PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO ÁGIL DE APLICAÇÕES WEB

ELISEU ALVES RIBEIRO PEREIRA

GOIÂNIA
2023

ELISEU ALVES RIBEIRO PEREIRA

DESENVOLVIMENTO ÁGIL DE APLICAÇÕES WEB

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Aníbal Santos Jukemura

GOIÂNIA
2023

ELISEU ALVES RIBEIRO PEREIRA

DESENVOLVIMENTO ÁGIL DE APLICAÇÕES WEB

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás em 12/06/2023.

_____

Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca Examinadora:

_____

Orientador: Prof. Me. Aníbal Santos Jukemura

_____

Prof. Me. Max Gontijo de Oliveira

_____

Profa. Dra. Carmen Cecília Centeno

GOIÂNIA
2023

À Deus que me proporcionou esta oportunidade.
À minha esposa por todo incentivo e apoio.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus pelas oportunidades e por sempre guiar meus passos.

À minha amada esposa, Camylla Silva Otto, que presenciou meus altos e baixos durante a graduação, e mesmo assim sempre esteve e está presente ao meu lado, com todo o amor, apoio, incentivo, compreensão e companheirismo.

Ao meu pai, Teobaldo Alves Ribeiro (*in memorian*) e à minha mãe, Terezinha Paula Pereira Alves por serem os braços fortes que me apoiaram durante a vida. Ao meu irmão, Lucas Alves Ribeiro por seu companheirismo.

Agradeço aos meus professores e aos examinadores deste trabalho, por todo o conhecimento proporcionado, sempre com dedicação e atenção.

Agradeço o meu professor e orientador, Aníbal Santos Jukemura, pelas riquíssimas orientações e conhecimentos compartilhados, além de toda paciência e dedicação a este trabalho.

Agradeço a todos os professores que contribuíram com a minha formação. Vocês contribuíram imensamente com a minha formação tanto pessoal quanto profissional.

Por fim, agradeço ao meu pastor e a todos os meus irmãos de fé da Igreja Assembleia de Deus – Ministério Jardim América em Bela Vista de Goiás.

RESUMO

Com a crescente demanda pelo rápido desenvolvimento de tecnologias que resolvessem os mais variados problemas, o tempo tornou-se um grande vilão no desenvolvimento de aplicações nas diversas plataformas, seja web, local ou nuvem. A fim de resolver esse problema, com o passar dos anos foram desenvolvidas ferramentas de trabalho (*frameworks*) que já vêm com recursos implementados e, em outra vertente, adotou-se uma nova metodologia de desenvolvimento, isto é, a metodologia ágil. Para este trabalho, foi utilizada a ferramenta *Ruby On Rails* (linguagem + *framework*) e a metodologia ágil *Extreme Programming* (XP), a fim de evidenciar as reais vantagens de desenvolvimento de aplicações *web* utilizando esse *framework* aliado com essa metodologia.

Palavras-chave: Desenvolvimento *web*; Desenvolvimento ágil; *Ruby On Rails;* metodologias ágeis; *Extreme Programming*.

ABSTRACT

With the growing demand for the rapid development of technologies that solve the most varied problems, time has become a great villain in the development of applications on different platforms, whether web, local or cloud. In order to solve this problem, over the years work tools (frameworks) have been developed that already come with implemented resources and, in another aspect, a new development methodology has been adopted, that is, the agile methodology. For this work, the Ruby On Rails tool (language + framework) and the Extreme Programming (XP) agile methodology were used in order to highlight the real advantages of developing web applications using this framework combined with this methodology.

Key Words: Web development; Agile development; Ruby On Rails; agile methodologies; Extreme Programming.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interface*

CEP – Código de Endereçamento Postal

CRUD – *Create, Retrieve, Update e Delete*

DRY – *Don't Repeat Yoursef*

HTML – *HyperText Markup Language*

MVC – *Model-View-Control*

MER – Modelagem de entidade e relacionamento

ORM – *Object Relational Mapper*

RHTML – Rails and HTML

RoR – *Ruby On Rails*

SQL – *Structured Query Language*

XP – *Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

Com a crescente demanda pelo rápido desenvolvimento de tecnologias que resolva problemas das mais diversas áreas, o tempo se tornou um grande vilão no desenvolvimento de aplicações tecnológicas, seja *web*, local ou nuvem.

O desenvolvimento *web* é a criação de aplicações (que dependem da rede de internet para o pleno funcionamento) por meio da programação. As aplicações *web* possuem arquitetura cliente-servidor e possuem integrações com sistemas e ferramentas que podem ou não estar no mesmo local (EDUCAÇÃO, 2022).

Para se construir uma aplicação, é necessário integrar o código *back-end* com o código *front-end.* Muitas das vezes, dependendo do recurso utilizado, essa integração deve ser feita de forma manual. Isso pode demandar muito tempo na construção da aplicação, tanto pelo trabalho gerado pela implementação, quanto pela grande complexidade, uma vez que esta integração é um dos pontos centrais para o funcionamento correto.

Existem diversas opções de metodologias de desenvolvimento, como por exemplo, a cascata, a espiral, a prototipagem, as ágeis, dentre outras. Para este trabalho, será abordada a metodologia ágil, *Extreme Programming* (XP).

Segundo Wildt et al. (2015), a *Extreme Programming* (XP) é uma metodologia ágil voltada para pequenas e médias equipes. Ela possui práticas e valores que apoiam o foco na agilidade das equipes e na satisfação do cliente.

Antes de construir uma aplicação *web*, é necessário também realizar uma análise criteriosa para melhor escolher um *framework* de desenvolvimento da aplicação. Para este trabalho, serão utilizados a linguagem e o *framework full-stack Ruby On Rails* (doravante apresentado apenas por *framework*), que possui várias ferramentas que possibilitam criar aplicativos web (“*Ruby on Rails”,* [s.d.]). A linguagem de programação utilizada no *back-end* pelo *Ruby On Rails é o Ruby* que apresenta características multiparadigma, ´interpretada, de é de tipagem dinâmica e forte, criada por Yukihiro “Matz” Matsumoto em 1995.

