõe de diversas bibliotecas que facilitam a criação de testes automatizados. É possível por exemplo visitar rotas, inspecionar o código HTML resultante e simular um usuário autenticado sem precisar instalar dependências adicionais;
- **Roteamento:** existe grande flexibilidade na criação das rotas da aplicação. Por exemplo, é possível até mesmo associar uma simples função anônima a um verbo HTTP como GET ou POST ;
- **Gerenciamento de configuração:** a configuração da aplicação para o ambiente atual é definida em arquivo de texto de nome `.env` (*dot env*) na raiz do projeto. Isso simplifica a instalação, migração e replicação do projeto para diferentes ambientes. Por exemplo, as credenciais de acesso ao banco de dados ou ao servidor de e-mail podem ser diferentes, ou as mensagens de erro precisam ser exibidas de outra forma, ou simplesmente a configuração usada no desenvolvimento difere da de produção.

A Figura 4 mostra a estrutura de pastas de um projeto Laravel e o seu arquivo de configuração.

Com o Laravel é possível criar facilmente *controllers* RESTful, que atendem as especificações REST (*Representational State Transfer*). REST é um estilo arquitetural que define um conjunto de restrições e propriedades baseadas no HTTP, com objetivo de prover interoperabilidade entre sistemas na Internet. A principal abstração de informação no REST é um **recurso** ([REST API Tutorial, 2018](#)). Para criar um *controller* no Laravel que gerencia as requisições do recurso “usuário”, basta executar o seguinte comando na raiz do projeto:

```
php artisan make:controller UserController
```

Com isso será criado um arquivo `UserController.php` contendo métodos para cada operação disponível para esse recurso. Depois, é necessário registrar a rota adicionando a seguinte linha no arquivo `api.php`:

⁵ <<https://getcomposer.org/>>

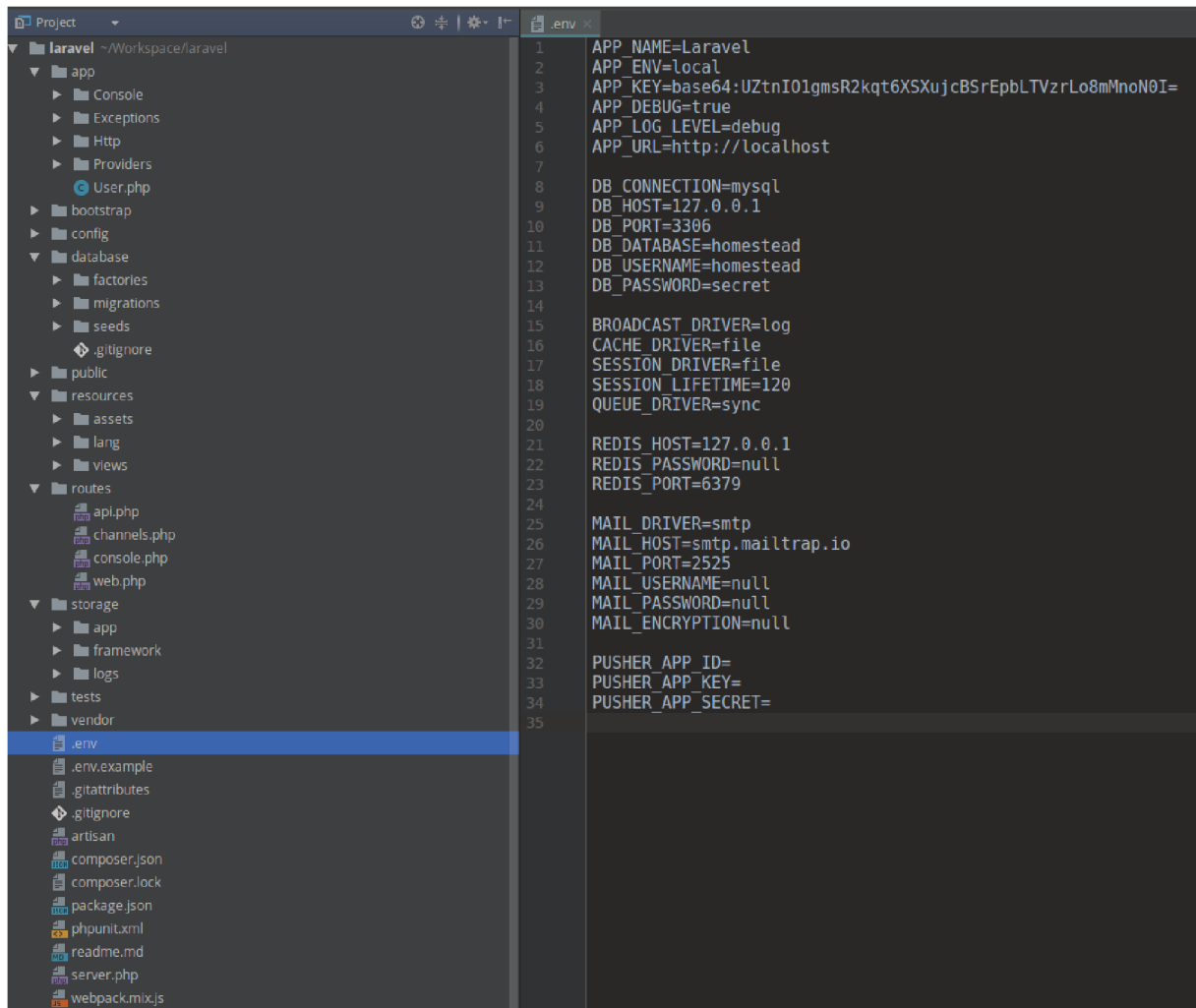


Figura 4 – Estrutura de um projeto em Laravel 5 e arquivo de configuração `.env`

```
<?php
```

```
Route::resource('users', 'UserController');
```

Essa única instrução faz com que sejam criadas rotas para cada operação RESTful (*index*, *store*, *update*, etc.) do *controller* criado. Executando o comando `php artisan route:list` são exibidas todas as rotas disponíveis:

Method	URI	Action
GET HEAD	users	App\Http\Controllers\UserController@index
POST	users	App\Http\Controllers\UserController@store
GET HEAD	users/{id}	App\Http\Controllers\UserController@show
PUT	users/{id}	App\Http\Controllers\UserController@update

```
| DELETE      | users/{id} | App\Http\Controllers\UserController@destroy |
+-----+-----+-----+
```

