Desenvolvimento do Front-End do Repositório de Trabalhos do IFC - iFBox

Gabriel Frontório[1], **Gabriel Peron**[1], **Mathias Artur Schulz**[1]

[1]Instituto Federal Catarinense - Campus Rio do Sul (IFC)
Rio do Sul – SC – Brasil

gabrielfrontorio@hotmail.com, peronn07@gmail.com, mathiasschulz34@gmail.com

Abstract. *This article discusses and details the construction of the IFBox site frontend and the integration with the API responsible for managing site data. IFBox is a repository of academic work produced by students of the sixth phase of the computer science course of the Federal Institute of Santa Catarina - Rio do Sul Campus - Urban Unit, its development was in the area of Web Development II and aims to function as a TCC repository and articles published by campus students. In the development of the site some design patterns were used, such as MVC, and a framework responsible for managing MVC. Their construction took place in stages and the way they were done are described below, along with the final result of the development.*

Resumo. *O presente artigo aborda e detalha a construção do frontend do site IFBox e da integração com a API responsável por gerenciar os dados do site. O IFBox é um repositório de trabalhos acadêmicos produzido por alunos da sexta fase do curso de ciência da computação do Instituto Federal Catarinense - Campus Rio do Sul - Unidade Urbana, seu desenvolvimento se deu na matéria de Desenvolvimento Web II e tem como objetivo funcionar como um repositório de TCC's e artigos publicados por alunos do campus. No desenvolvimento do site foi usado alguns padrões de projeto, como o MVC, e um framework responsável por gerenciar o MVC. Sua construção se deu em etapas e a forma como foram realizadas estão descritas abaixo, junto com o resultado final do desenvolvimento.*

1. Introdução

Os trabalhos acadêmicos, artigos científicos e afins, são figuras presentes durante toda a trajetória do graduando no curso superior. Fazer tais trabalhos e artigos demanda tempo e esforço, mesmo assim muitas vezes os trabalhos não são aproveitados como deveriam após a sua entrega. Pensando em divulgar mais os trabalhos e artigos elaborados em nossa instituição de ensino surgiu o projeto de um repositório online para armazenar tais arquivos.

Esses trabalhos estarão disponíveis para qualquer pessoa que acesse o site o repositório, batizado de IFBox. O IFBox foi construído usando métodos e padrões aprendidos durante o curso de ciência da computação, fazendo assim com que todo o conhecimento passado pelos professores retornasse com uma pequena contribuição para a instituição.

Para o desenvolvimento do repositório foram usadas metodologias de diversas matérias que preenchem a grade curricular do curso, fazendo assim com que ele não

seja um projeto específico de uma matéria, mas sim um conjunto de técnicas aprendidas durantes diversas aulas.

2. Model View Controller

O MVC (Model View Controller) em alguns casos é referido pelo termo arquitetura (architecture) e em alguns casos é referido pelo termo padrões de projeto (design patterns), contudo o MVC utiliza outros padrões de projeto para prover a solução do problema (Duarte, 2011).

Os padrões de projeto não são implementações, mas sim descrições de como classes e objetos devem ser organizados para solucionar um determinado problema no projeto. Sendo assim, o MVC possui como objetivo solucionar uma categoria de problemas como:

- Interfaces com o usuário com mudanças constantemente;
- Usuário exigindo novas funcionalidades;
- Aplicação implementada em outra plataforma;
- Mesma aplicação com diferentes requisitos de acordo com cada usuário (Almeida, 2019; Duarte, 2011).

O padrão de projeto MVC consiste em três tipos de objetos que compõem camadas bem definidas, que são: Model, View e Controller.

Modelo (Model): É a camada que compõe o objeto da aplicação, responsável por implementar a lógica da regras de negócio e realizar o armazenamento persistente. Frequentemente o modelo funciona como uma aproximação de um processo do mundo real;

Visão (View): É a camada que compõe a apresentação do sistema para o usuário;

Controlador (Controller): É a camada que conduz a aplicação, no qual recebe eventos do mundo externo, interage com o model e exibe uma view apropriada; O controller possui como funções: ativar processos de negócios, mudar o estado do model, determinar a view apropriada de acordo com as ações dos usuários e com os resultados das mudanças no model (Duarte, 2011; Margam, 2018).

A figura 1 abaixo apresenta como as três camadas (Model, View e Controller) interagem entre si e o usuário, no qual o usuário realiza requests para o controller, o controller busca e atualiza informações do model e também envia dados para a view, por fim, a view apresenta a resposta para o usuário (Margam, 2018).