O padrão de arquitetura de software abordado neste trabalho será o *Model-View-Controller (MVC)*, criado em 1979 pelo cientista da computação norueguês

Trygve Reenskaug. A arquitetura MVC divide a aplicação em três camadas: manipulação dos dados (*model*), interação do usuário (*view*), camada de controle (*controller*) (Coodesh, 2023).

Portanto, com o *Ruby On Rails* aliado ao *Extreme Programming* (XP), o processo de construção de aplicações *web* se torna relativamente simples e fácil, pois o *framework* integra todas as ferramentas necessárias para a construção de uma aplicação web utilizando o padrão *Model-View-Controller (MVC)* com entregas incrementais.

Para fins de análise, no estudo de caso será apresentada uma aplicação construída com base em todos os conceitos abordados neste trabalho, comprovando as reais vantagens do uso do *Ruby On Rails* juntamente com a metodologia ágil *Extreme Programming* (XP) no desenvolvimento de aplicações web.

1.2 Objetivo geral

Dadas as caraterísticas supracitadas, é o objetivo principal deste trabalho:

- Desenvolver uma aplicação *web* de perguntas e respostas sobre a bíblia evangélica, denominada Fé em Foco, de modo que esse desenvolvimento evidencie as reais vantagens de utilizar o *framework Ruby On Rails* juntamente com a metodologia ágil *Extreme Programming* (XP).

1.3 Objetivos específicos

Considerando o objetivo principal deste trabalho, espera-se também:

- Demonstrar as facilidades de implementações de aplicações web utilizando o padrão MVC;
- Evidenciar as vantagens do *Ruby On Rails* na integração do *front-end* com o *back-end*;
- Apresentar as vantagens de se utilizar a metodologia *Extreme Programming* (XP) na criação de aplicações web.

1.4 Justificativa

Justifica-se estudar este tema, pois o *Ruby On Rails* aliado à metodologia ágil *Extreme Programming* (XP) permite a construção de aplicações web robustas de maneira rápida e simplificada atendendo as expectativas dos clientes.

Diante deste contexto, este trabalho visa responder a seguinte questão de pesquisa: Quais as vantagens de construir aplicações web utilizando o *Ruby On Rails* com a metodologia *Extreme Programming* (XP)?

1.5 Resultados esperados

Espera-se que os resultados deste trabalho possam contribuir para a compreensão das vantagens e facilidades de se utilizar o *Ruby On Rails* com o padrão *MVC* (*Model-View-Controller*) integrando o *front-end* com o *back-end*.

Espera-se também que sejam apresentadas as reais vantagens de se utilizar o *framework Ruby On Rails* juntamente com a metodologia ágil *Extreme Programming* (XP) no desenvolvimento de aplicações *web*.

1.6 Estrutura do trabalho

Este trabalho está organizado como segue: o Capítulo 2 apresenta a metodologia. O Capítulo 3 descreve o estudo de caso. No Capítulo 4, os resultados são apresentados. No Capítulo 5 descreve a conclusão. Por fim, no Capítulo 6 são apresentadas as propostas de trabalhos futuros.

2 METODOLOGIA

Este capítulo apresenta conceitos metodológicos e técnicos deste trabalho. Na seção 2.1 são apresentados os procedimentos metodológicos. Na seção 2.2 são

apresentados, com base na engenharia de *software*, os modelos de desenvolvimento de *software*. Na seção 2.3 são apresentados os métodos ágeis de desenvolvimento de *software*. Na seção 2.4 é apresentada a metodologia ágil *Extreme Programming* (XP). Na seção 2.5 é apresentada a linguagem *Ruby*. Na seção 2.6 é apresentado o *framework Ruby on Rails*. Na seção 2.7 é apresentado o padrão de arquitetura MVC. Por fim, na seção 2.8 são apresentados os trabalhos relacionados.

2.1 Procedimentos metodológicos

Esta seção apresenta os procedimentos metodológicos utilizados nas pesquisas realizadas para construir este trabalho.

Quanto à natureza, essa pesquisa é um resumo de assunto.

Para Wazlawick (2014, p. 21), *"Os resumos de assunto buscam apenas sistematizar uma área de conhecimento, usualmente indicando sua evolução histórica e estado da arte".*

Em relação aos objetivos, foi utilizada a pesquisa exploratória, que para Gil (2010, p. 25):

> Estas pesquisas têm como objetivo proporcionar maior familiaridade com o problema com vistas a tomá-lo mais explícito ou a construir hipóteses. Pode-se dizer que estas pesquisas têm como objetivo principal o aprimoramento de ideias ou a descoberta de intuições. Seu planejamento é, portanto, bastante flexível, de modo que possibilite a consideração dos mais variados aspectos relativos ao fato estudado.

Quanto aos procedimentos técnicos, essa pesquisa é classificada em bibliográfica e documental.

Para Gil (2010, p. 27), *"A pesquisa bibliográfica é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos".*

A pesquisa documental é desenvolvida com base em materiais que não receberam ainda um tratamento analítico, ou que ainda podem ser reelaborados de acordo com os objetos da pesquisa.

> Nem sempre fica clara a distinção entre a pesquisa bibliográfica e a documental, já que, a rigor, as fontes bibliográficas nada mais são do que documentos impressos para determinado público. Além do mais, boa parte das fontes usualmente consultadas nas pesquisas documentais, tais como jornais, boletins e folhetos, pode ser tratada como fontes bibliográficas. Nesse sentido, é possível até mesmo tratar a pesquisa bibliográfica como um tipo de pesquisa documental, que se vale especialmente de material impresso fundamentalmente para fins de leitura (Gil, 2010, p. 51).

2.2 Modelos de desenvolvimento de *software*

Para se desenvolver um software, independente da finalidade deste, é necessário adotar um modelo, uma metodologia e um padrão de desenvolvimento, que serão utilizados em todo o processo, desde o levantamento de requisitos até a entrega e manutenção do *software*.

De acordo com Sommerville (2011), a engenharia de software aborda todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.

Os modelos de *softwares* são, basicamente, representações simplificadas de um processo de *software*. Para Sommerville (2011, p. 19), *"cada modelo representa uma perspectiva particular de um processo e, portanto, fornece informações parciais sobre ele"*.