Dessa forma, uma requisição do tipo GET para a rota `/users` retorna a lista de usuários, enquanto que uma requisição do tipo POST contendo um objeto JSON para essa mesma rota insere um novo usuário, conforme o exemplo a seguir. Toda a comunicação feita entre o *front-end* e o *back-end* da aplicação é feita seguindo esse modelo.

```
curl -X POST \
  http://localhost/api/users \
  -d @- << EOF

{
  "name": "Rafael Vieira Mendes",
  "email": "rafaelvmendes@ufu.br",
  "password": "123456"
}
EOF
```

Outro ponto importante a ser mencionado é a forma de escolha do sistema de banco de dados. Mesmo que a versão atual da aplicação desenvolvida esteja utilizando o MySQL, o Laravel permite que o sistema de banco de dados utilizado seja facilmente trocado, pois todas as consultas e operações foram feitas por meio de suas interfaces. Para fazer a troca, basta que o parâmetro `DB_CONNECTION` seja alterado no arquivo `.env`. O trecho de código a seguir mostra um exemplo de criação de um novo usuário no Laravel:

```
<?php

public function store(Request $request)
{
    // ...

    $user = User::create([
        'name'      => $request['name'],
        'email'     => $request['email'],
        'password'  => bcrypt($request['password']),
    ]);

    // ...
}
```

É interessante citar também algumas outras bibliotecas que foram adicionadas ao projeto na parte do *back-end*. O Twig⁶ é uma biblioteca PHP de *templates* que foi utilizada para geração da tabela de atividades nos relatórios. Também foram usados o wkhtmltopdf⁷ e Ghostscript⁸ para transformar a tabela de atividades em um arquivo PDF e para juntar esse arquivo aos comprovantes anexados. Os pacotes Entrust⁹ e Passport¹⁰ foram utilizados para gerenciar o sistema de permissões e autenticação respectivamente.

3.1.6 MySQL

Para a persistência dos dados da aplicação foi escolhido o MySQL. O MySQL é um sistema de gerenciamento de banco de dados relacional, de código-aberto, e é utilizado por grandes empresas e organizações como Google, Yahoo! e NASA. Em 2007 já haviam sido feitos mais de 100 milhões downloads do instalador do MySQL. Alguns fatores-chave que contribuíram para a popularização do MySQL foram (GILMORE, 2006, pp. 477–481)

- **Flexibilidade:** o MySQL pode ser instalado em diversos sistemas operacionais (Windows, Linux, MacOS, FreeBSD, etc.) e várias *APIs* estão disponíveis para as linguagens de programação mais populares como C++, Java e PHP ;
- **Performance:** desde o início esse foi um dos principais focos no desenvolvimento do MySQL. Foram implementados ao longo do tempo funcionalidades como *cache* de consultas e replicação de instâncias;
- **Licenças flexíveis:** além da licença comercial também é oferecida uma versão gratuita licenciada sob a GPL¹¹, o que significa que se o seu programa também for licenciado sob a GPL você pode até mesmo modificar e redistribuir o código-fonte do MySQL, desde que as mudanças também estejam de acordo com os termos da licença;
- **Comunidade ativa:** como existem milhares de projetos que dependem do MySQL para gerenciar vários tipos diferentes de informações como registros de *log*, e-mail, conteúdo web, etc., também foram escritos centenas de tutoriais que cobrem todos os aspectos do *software*. Se for necessário buscar suporte para um problema específico ou alguma dúvida, basta criar uma postagem em algum fórum de discussão de desenvolvimento.

⁶ <<https://twig.symfony.com/>>

⁷ <<https://wkhtmltopdf.org/>>

⁸ <<https://www.ghostscript.com/>>

⁹ <<https://github.com/Zizaco/entrust>>

¹⁰ <<https://laravel.com/docs/5.6/passport>>

¹¹ *General Public License* <<https://www.gnu.org/licenses/gpl.html>>

3.2 Atividades Desenvolvidas

3.2.1 Análise de dados e diagramas da aplicação

A modelagem de dados é uma parte importante da modelagem de sistemas pois define a forma lógica dos dados processados. A técnica de modelagem de dados mais utilizada é o modelo Entidade-Relacionamento-Atributo (ERA), que exhibe as entidades, seus atributos associados e os relacionamentos entre entidades. É interessante notar que a UML¹² não define uma notação específica para essa modelagem de banco de dados, já que ela assume que o processo de desenvolvimento é orientado a objetos e que os dados são modelados usando os objetos e seus relacionamentos. (SOMMERVILLE, 2007, pp. 118–119).

O modelo ERA da aplicação é mostrado na Figura 5, gerado com auxílio do SQL Power Architect¹³. Para completar esse modelo, um dicionário de dados é descrito na Tabela 1. Como os modelos gráficos são geralmente pobres em detalhes, o dicionário de dados pode ser utilizado para descrever mais detalhadamente as entidades e relacionamentos do sistema.

Para melhor entendimento de como deveriam ser os fluxos da aplicação foi criado um diagrama de casos de uso. Os casos de uso constituem uma técnica baseada em cenários para o levantamento de requisitos. Eles identificam interações individuais com o sistema e podem ser documentados em forma de texto ou modelos UML que detalham mais o cenário. Os elementos essenciais da notação de caso de uso são os agentes (atores) no processo, representados como bonecos, e as classes de interação, representadas como uma elipse com identificação (SOMMERVILLE, 2007, pp. 102–103).

O diagrama da Figura 6 mostra os casos de uso gerais do sistema para os atores docente, secretário e administrador. Esses atores também representam cada papel que um usuário pode ter no sistema. Note que mesmo que para realizar quaisquer operações no sistema os usuários precisem estar autenticados, a ação “fazer login” não foi incluída como caso de uso, já que o processo de autenticação no sistema não é o foco.

3.2.2 Explicação das telas

Como o AdminLTE foi escolhido para servir de base para o *front-end* da aplicação, não foi necessário desenvolver o *layout* desde o início, isto é, a organização dos menus e conteúdo estava já pré-definida. Quase todas as telas do sistema foram feitas no modelo CRUD¹⁴, possuindo um botão para criar um novo registro, visualizar e/ou editar um

¹² Linguagem de Modelagem Unificada

¹³ Ferramenta de modelagem de dados, disponível em <https://code.google.com/archive/p/power-architect/>