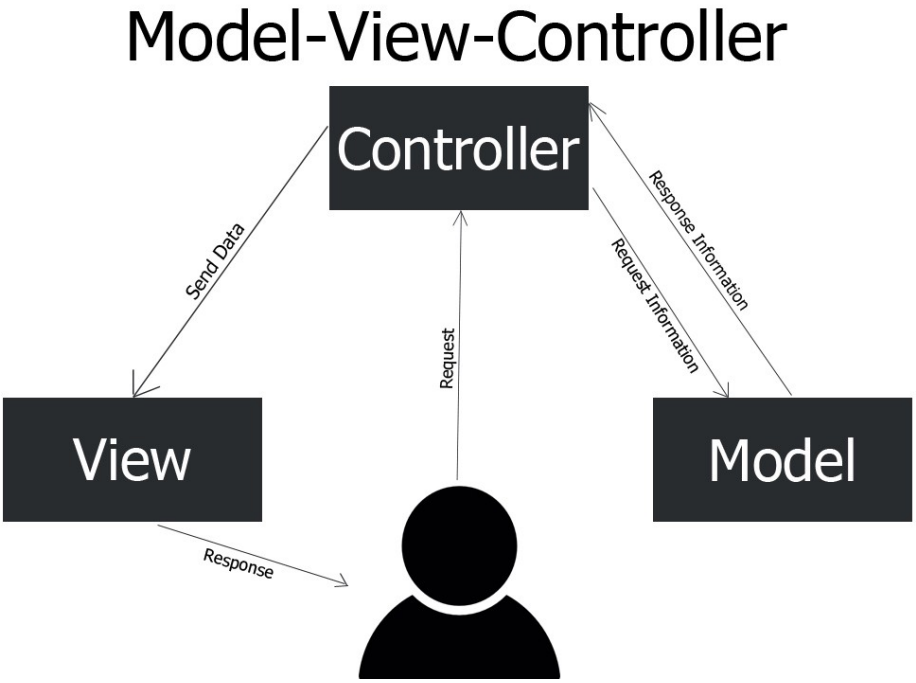

Figura 1. Relacionamento entre as camadas da arquitetura MVC (Margam, 2018)

O MVC realiza um desacoplamento entre as views e os models. No qual, a view deve assegurar que sua aparência reflita o estado do model, sempre que houver alterações nos dados do model, a view, que é dependente dos dados, recebe uma notificação para que possa se atualizar. Com isso, é possível que um mesmo model possua diferentes views sem a necessidade de sua alteração, permitindo views com diferentes tratamentos em sua interface (Duarte, 2011).

A view não é necessariamente uma tela de usuário, entretanto é uma interface de comunicação entre o sistema e seu cliente, podendo ser uma tela ou outro sistema (Duarte, 2011).

3. Laravel

O Laravel é um framework utilizado na linguagem PHP para agilizar o processo de desenvolvimento e construção de sistemas web. Uma de suas principais características é agilidade que ele dá ao seu projeto por conta da sua forma estruturada, fazendo com que seja um dos frameworks mais utilizados atualmente, como mostrado na figura 2 abaixo.

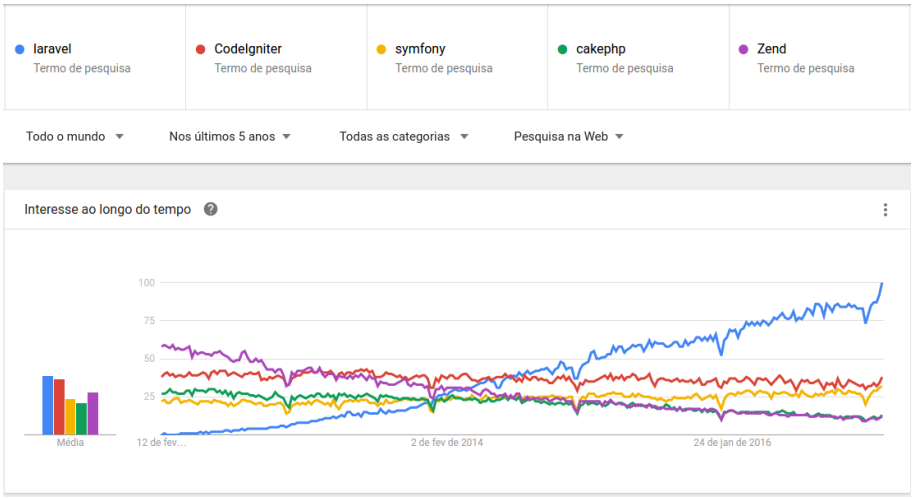

Figura 2. Ascensão do Laravel segundo o Google Trends. Fonte: `https://medium.com/joaorobertopb/o-que-%C3%A9-laravel-porque-us%C3%A1-lo-955c95d2453d`