Ainda segundo Sommerville (2011), os modelos de processo são:

- ***Cascata*:** Modelo que considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, representando cada uma delas como fases distintas.
- ***Incremental*:** Modelo que considera que o sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão adiciona novas funcionalidades à anterior.
- ***Reuso*:** Modelo em que o processo de desenvolvimento do sistema concentra-se na integração de componentes reusáveis em um sistema já existente, em vez de desenvolver um sistema a partir do zero.

Outro ponto fundamental no desenvolvimento do *software* é escolher uma metodologia de desenvolvimento, preferencialmente a que melhor se encaixa no escopo do projeto, seja ela espiral, prototipagem, ágil, dentre outras. Para este trabalho, foi utilizada a metodologia ágil.

2.3 Métodos ágeis de desenvolvimento de software

A metodologias de desenvolvimento, basicamente, são formas de se estruturar o desenvolvimento de *software*.

Até os anos de 1990, havia uma visão generalizada de que o processo de desenvolvimento de *software* devia ser rigoroso e controlado por meio de um planejamento cuidadoso do projeto e da qualidade da segurança formalizada.

Segundo Sommerville (2011), a abordagem pesada de desenvolvimento dirigido a planos era aplicada aos sistemas corporativos de pequeno e médio porte, o *overhead* envolvido era tão grande que dominava o processo de desenvolvimento de software.

Portanto, o tempo gasto em análises de como o sistema devia ser desenvolvido acabava sendo maior do que o tempo gasto em desenvolvimento de programas e testes.

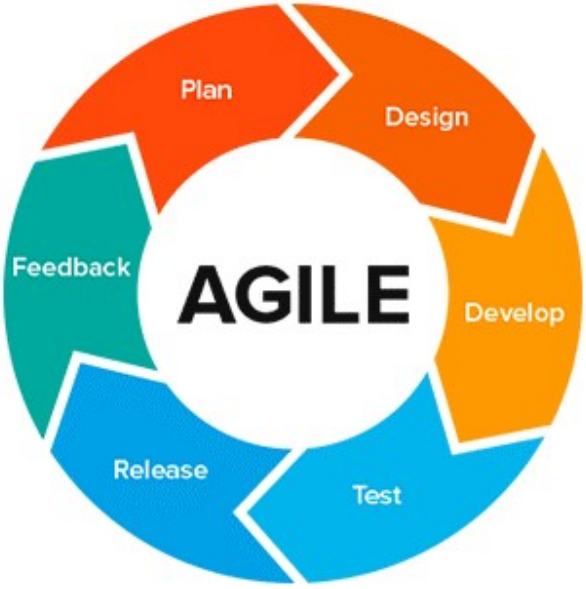
Como os requisitos podem ser alterados durante o desenvolvimento do *software*, é essencial que a especificação e o projeto possam mudar com o programa. Somando também à insatisfação com as abordagens pesadas da engenharia de *software*, na década de 1990 são propostos os métodos ágeis.

> Estes permitiram que a equipe de desenvolvimento focasse no software em si, e não em sua concepção e documentação. Métodos ágeis, universalmente, baseiam-se em uma abordagem incremental para a especificação, o desenvolvimento e a entrega do software. Eles são mais adequados ao desenvolvimento de aplicativos nos quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento. Destinam-se a entregar o software rapidamente aos clientes, em funcionamento, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações posteriores do sistema. Têm como objetivo reduzir a burocracia do processo, evitando qualquer trabalho de valor

> duvidoso de longo prazo e qualquer documentação que provavelmente nunca será usada (Sommerville, 2011, p. 40).

Os métodos ágeis são baseados na noção de desenvolvimento e entrega incremental. A Figura 1 a seguir ilustra de forma resumida, as etapas da metodologia ágil quando aplicada no desenvolvimento de *software*:

Figura 1 - Metodologia ágil



Fonte: TabNews, 2023

A *Extreme Programming* (XP), o *Scrum*, o *Crystal*, o *Adaptative Software Development* e o *Feature Driven Development* são alguns exemplos de métodos ágeis de desenvolvimento. Para este trabalho foi utilizado a metodologia ágil *Extreme Programming* (XP), que será descrita a seguir.

2.4 *Extreme Programming*

Segundo Wildt et al. (2015, p. 16), “O eXtreme Programming é uma metodologia ágil de desenvolvimento de software voltada para times de pequeno a médio porte, no qual os requisitos são vagos e mudam frequentemente”, com foco em agilidade de equipes e na satisfação do cliente, apoiados em valores como simplicidade, comunicação, coragem, respeito e o feedback constante.

> Em *Extreme Programming,* os requisitos são expressos como cenários (chamados de estórias do usuário), que são implementados diretamente como uma série de tarefas. Os programadores trabalham em pares e desenvolvem testes para cada tarefa antes de escreverem o código. Quando o novo código é integrado ao sistema, todos os testes devem ser executados com sucesso. Há um curto intervalo entre os *releases* do sistema (Sommerville, 2011, p. 40).

Ainda, segundo Sommerville (2011, p. 44), *“a Extreme Programming (XP) foi desenvolvida para impulsionar práticas reconhecidamente boas, como o desenvolvimento incremental, a níveis ‘extremos’”.*

Em *Extreme Programming* (XP), novas versões do *software* podem ser construídas diariamente e os *releases* são entregues aos clientes, normalmente a cada duas semanas, mantendo um engajamento contínuo do cliente com a equipe de desenvolvimento.

Quando uma nova versão do *software* é criada, deve-se executar todos os testes para esta nova versão. A nova versão do *software* só será aceita se todos os testes forem executados com êxito. Esta se torna, então, a base para a próxima iteração do sistema.

Mudanças de escopo podem acontecer durante o processo de desenvolvimento do *software* e estas devem ser apresentadas e podem ser aceitas ou não por meio dos *releases* contínuos para os clientes.

Para a *Extreme Programming* (XP), a manutenção é feita por meio da refatoração constante, o que melhora a qualidade do código, fazendo com que o *software* seja fácil de compreender e mais fácil de mudar à medida que novas estórias sejam implementadas.