¹⁴ *Create, Read, Update, and Delete*

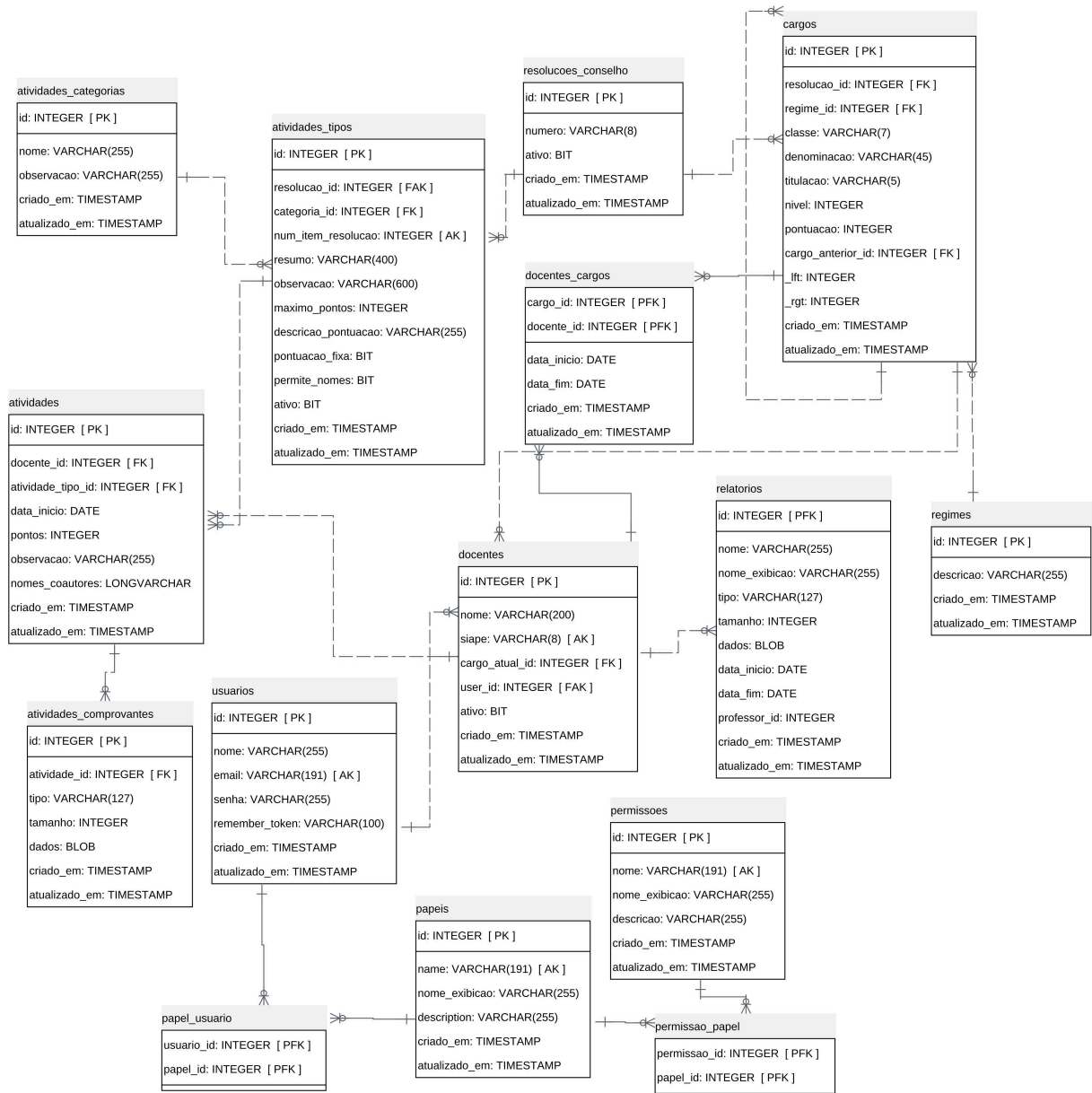


Figura 5 – Modelo Entidade-Relacionamento-Atributo (ERA) da aplicação

Nome	Descrição
atividades	Atividades de um docente.
atividades_categorias	Categorias de tipos de atividades segundo uma resolução do conselho. Exemplo: Atividades de Ensino.
atividades_comprovantes	Comprovantes das atividades submetidas.
atividades_tipos	Tipos de atividades segundo uma resolução do conselho. Exemplo: Artigo técnico-científico publicado em periódico indexado.
cargos	Denominações que um docente pode ter, segundo uma resolução do conselho. Exemplo: Assistente, titulação M, nível I, com pontuação de referência 630.
docentes	Docentes que fazem uso do sistema.
docentes_cargos	Relação entre um docente e a sua denominação atual.
permissoes_papeis	Relações entre um papel e suas permissões.
permissoes	Tipos de permissões do sistema, que permitem utilizar alguma funcionalidade específica. Exemplo: criar usuário.
regimes	Regimes de trabalho que uma denominação (cargo) pode ter. Exemplo: 40 horas.
relatorios	Relatórios gerados para um período específico, com a lista de atividades e seus comprovantes.
resolucoes_conselho	Resoluções emitidas pelo Conselho Diretor
papel_usuario	Relações entre um usuário e seus papéis.
papeis	Papéis do sistema, que representam um conjunto de permissões que um usuário possui.
usuarios	Usuários utilizadores do sistema.

Tabela 1 – Dicionário de dados do modelo da Figura 5

registro existente e apagar um registro. Foi também implementado um filtro simples para todas as telas que possuem tabelas. Se o campo “Busca” acima da tabela estiver preenchido e o botão “Buscar” for pressionado, é feita uma nova consulta na base dados que retorna para usuário apenas as linhas em que pelo menos uma das colunas contém parte do texto digitado no campo.

A Figura 7 mostra a tela onde são gerenciadas as atividades cadastradas a partir de um determinada resolução do conselho. Como visto no modelo de dados (Figura 5), os tipos de atividades sempre estão associados à uma resolução do conselho emitida em determinado ano. Logo, é necessário selecionar a resolução desejada para gerenciar as atividades da mesma através do campo de seleção no canto superior esquerdo da tabela. Apenas usuários com um perfil de *secretário* ou *administrador* possuem acesso de escrita nessa tela, ou seja, apenas esses usuários são capazes de manipular essas informações. No entanto, usuários com perfil de *docente* podem visualizar esses dados normalmente.