A popularidade do Laravel também se dá pelo uso de facilitadores que estão integrados no framework, entre os principais facilitadores podemos destacar:

- Composer: o composer é um gerenciador de dependências, facilitando muito o gerenciamento de pacotes de terceiros dentro da aplicação;
- Sistemas de rotas: o framework simplifica o sistema de rotas, fazendo com que seja mais fácil e prático para o usuário implementar as rotas. Essas rotas mapeiam a URL digitada e encaminham para alguma ação dentro da aplicação;
- Blade: O blade é um template engine, ele age reduzindo a quantidade de código inserido no HTML da página, aumentando assim o reuso de código. Entre os principais ganhos de se usar o blade, podemos destacar as seções e a herança, benefícios esses que facilitam a implementação do conceito de máster page.

4. Desenvolvimento

O site foi desenvolvido no formato MVC disponibilizado pelo próprio Laravel. Entretanto, como o foco do desenvolvimento foi o front-end, não é realizado nenhuma operação com o banco de dados, apenas é realizado a comunicação com uma API por meio de requests com o recebimento e envio de arquivos json.

Como todas as operações com o banco de dados são realizados a partir da API, o model não é necessário, neste caso o controller é o responsável por realizar a comunicação com a API e enviar os dados para a view.

A figura 3 abaixo apresenta a camada controller criada com os respectivos arquivos utilizados para o desenvolvimento do site. No Laravel, o controller se encontra em '/app/Http/Controllers'.

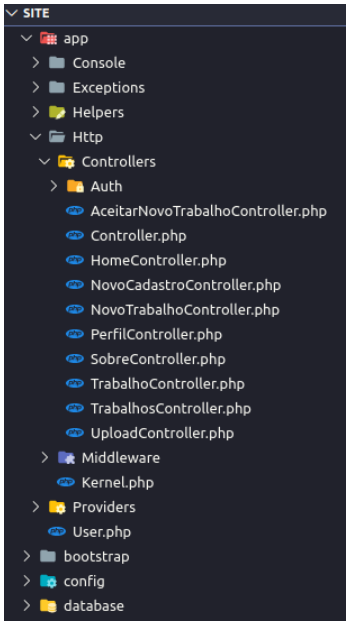

Figura 3. Camada controller

A figura 4 abaixo apresenta a camada view criada com os respectivos arquivos utilizados para o desenvolvimento do site. No Laravel, o controller se encontra em '/resources/views'.

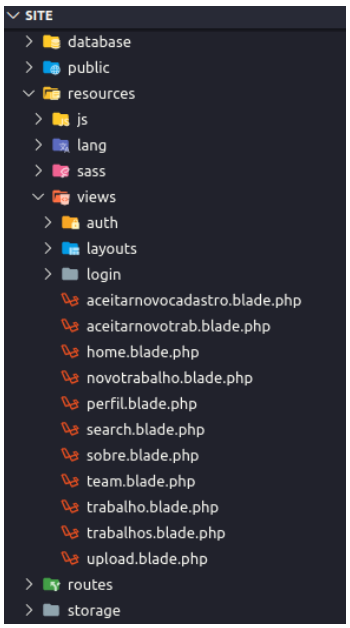

Figura 4. Camada view

Todos as páginas e funcionalidades podem ser acessadas por meio da utilização de rotas amigáveis, as rotas desenvolvidas podem ser visualizadas abaixo.

Como o site ainda não se encontra disponível na web, apenas localmente, as rotas serão apresentadas no formato do localhost.

1. Rota da página principal:
   localhost/

2. Rota da página sobre:
   localhost/sobre

3. Rota da página de visualização de um trabalho (id representa o id do trabalho a ser visualizado):
   localhost/trabalho/{*id*}

4. Rota da página de upload de um trabalho:
   localhost/upload

5. Rota da página de visualização do perfil pessoal:
   localhost/upload/perfil

6. Rota da página de visualização do trabalhos pessoais:
   localhost/upload/trabalhos

7. Rota da página de upload de um novo trabalho:
   localhost/upload/novotrabalho

8. Rota da página para aceitar um novo usuário:
   localhost/upload/novocadastro

9. Rota da página para aceitar um novo trabalho:
   localhost/upload/aceitarnovotrabalho

5. Resultados

Abaixo é apresentado as principais páginas construídas com uma breve explicação.