Em *Extreme Programming* (XP), é adotada a programação em pares, onde o código do sistema possui propriedade coletiva e um processo de desenvolvimento sustentável. De fato, mas sem detrimento às atividades adotadas durante o desenvolvimento deste trabalho, tal prática não foi aplicada, uma vez que o código da aplicação exposta no estudo de caso, foi escrito, sob orientação, de forma individual.

Para um entendimento mais profundo a respeito da metodologia *Extreme Programming* (XP), é necessário compreender seus valores e suas práticas conforme desenvolvido por Kent Beck.

2.4.1 Valores

A metodologia *Extreme Programming* (XP) possui quatro valores. Tais valores são:

- **Feedback:** troca de informações constantes a fim escolher e implementar as melhores funcionalidades do *software* dentro daquela *release*;
- **Comunicação:** entre o cliente e desenvolvedores a fim entender o que precisa ser desenvolvido para definir a complexidade, os detalhes técnicos, os custos, o tempo necessário, dentre outros;
- **Simplicidade:** foco apenas naquilo que é claramente essencial. Ou seja, pequeno escopo de funcionalidades no início de cada iteração para poder implementá-las completamente dentro de um curto prazo de tempo;
- **Coragem:** confiança dos desenvolvedores nos mecanismos de segurança utilizados para proteger o projeto, ajudando a reduzir ou eliminar as consequências de eventuais problemas. Confiança do cliente de que a equipe implementará que foi pedido no início da *release;*

2.4.2 Práticas

Na metodologia *Extreme Programming* (XP) são consideradas as seguintes práticas durante o desenvolvimento:

- **Cliente presente:** interação contínua do cliente com a equipe de desenvolvimento ao longo da implementação;
- **Planejamento:** atividade contínua a ser desempenhada ao longo de todo o projeto a fim de definir o que implementar e quando implementar considerando os *releases* e as iterações;
- **Reuniões diárias:** ocorrem de forma rápida e objetiva. Têm como finalidade apresentar e entender os resultados obtidos no dia anterior, permitindo que os participantes alinhem e priorizem as atividades do dia que se inicia;
- **Programação em Pares:** descreve a atividade de dois programadores trabalhando em conjunto continuamente colaborando no mesmo *design*, algoritmo, código e teste, visando reduzir os riscos de eventuais falhas;
- **Código Coletivo:** em complemento a programação em par, todas as classes e métodos do *software* são visíveis à equipe e qualquer membro da equipe pode melhorar o que for necessário;
- **Código Padronizado:** padrão de código acordado e adotado pelos membros da equipe com o objetivo de simplificar a comunicação, facilitar a programação em par e tornar o código coletivo;
- **Design Simples:** *software* com integridade conceitual e capacidade de evolução, sendo mais genérico e simples de modo a se ajustar mais facilmente a eventuais mudanças. Refatorar o código é uma técnica adotada com objetivo mantê-lo simples ao longo do tempo;
- **Desenvolvimento Orientado a Testes:** técnica preventiva que consiste em escrever um mecanismo de teste automatizado antes de codificar cada história e cada método do sistema com o objetivo de reduzir a incidência de *bugs* e o custo associado à depuração e eliminação dos mesmos. O teste de unidade visa assegurar que cada componente do sistema funcione

corretamente e o teste de aceitação visa assegurar a interação entre os componentes, as quais dão origem às funcionalidades.

Além do modelo e da metodologia de desenvolvimento, é necessário escolher uma linguagem de programação para escrever o *software*. Existem diversas linguagens de programação, cada uma com características específicas. Para este trabalho foi utilizado a linguagem de programação *Ruby*.

2.5 *Ruby*

A linguagem *Ruby* é uma linguagem de programação criada por Yukihiro "Matz" Matsumoto em 1995. *Ruby* é uma linguagem multiparadigma, interpretada e de tipagem dinâmica e forte ("Sobre o Ruby", [s.d.]).

Criada para simplificar o desenvolvimento, a linguagem possui uma sintaxe simples e é inspirada em *Perl*, *Eiffel*, *Ada* e *Lisp*, apresenta recursos de tratamento de exceção como em Java e Python e é completamente orientada a objetos como *SmallTalk* (*About Ruby*, 2022). *Ruby* tem herança única, porém trabalha com o conceito de módulos, que são uma coleção de métodos, sendo possível importá-los. O objetivo desses módulos é diminuir a complexidade que a herança múltipla tem na orientação a objetos (SOUZA, 2013).

Para utilizar a linguagem *Ruby* no desenvolvimento de aplicações *web*, é necessário também utilizar o *framework full-stack Ruby On Rails* (ou *Rails*) que é descrito de forma aprofundada na próxima seção.

2.6 *Ruby On Rails*

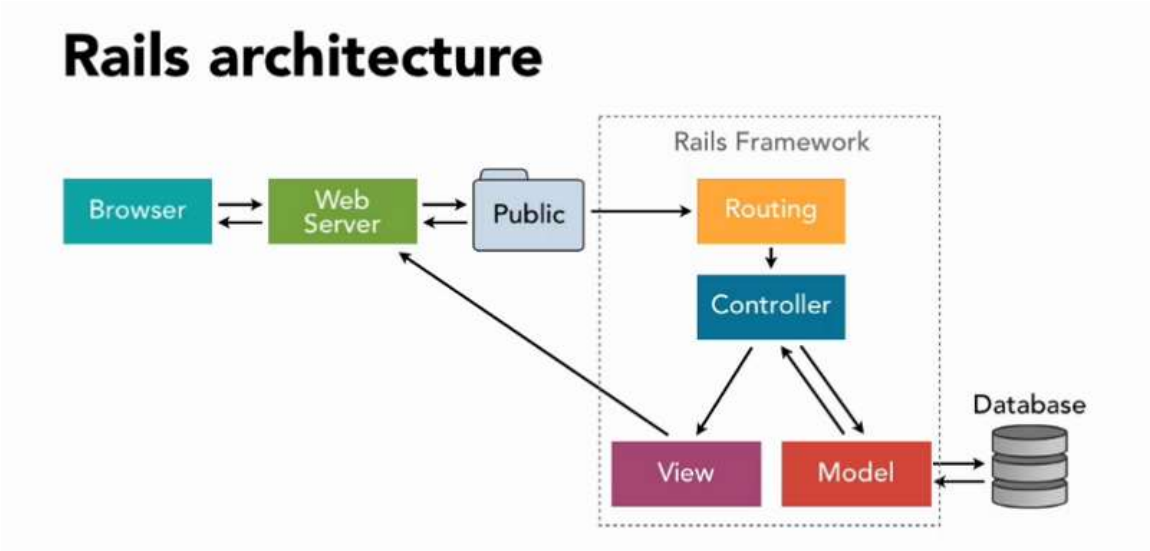
A linguagem *Ruby* possui um *framework full-stack* denominado *Ruby On Rails* que possui várias ferramentas que possibilitam criar aplicativos web ("*Ruby on Rails",* [s.d.]). O *Ruby On Rails* foi criado com o objetivo de proporcionar um desenvolvimento web rápido e estruturado, trazendo recursos pré-configurados e suporte nativo aos principais bancos de dados do mercado e ao *framework JavaScript JQuery.*