Os botões do lado direito da tabela representam as ações que podem ser feitas

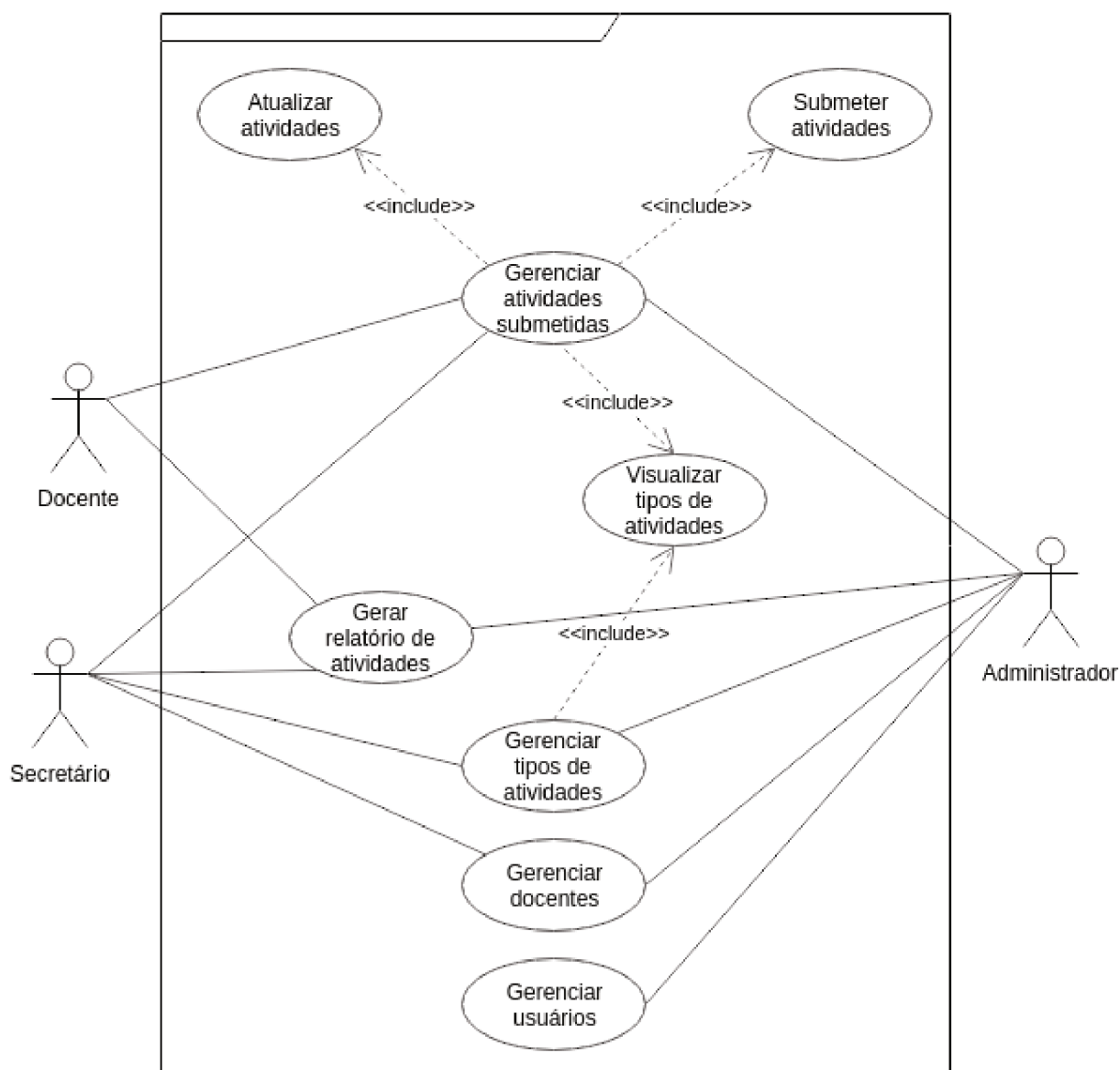


Figura 6 – Diagrama de casos de uso do sistema

naquela linha. Esses botões só são mostrados para o usuário caso o mesmo tenha permissão para realizar aquela ação. A Figura 8 mostra a *modal*¹⁵ de edição de um tipo de atividade que é exibida quando o usuário pressiona o botão “+ atividade” ou um dos botões de edição mostrados na Figura 7. Nessa janela, o usuário pode criar ou atualizar os dados de um dos tipos de atividades que os docentes podem cadastrar. Os campos obrigatórios como resumo e pontuação são marcados com um “*” (asterisco) vermelho.

A Figura 9 mostra a tela na qual o docente pode cadastrar suas atividades, de acordo com os tipos cadastrados anteriormente. Novamente, se o botão de “+ atividade” ou um dos botões de edição for pressionado, é exibida uma modal como a da Figura 10.

¹⁵ Um tipo de janela *popup* que faz com que o usuário tenha que interagir com ela antes de voltar para a janela principal

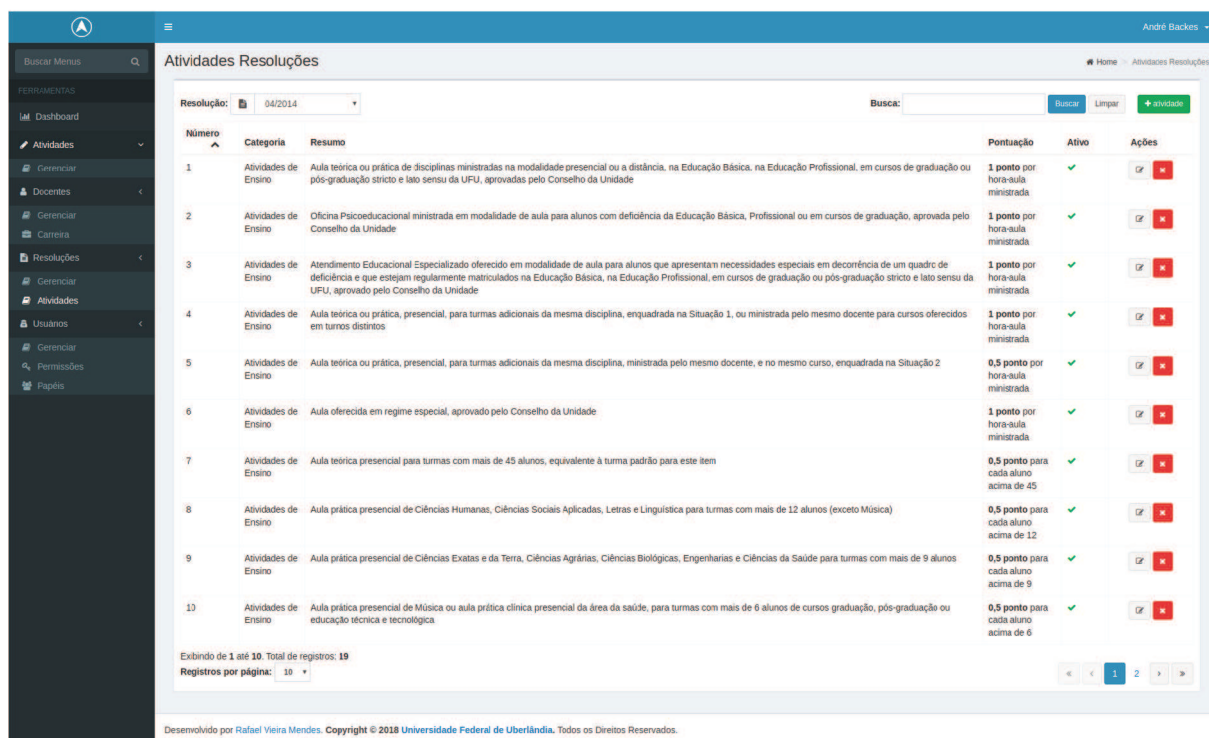


Figura 7 – Tela de cadastro de tipos de atividades descritas na resolução do conselho