A figura 5 abaixo apresenta a tela home do iFBox. No qual, possui um menu com as opções para 'Pesquisar', 'Sobre' nós e 'A equipe'. O menu também possui os botões para realizar o login no site e também para realizar o cadastro no site, possui um o botão para acessar o GitHub do trabalho, outro botão para realizar o upload de um trabalho no repositório e outro botão para acessar a dashboard do repositório de trabalhos.

Além do menu, a figura 5 abaixo também apresenta o barra de pesquisa de trabalhos no repositório e logo abaixo apresenta o objetivo e uma breve explicação do repositório de trabalhos iFBox.

Figura 5. Página home - Menu, barra de pesquisa e sobre

A figura 6 abaixo é uma continuação da página home do site. Nela é apresentado o time de desenvolvimento do site e se encontra localizada logo abaixo das informações sobre o site.

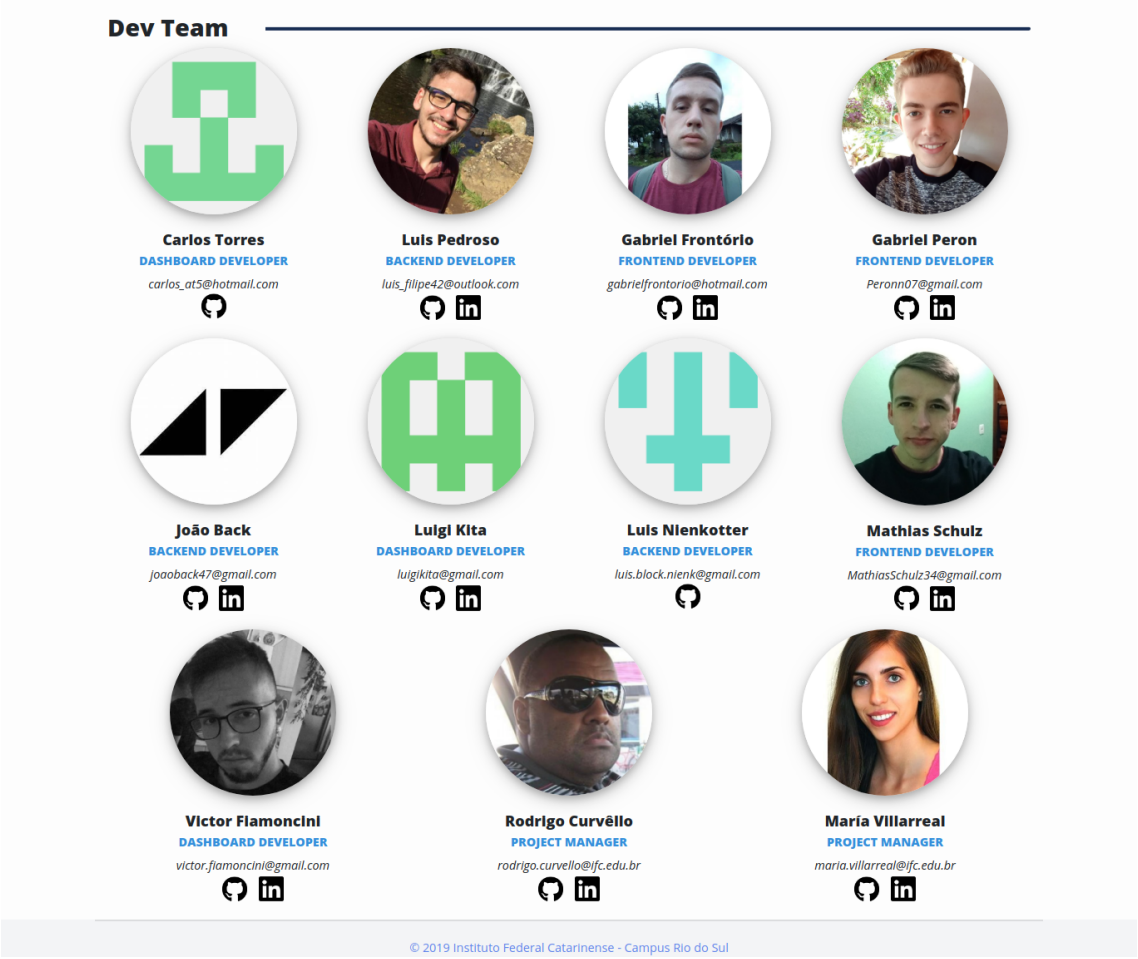

Figura 6. Página home - Time

A figura 7 abaixo apresenta a tela de visualização de um trabalho, com as respectivas informações sobre o trabalho.