O *Rails* é de código fonte aberto, utiliza pacotes integrados de programação e código predefinido, conhecido como convenções sobre configuração (*convention over configuration*), projetado para ser completo e pronto para usar, sem configurações.

Os componentes que compõem o *Rails* são, entre outros, *Action Mailer* (envio de e-mails), *Action Pack* (geração de estrutura *Model-View-Controller* (MVC), *Active Record* (manipulação de banco de dados), *Active Suport* (utilidades gerais como tratamentos de erros e extensões de linguagem) (Rails Guides, 2023). Tais componentes facilitam o desenvolvimento do projeto, principalmente o *Active Record*, por meio do qual é possível gerar um *Create*, *Retrieve*, *Update* e *Delete* (CRUD) que é responsável pela maioria de operações com o banco de dados.

Segue a Figura 2 representando a arquitetura do *framework Rails:* percebe-se uma arquitetura do tipo cliente-servidor, com o cliente acessando um Web Server através de um navegador Web (Browser). O Rails Framework é projetado sobre um padrão de arquitetura MVC, que será descrito no subtópico 2.7, e é responsável pela comunicação direta com o banco de dados (Database).

Figura 2 – Arquitetura Rails



Fonte: FreeCodeCamp, 2018

O *Rails* segue a convenção do *less software* que significa menos código, complexidade e *bugs* e o princípio do DRY (*Don't Repeat Yourself)*, em português, não se repita, que significa que determinada informação deve estar em apenas um lugar no sistema, facilitando a manutenção e o entendimento do código.

Para Loss e Dal Ponte, (2015, p. 15), *"Rails é baseado no padrão de projetos MVC para o desenvolvimento das aplicações e esse padrão é usado para vincular ações do usuário, objetos da aplicação e apresentar informações".* Portanto, na próxima seção, será abordado a arquitetura de *software Model-View-Controller (MVC).*

2.7 Padrão de Arquitetura MVC

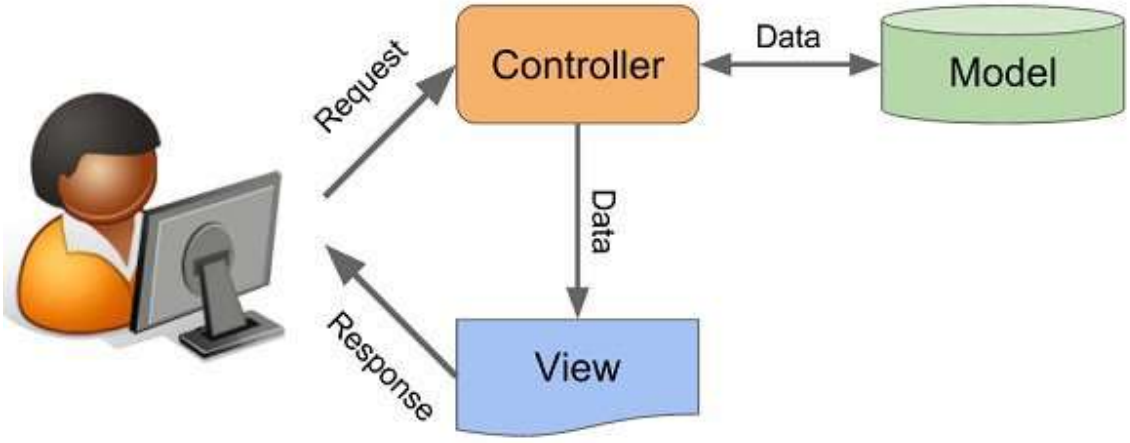
A arquitetura de software define padrões de projeto que podem ser usados em diferentes aplicações.

O padrão de arquitetura de *software Model-View-Controller (MVC)*, foi criado em 1979 pelo cientista da computação norueguês Trygve Reenskaug com o objetivo de separar o projeto em camadas reduzindo assim a dependência entre elas. A arquitetura MVC divide a aplicação em três camadas: manipulação dos dados (*model*), interação do usuário (*view*), camada de controle (*controller*) (SILVA, 2022).

Em uma aplicação que utiliza o MVC, o *Controller* se comunica com a *View* e com o *Model* para gerar a requisição. O *Model* avisa quando as solicitações foram atendidas para que a *View* possa mostrá-las ao usuário em forma de resposta. O detalhe desta arquitetura é que o *Model* nunca se comunica com a *View*, cabendo a função de renderização e entrega ao *Controller*.

Segue na Figura 3, a representação gráfica do padrão de arquitetura MVC:

Figura 3 - MVC



Fonte: Coodesh, 2023

Segundo Loss e Dal Ponte (2015, p. 16):

> Vinculando as funcionalidades do framework Rails com o padrão MVC, qualquer requisição web realizada pelo cliente resulta em uma chamada a algum método em um controle, que por sua vez usa um modelo para realizar os acessos ao banco de dados e retornar para a visão, isto é, o texto da página web como resposta.