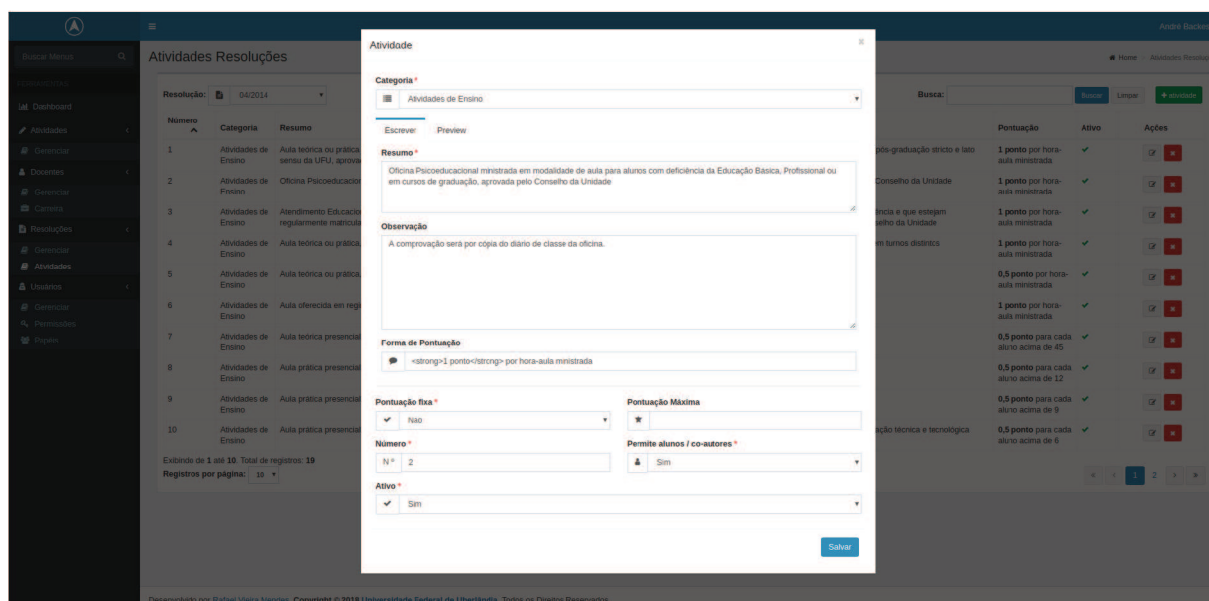


Figura 8 – Janela modal para cadastro ou edição de um tipo de atividade

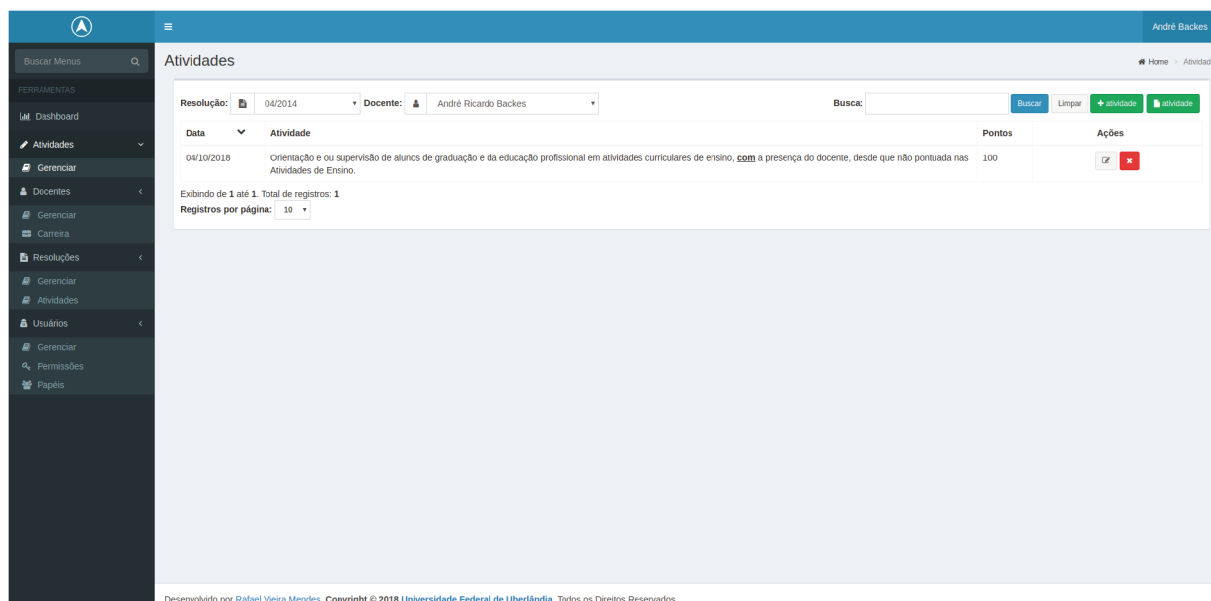
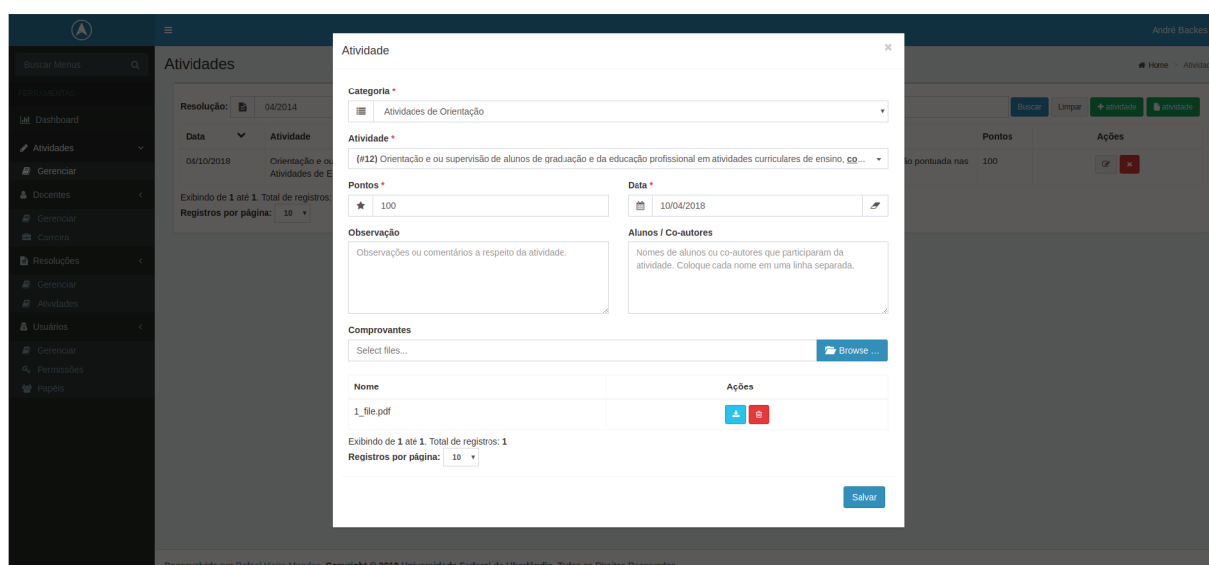