Figura 7. Página visualizar um trabalho

A figura 8 abaixo apresenta a tela com o perfil do usuário logado no site, apresenta as informações pessoais do usuário e também permite que alguns campos sejam alterados, os campos que podem ser alterados possuem o fundo branco e os campos não podem ser alterados possuem o campo cinza. Além disso, também permite e alteração da foto de perfil.

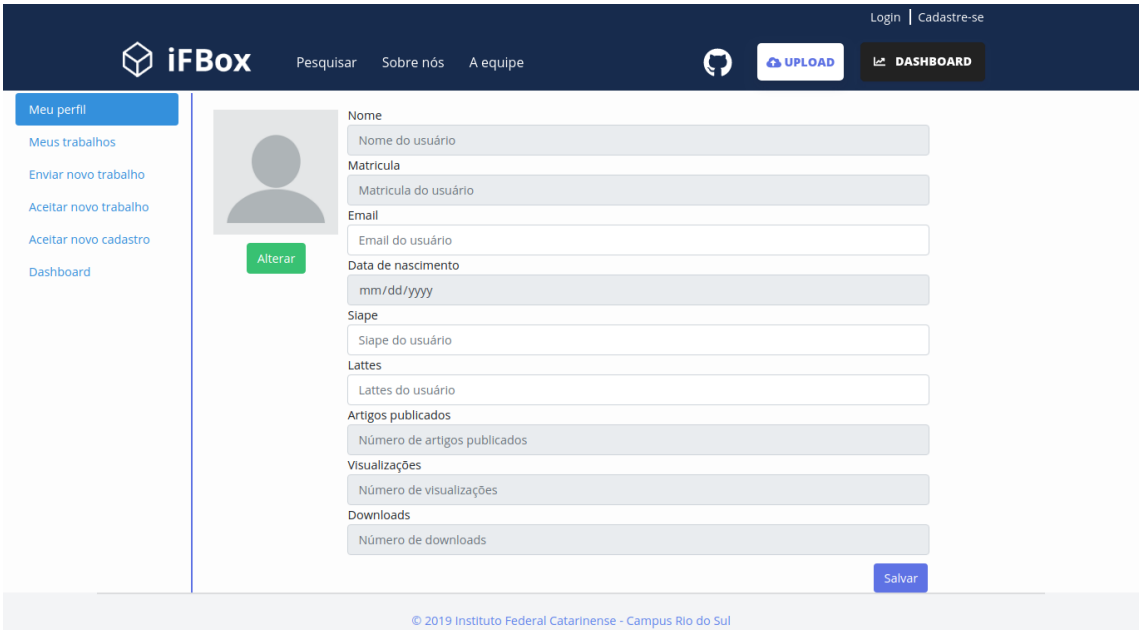

Figura 8. Página de visualização do perfil pessoal

Além da visualização do perfil pessoal, a página da figura 8 acima também possui o menu lateral para acessar a página de trabalhos já postados pelo usuário, a página de envio de um novo trabalho, a página para aceitar um novo trabalho, a página para aceitar um novo cadastro e, por último, o acesso a dashboard.

6. Conclusão

O site desenvolvido é uma ótima forma de compartilhamento de trabalhos, permitindo que os usuários postem e visualizem trabalhos de uma maneira simples, rápida e fácil.

Embora o tempo para desenvolvimento tenha sido curto, o site foi concluído com sucesso e apresenta todos os requisitos esperados e necessários para seu funcionamento. Assim como, permite que qualquer pessoa acesse o site para visualização dos trabalhos disponibilizados.

7. Referências

ALMEIDA, Rodrigo Rebouças de; **Model-View-Controller (MVC)**; Disponível em: `http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm`; Acesso em: 04/12/2019 10:00;

DUARTE, Aldrey Rocha; **METODOLOGIA RAILS: ANÁLISE DA ARQUITETURA MODEL VIEWCONTROLLER APLICADA**; Disponível em: `https://repositorio.ufmg.br/bitstream/1843/BUOS-94MMY9/1/andreyrochaduarte.pdf`; Acesso em: 04/12/2019 09:43;

MARGAM, Girish; **Model-View-Controller (MVC)**; Disponível em: `https://medium.com/datadriveninvestor/model-view-controller-mvc-75bcb0103d66`; Acesso em: 04/12/2019 10:25;

ROBERTO, João; **O que é Laravel? Porque usá-lo?**; Disponível em: https://medium.com/joaorobertopb/o-que-%C3%A9-laravel-porque-us%C3%A1-lo-955c95d2453d; Acesso em 04/12/2019 22:30;