Segue a explicação das camadas do MVC de acordo com o seu uso em *Rails*:

- **Model:** O pacote *Active Record* é um mapeador objeto-relacional do *Rails* (define as conexões entre os objetos do domínio e o banco de dados) responsável pelas funções de CRUD[1] que vem do *Action Controller* em instruções *Structured Query Language* (SQL) (Loss; Dal Ponte, 2015);
- **View:** O componente *Action View* do *Rails* apresenta os dados por meio de visões, que representam a aparência visual da aplicação em resposta a requisições no formato *HyperText Markup Language* (HTML). A biblioteca

[1] Segundo Santos (2023), CRUD é um acrônimo para *Create* (criar), *Read* (ler), *Update* (atualizar) e *Delete* (apagar), que se refere as quatro operações básicas que podem ser realizadas em um banco de dados ou em uma aplicação *web*, permitindo que os usuários interajam com os dados armazenados em um sistema, podendo criar novos registros, ler informações existentes, atualizar registros existentes e excluir registros que não são mais necessários.

Active View renderiza *templates* para retornar a resposta para a camada de apresentação. O *template* RHTML (*Rails and HTML*) é uma junção de código Ruby e HTML. Esse *template* coloca o código *Ruby* em *tags* <%...%>. (Loss; Dal Ponte, 2015);

- **Controller:** O *Action Controller* é o responsável pelo controle central da aplicação, atuando como mediador entre os *Models* e a *Views.* Ele recebe as requisições de informações vindas de uma interface *web*, analisa e as decodifica para determinar o que o usuário quer (ou o que é solicitado pela requisição) e então decide qual parte do código da aplicação deve manipular a tarefa. O *Rails* permite aos *controllers* selecionar, condicionalmente, *views* diferentes de acordo com a necessidade dos dados a serem apresentados (Loss; Dal Ponte, 2015).

Após todos os conceitos definidos e expostos, na próxima seção, são apresentadas as referências bibliográficas que foram utilizadas como base para a construção deste trabalho.

2.8 Trabalhos relacionados

Para exemplificar o uso do *Ruby On Rails*, são utilizados o trabalho "Desenvolvimento de um aplicativo *web* para salões de beleza utilizando *Ruby on Rails"* de Loss e Dal Ponte (2015) e o curso "*Ruby On Rails* 5.x - Do início ao fim!" de Pires (2022).

No trabalho de Loss e Dal Ponte (2015), é utilizado o *framework Ruby On Rails* para desenvolver um aplicativo *web* para salões de beleza.

Durante o desenvolvimento, os autores relatam que várias dificuldades foram encontradas no uso da tecnologia (*framework Ruby on Rails*), decorrentes do desconhecimento do seu uso e da relativa falta de material disponível para a versão utilizada.

O objetivo do trabalho foi apresentar a agilidade e a possibilidade de reuso devido os recursos disponíveis no *Ruby On Rails*.

Por fim, os autores concluíram que a vasta gama de recursos disponíveis no *framework Ruby On Rails*, fazem da ferramenta, uma promissora opção para desenvolvimento *web*.

O trabalho de Loss e Dal Ponte (2015) foi relevante, pois se identifica com o propósito deste trabalho no sentido de se utilizar uma tecnologia, que devido aos seus recursos, proporciona maior agilidade no desenvolvimento de aplicações *web*.

No curso "*Ruby On Rails* 5.x - Do início ao fim!" de Pires (2022), é apresentado, em sete seções, o *Ruby* e o *Ruby On Rails.* Durante o curso é apresentado conceitos e conteúdos relacionados ao *Ruby On Rails.* Para exemplificar melhor e aplicar o que é explicado durante as aulas, no decorrer do curso são desenvolvidas duas aplicações.

A aplicação desenvolvida para o estudo de caso (próximo capítulo), a fim de comprovar o que foi proposto no presente trabalho, teve como referência a segunda aplicação do curso "*Ruby On Rails* 5.x - Do início ao fim!" de Pires (2022), na plataforma de aprendizagem *Udemy*.

3 ESTUDO DE CASO

Este capítulo apresenta o estudo de caso deste trabalho. Na seção 3.1 é apresentado o contexto que justifica a escolha da aplicação desenvolvida cujo nome é Fé em Foco. Na seção 3.2 é apresentada a descrição da aplicação. Na seção 3.3 são apresentados os materiais e recursos utilizados ao longo do desenvolvimento da aplicação. Por fim, na seção 3.4 são apresentadas todas as fases do desenvolvimento da aplicação utilizando *Ruby On Rails,* seguindo as práticas e valores da metodologia *Extreme Programming* (XP).

3.1 Contexto

É notório os inúmeros aplicativos ao estilo perguntas e respostas baseada na Bíblia Cristã, para dispositivos móveis, disponíveis para *downloads*. Porém, ao realizar uma breve pesquisa na *internet* por aplicações em português do Brasil de

perguntas e respostas sobre a Bíblia Cristã, o mesmo cenário não é percebido no ambiente *web*.

Para fins de comprovações dos objetivos deste trabalho, foi escolhido a implementação de uma aplicação *web* ao estilo perguntas e respostas com questões baseadas na Bíblia Cristã. Essa aplicação pode ser acessada pelo navegador *web*, e sua descrição é apresentada em detalhes na próxima seção.

3.2 Descrição da aplicação

A aplicação *web,* ao estilo perguntas e respostas possui o nome “Fé em Foco” e as questões são baseadas na Bíblia Cristã. A aplicação possui dois perfis, sendo um de administrador e um de usuário. A aplicação apresenta uma parte pública, visível para todos, onde as pessoas podem responder uma quantidade limitada de perguntas para fins de demonstração, sem acumular pontos e o recurso de pesquisa de palavras presentes nas perguntas.

O administrador tem o controle total da aplicação, podendo gerenciar outros administradores, cadastrar, editar e remover perguntas e/ou assuntos, visualizar a quantidades de perguntas cadastradas, nomes e *e-mails* dos usuários cadastrados.

Para o uso completo por parte do usuário, é necessário realizar o cadastro na aplicação. Após isso, será habilitada a edição das informações do perfil e a visualização detalhada das perguntas já respondidas, podendo saber o total de perguntas respondidas, bem como quantas acertou e quantas errou. Por fim, na página inicial, é visível também para o usuário o ranking contendo o Top 5 de usuários cadastrados na aplicação.