Figura 9 – Tela de cadastro das atividades realizadas pelo docente

Figura 10 – Janela *modal* para cadastro ou edição de uma atividade

Nela o docente deve escolher o tipo de atividade e preencher alguns detalhes a respeito da mesma, como por exemplo os nomes dos alunos ou co-autores (se houverem) e a pontuação obtida, de acordo que com a regra daquela atividade. O docente também pode fazer o *upload* de um ou mais comprovantes apertando o botão “Enviar”, que abre um *popup* para a escolha do(s) arquivo(s). Na tela da Figura 11 é possível gerar os relatórios contendo todas as atividades realizadas em um determinado período e os seus comprovantes.

A Figura 12 mostra a tela de gerenciamento de usuários. Na Figura 13 é exibida a *modal* de edição ou cadastro de usuário.

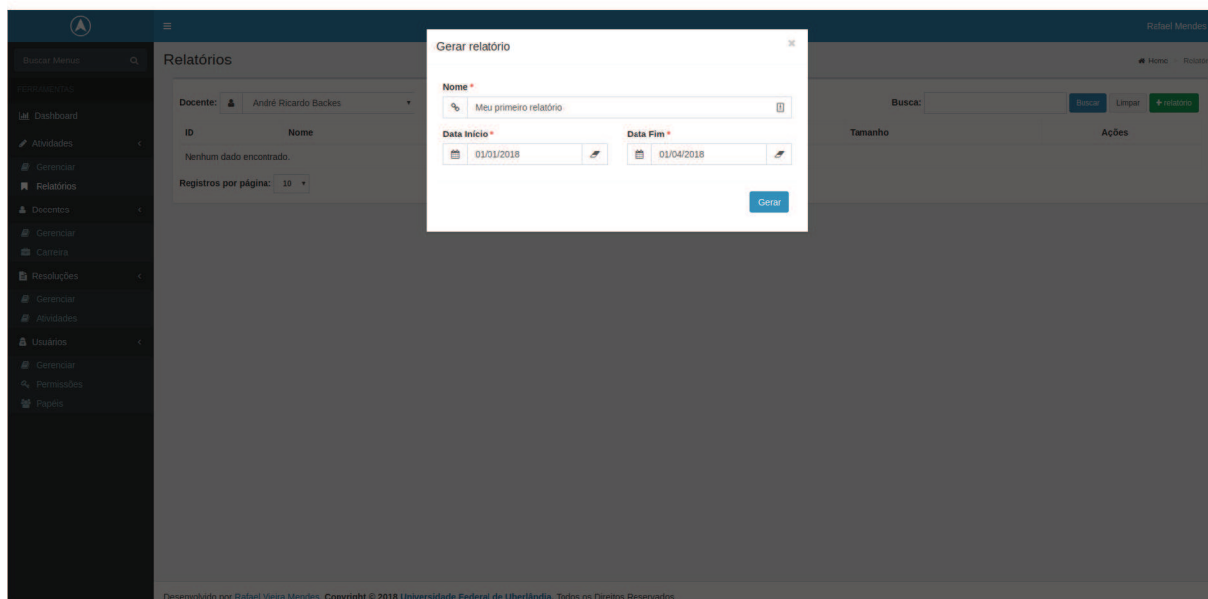


Figura 11 – Tela para geração de relatórios

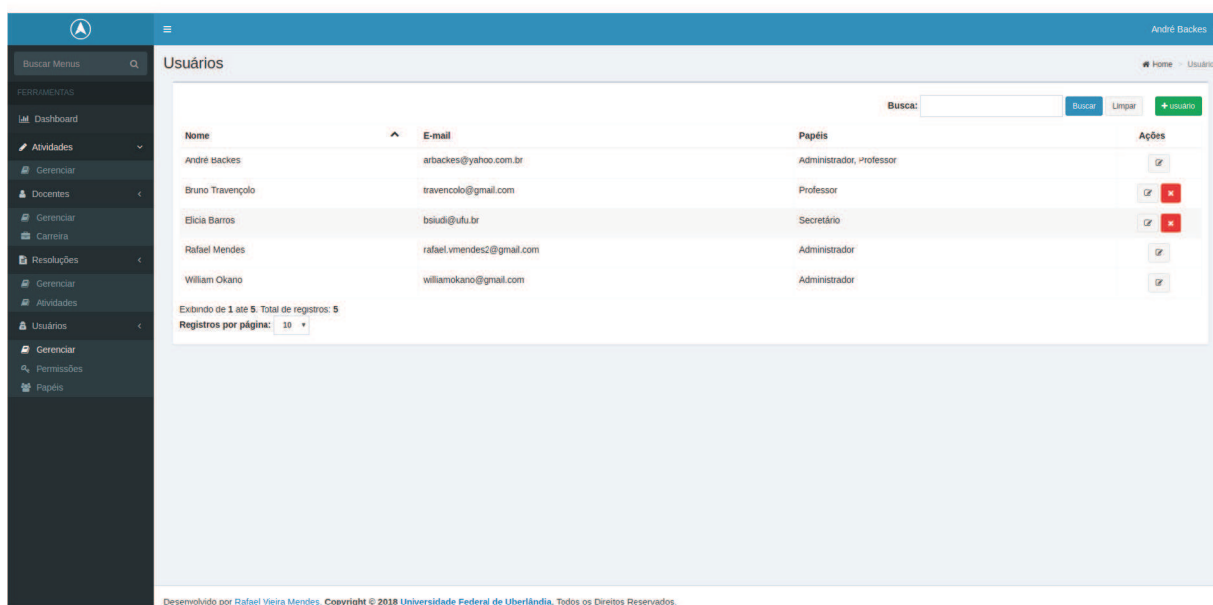


Figura 12 – Tela de gerenciamento de usuários

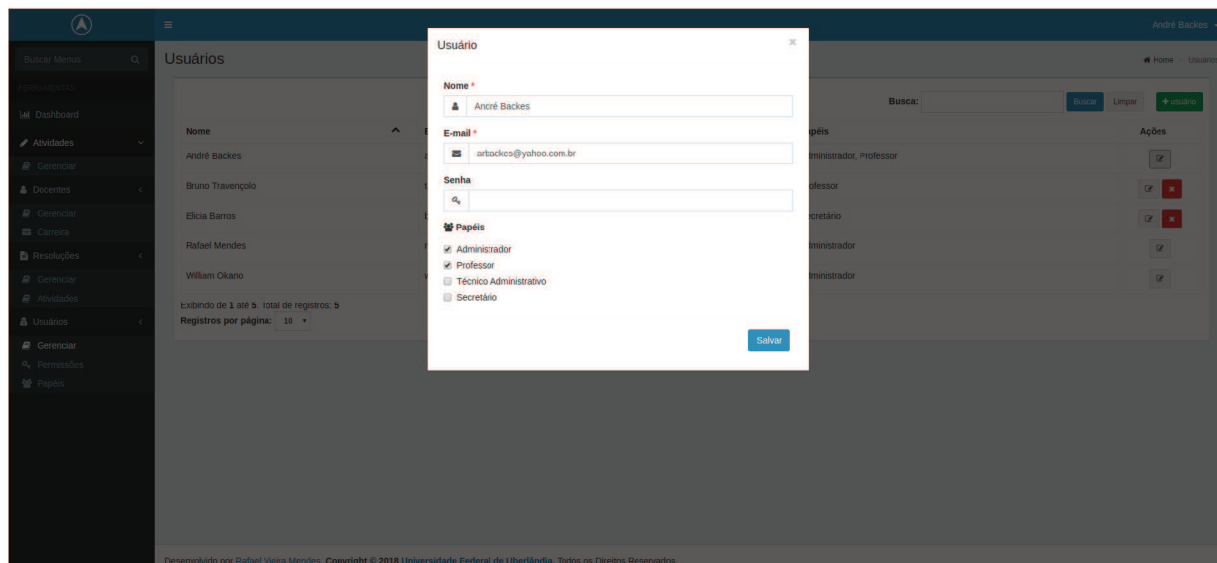


Figura 13 – Janela *modal* para edição de um usuário, incluindo atribuição de papéis

4 Resultados Obtidos

4.1 Desafios Encontrados

Um dos motivos que justificam a escolha das tecnologias utilizadas é que elas foram selecionadas para aproveitar o desenvolvimento do projeto para aprofundar os conhecimentos em tecnologias *web*, mas essa falta de experiência dificultou um pouco a implementação de algumas funcionalidades. Ao utilizar o Laravel, por exemplo, é desejável também ter conhecimento de quais funcionalidades foram implementadas ou depreciadas nas versões mais recentes da linguagem PHP (atualmente versão 7.2.x), já que novas versões do *framework* quase sempre elevam a versão mínima do PHP para a mais recente e estável da linguagem.

Com relação ao Vue.js, por se tratar de um *framework* relativamente novo, alguns componentes ou bibliotecas menos populares ainda não possuem uma versão nativa para uso. Para utilizar o *bootstrap-fileinput*¹, um componente usado no projeto para controlar o *upload* de arquivos, foi necessário implementá-lo como um componente Vue.js.

Além das dificuldades com as tecnologias, também houveram dificuldades no que diz respeito ao levantamento de requisitos, já foi necessário por exemplo revisar conceitos de Modelagem e Engenharia de Software para a criação do diagrama de casos de uso e modelo de dados. Além disso, os requisitos em si também se alteraram conforme a evolução do sistema pois novas melhorias foram sendo levantadas, o que fez com algumas regras definidas inicialmente precisassem ser alteradas.

4.2 Estado Atual do Desenvolvimento

No estado atual do desenvolvimento, todo o fluxo principal proposto foi implementado. Os docentes podem cadastrar suas atividades e realizar a geração do relatório e usuários com papel de secretário ou administrador podem cadastrar novos tipos de atividades ou níveis de progressão ou promoção de carreira. Foram implementadas até mesmo algumas funcionalidades necessárias para uma melhor experiência de usuário, como uma página de perfil para que o próprio usuário consiga alterar seus dados básicos (e-mail, senha, etc.) e um botão para envio de e-mail de recuperação de senha.

Outro ponto importante é que como não foram criados testes automatizados de tela ou testes unitários, todos os testes de funcionamento do sistema precisam ser feitos manualmente. Ou seja, atualmente não é possível garantir de forma fácil e com confiança

¹ <<https://github.com/kartik-v/bootstrap-fileinput>>

que todas as funcionalidades implementadas continuam funcionando corretamente após algum lançamento de versão.

O código-fonte do sistema pode ser encontrado repositórios a seguir. Na página inicial de cada um (arquivo *README*), existem instruções detalhadas para instalação e montagem do ambiente.

- <<https://gitlab.com/vieiram2/scad-frontend>>: Contém o código do *front-end* do sistema;
- <<https://gitlab.com/vieiram2/scad-backend>>: Contém o código do *back-end* do sistema. Esse repositório também possui uma carga inicial de dados, contendo por exemplo as atividades da resolução de 2017 já cadastradas.

4.3 Trabalhos Futuros

Algumas funcionalidades que não foram especificadas tiveram sua implementação iniciada mas ainda foram completamente finalizadas, por exemplo:

- Gráficos na página principal que apresentam algumas estatísticas a respeito das informações cadastradas no sistema como, por exemplo, a relação pontos/mês para cada docente e o quão próximo determinado docente está de alcançar o próximo nível na sua progressão ou promoção de carreira;
- Permitir fazer o cadastro “massivo” de atividades por meio do botão “atividade” (ícone de documento), ao lado do botão “+ atividade”. Nesse caso, é aberto um *modal* para a seleção de um arquivo no formato “.CSV” (Figura 14) cujo preenchimento pode ser feito a partir do modelo obtido ao apertar o botão “Modelo”. Caso opte por cadastrar as atividades dessa forma, o docente pode inserir nesse arquivo os dados de várias atividades de uma só vez, sendo que cada linha corresponde a uma atividade. A desvantagem em fazer o cadastro dessa forma é que os comprovantes precisam ser enviados em um segundo momento da forma “convencional”, por meio da edição de atividades.