Os usuários cadastrados podem responder todas as perguntas disponíveis, acumulando pontos por questões respondidas corretamente, sendo que a cada questão acertada se soma um ponto, tendo a soma acumulada utilizada para construir e apresentar o *ranking* dos usuários.

Na página inicial da aplicação está presente o recurso de pesquisa de palavras que fazem parte das perguntas disponíveis.

Na próxima seção são expostos os materiais e os recursos que foram necessários para construir a aplicação conforme o que foi descrito.

3.3 Materiais e recursos utilizados

No desenvolvimento da aplicação Fé em Foco, para a modelagem e a implementação foram utilizados os materiais e os recursos descritos na Tabela 1.

Tabela 1 - Materiais e recursos

Nome	**Versão**	**Referência**	**Aplicação**
Ruby On Rails 5.x - Do início ao fim!	-	https://www.udemy.com/course/rubyonrails-5x/	Estruturação da aplicação.
Ruby	2.7.1	https://www.ruby-lang.org/pt/	Linguagem de programação.
Rails	5.2.0	https://rubyonrails.org/	Framework *web* de desenvolvimento.
HTML	5	https://html.spec.whatwg.org/	Desenvolvimento da interface web.
CSS	3	https://www.w3.org/Style/CSS/	Desenvolvimento da interface web.
Bootstrap	3.3.7	https://blog.getbootstrap.com/	Desenvolvimento da interface web.
Node JS	18.13.0	https://nodejs.org/pt-br	Ambiente de execução *JavaScript*
NPM	9.3.1	https://www.npmjs.com/	Gerenciador de pacotes.
Git Bash	2.37.3	https://git-scm.com/	Emulador de terminal *Unix*.
Yarn	1.22.19	https://classic.yarnpkg.com/lang/en/	Gerenciador de pacotes.
Sqlite3	0.14.1		Banco de dados relacional.
VS Code	1.78.0	https://visualstudio.microsoft.com/pt-br/	Editor de código-fonte.

Nome	Versão	Referência	Aplicação
Gentelella Alela Admin	1.4.0	https://github.com/ColorlibHQ/gentelella	Tema do perfil do Administrador.
SB Admin 2	3.3.7+1	https://startbootstrap.com/template-overviews/sb-admin-2/	Tema do perfil do Usuário.
Javascript	*ECMAScript* 2022	https://brasil.js.org/	Linguagem de programação.
Jquery	3.3.1	https://blog.jquery.com/2018/01/20/jquery-3-3-1-fixed-dependencies-in-release-tag/	Biblioteca.
BrModelo	3.31	https://www.brmodeloweb.com/lang/pt-br/index.html	Modelagem do banco de dados.
Astah UML	9.0.0/1778f1	https://astah.net/products/astah-uml/	Diagrama de caso de uso, diagrama de classe e diagrama de atividade.

Fonte: elaborado pelo autor (2023)

Para a modelagem dos diagramas, conforme exposto na Tabela 1, foi utilizado o software *Astah UML* com a licença de estudante. Já para a modelagem conceitual do banco de dados, conforme exposto acima, foi utilizado o software BrModelo. Por fim, para a codificação da aplicação, foi utilizado o editor de código-fonte *VS Code*.

Isto posto, na próxima seção são explicadas todas as etapas do desenvolvimento da aplicação utilizando a metodologia *Extreme Programming* (XP).

3.4 Desenvolvimento da aplicação utilizando *Extreme Programming* (XP)

O desenvolvimento da aplicação Fé em Foco foi iniciado em março de 2023 e concluído em maio de 2023, sendo utilizados todos os recursos descritos na seção anterior, contendo como metodologia ágil a *Extreme Programming (XP),* com ênfase em seus valores e práticas sugeridas.

O projeto como um todo teve o planejamento definido de modo que o desenvolvimento fosse divido em pequenas *Releases*.

As *Releases* foram guiadas pelo prazo, visto que o objetivo era a construção rápida, simples e funcional da aplicação Fé em Foco que satisfizesse o escopo descrito na Seção 3.2. Juntamente com o objetivo definido, foi planejado e definido também os testes que validaram o seu funcionamento.

Foi definido, de acordo com o escopo do desenvolvimento planejado, que cada *Release* teria um curto prazo de duração, durando uma a duas semanas cada.

No final de cada *Release*, foram entregues novas funcionalidades ou melhorias que agregavam valores, isto é, com foco nos objetivos esperados, que por sua vez, tornaram-se perceptíveis para o usuário. Rapidamente, o escopo da próxima *Release* já era descrito, combinando prioridades e estimativas, sempre mantendo o planejamento constante.

A seguir, será visto detalhadamente, como foi todo o desenvolvimento do projeto, desde a definição, estruturação e implementação da aplicação Fé em Foco.