Também foram levantadas algumas ideias para facilitar ainda mais o uso da aplicação e adicionar novas funcionalidades, porém elas não foram implementadas nessa versão da aplicação por não estarem muito bem definidas. Entre elas estão

- **Cálculo automático de pontos:** é desejável que fosse possível calcular automaticamente a quantidade de pontos por atividade baseada em variáveis definidas pelo secretário. Dessa forma, o docente precisaria apenas informar o valor de cada variável;

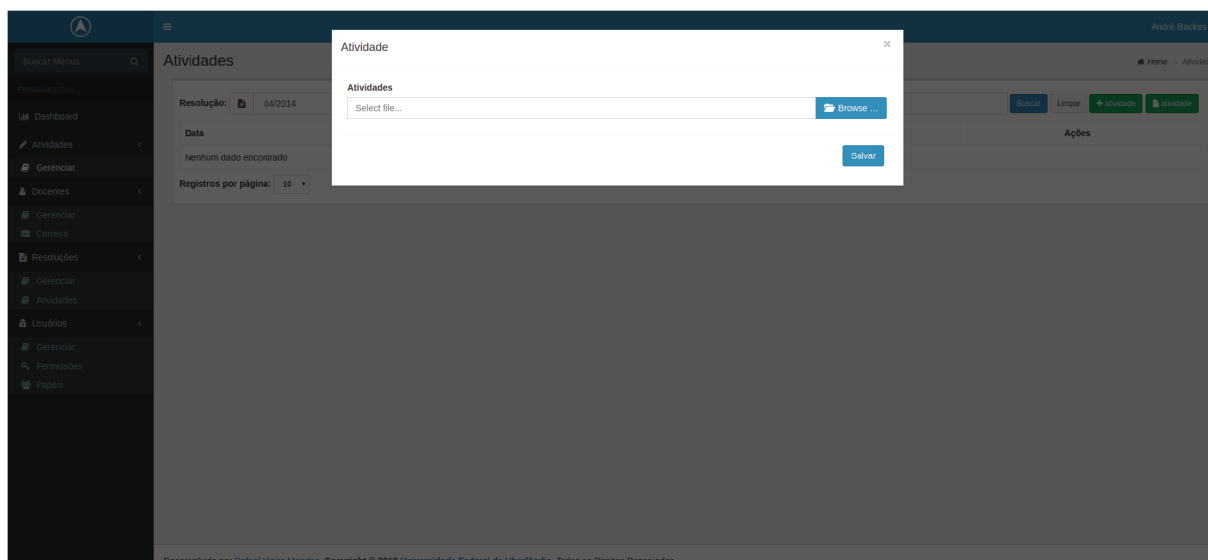


Figura 14 – Janela *modal* para importação “massiva” de atividades

- **Importação de atividades do Currículo Lattes:** a ideia é que exista uma funcionalidade para ler um arquivo XML exportado de um currículo cadastrado na Plataforma Lattes², que já contém diversas atividades cadastradas, e importá-las no sistema. Uma das limitações dessa funcionalidade é que os comprovantes precisariam ser enviados separadamente.
- **Criar testes automatizados:** testes são muito importantes para manter a estabilidade de um sistema em funcionamento, pois garantem que nenhuma parte (coberta pelos testes) parou de funcionar e que assim lançamentos de novas versões possam ser feitos com confiança.

² Plataforma de currículos acadêmicos criada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). <<http://lattes.cnpq.br/>>

Conclusão

Durante o desenvolvimento do projeto foi possível colocar em prática conceitos teóricos que não muito utilizados no dia-a-dia. Além disso, também surgiu uma oportunidade de estudar novas tecnologias para resolver o problema apresentado. Isso contribuiu não só para o desenvolvimento acadêmico como também para o desenvolvimento profissional.

As disciplinas de Modelagem de Software e Engenharia de Software oferecidas pelo curso de Sistemas de Informação foram importantes, já que providenciaram boa parte do conhecimento acadêmico necessário. Foram utilizados conceitos de modelagem de dados para entender melhor os relacionamentos, e por meio dos modelos criados foi possível encontrar soluções e entender “graficamente” o problema a ser resolvido.

Com a implementação do sistema, espera-se que os docentes sejam capazes de organizar melhor suas atividades realizadas e que consigam facilmente identificar o quanto próximos estão de avançar na carreira. Além disso, espera-se também que como os relatórios submetidos que foram gerados pelo sistema facilitem o processo de avaliação.

Referências

- BEAN, M. *Laravel 5 Essentials*. Packt Publishing, 2015. (Community Experience Distilled). ISBN 9781785283291. Disponível em: <<https://books.google.com.br/books?id=BWO4CAAQBAJ>>. Citado 2 vezes nas páginas 13 e 14.
- Bootstrap. *Bootstrap – The most popular HTML, CSS, and JS library in the world*. 2018. Acesso em: 17 jun. 2018. Disponível em: <<https://getbootstrap.com/>>. Citado na página 12.
- CONDIR. *RESOLUÇÃO Nº 03/2017*. 2017. Acesso em: 29 nov. 2017. Disponível em: <<http://www.reitoria.ufu.br/Resolucoes/resolucaoCONDIR-2017-3.pdf>>. Citado na página 6.
- CROMWELL, V. *Evan You*. Between the Wires, 2016. Acesso em: 29 nov. 2017. Disponível em: <<https://web.archive.org/web/20171022024046/https://betweenthewires.org/2016/11/03/evan-you/>>. Citado na página 11.
- GILMORE, W. J. *Beginning PHP and MySQL 5: From novice to professional*. [S.l.]: Apress, 2006. ISBN 9781430231158. Citado na página 17.
- GitHub. *Repositórios PHP mais populares*. 2017. Acesso em: 30 nov. 2017. Disponível em: <<https://github.com/search?l=PHP&q=stars%3A%3E1&s=stars&type=Repositories>>. Citado na página 14.
- REST API Tutorial. *What is REST*. 2018. Acesso em: 3 jul. 2018. Disponível em: <<https://restfulapi.net/>>. Citado na página 14.
- SOMMERVILLE, I. *Engenharia de Software-8ª Edição (2007)*. [S.l.]: São Paulo: Pearson Addison Wesley, 2007. ISBN 9788588639287. Citado 3 vezes nas páginas 8, 10 e 18.
- Vue.js. *Vue.js – The Progressive JavaScript Framework*. 2018. Acesso em: 13 jun. 2018. Disponível em: <<https://vuejs.org/>>. Citado na página 11.