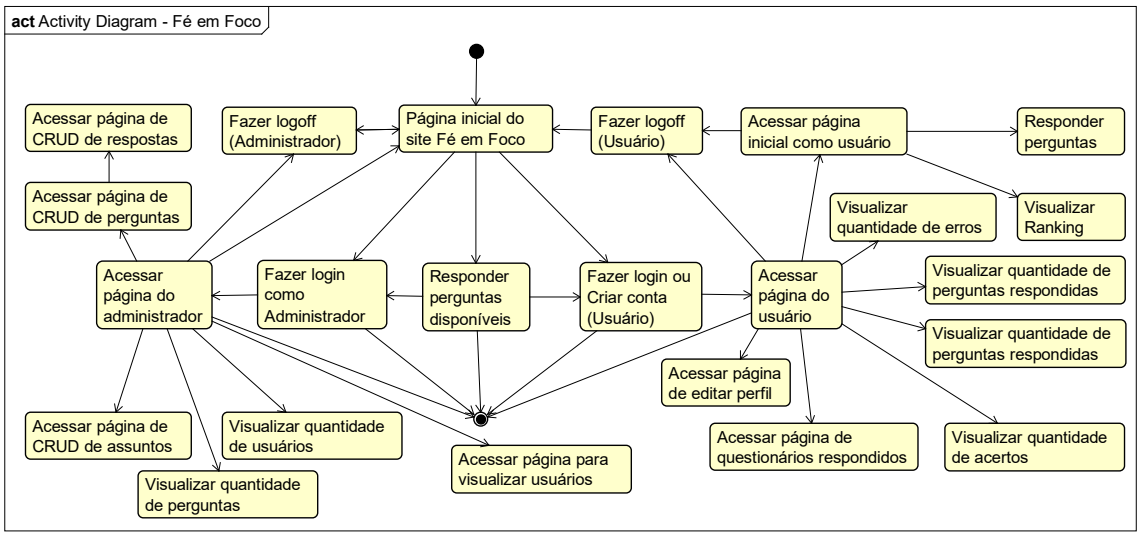
3.4.1 Fase de Planejamento: *Release 1*

A primeira *Release* teve a duração de uma semana sendo definido que deveriam ser desenvolvidos e entregues o diagrama de atividades e o diagrama de caso de uso.

Os diagramas foram desenvolvidos utilizando o software *Astah UML* na versão 9.0.0/1778f1 com a licença de estudante.

A Figura 4 mostra o diagrama de atividade da aplicação Fé em Foco. Pode-se observar, que acessando a página inicial do site Fé em Foco, o usuário pode visualizar o seu conteúdo e as opções de *login*, seja de Administrador ou Usuário. Uma vez logado, caso seja em modo administrador, ficam disponíveis na página inicial uma série de opções, funcionalidades que vão desde o gerenciamento de outros administradores, até o gerenciamento de assuntos e perguntas e a visualização dos usuários cadastrados. Logado em modo usuário, também fica disponível na página inicial do usuário opções limitadas que se restringem ao gerenciamento do próprio perfil voltado para responder perguntas e para visualizar histórico das perguntas respondidas.

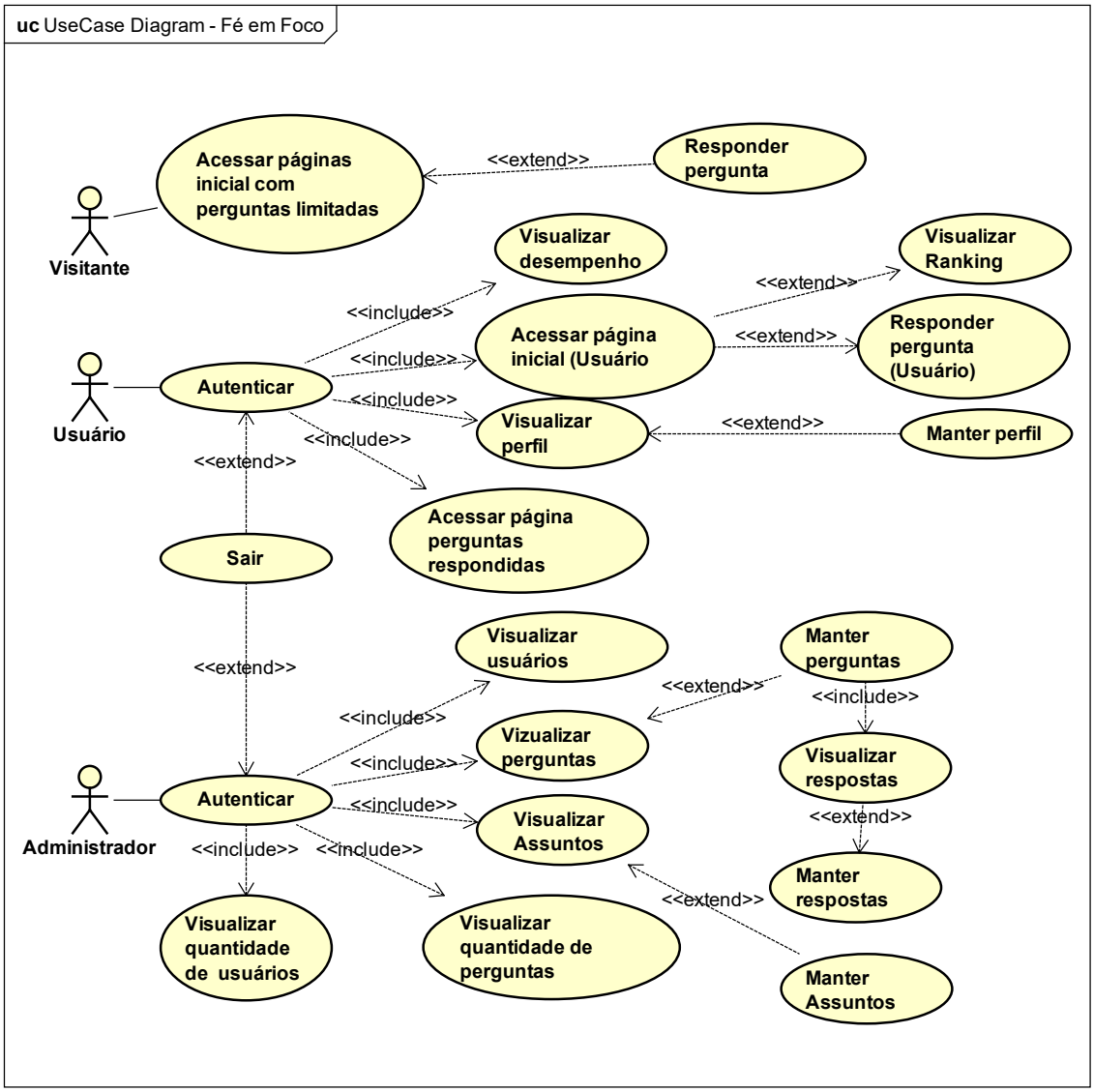
Figura 4 - Diagrama de Atividade



Fonte: elaborado pelo autor (2023)

A Figura 5 mostra o diagrama de caso de uso da aplicação Fé em Foco, onde cada ator terá suas ações disponíveis para atuar, baseadas no diagrama de atividades, apresentado anteriormente na Figura 4.

Figura 5 - Diagrama de Caso de Uso



Fonte: elaborado pelo autor (2023)

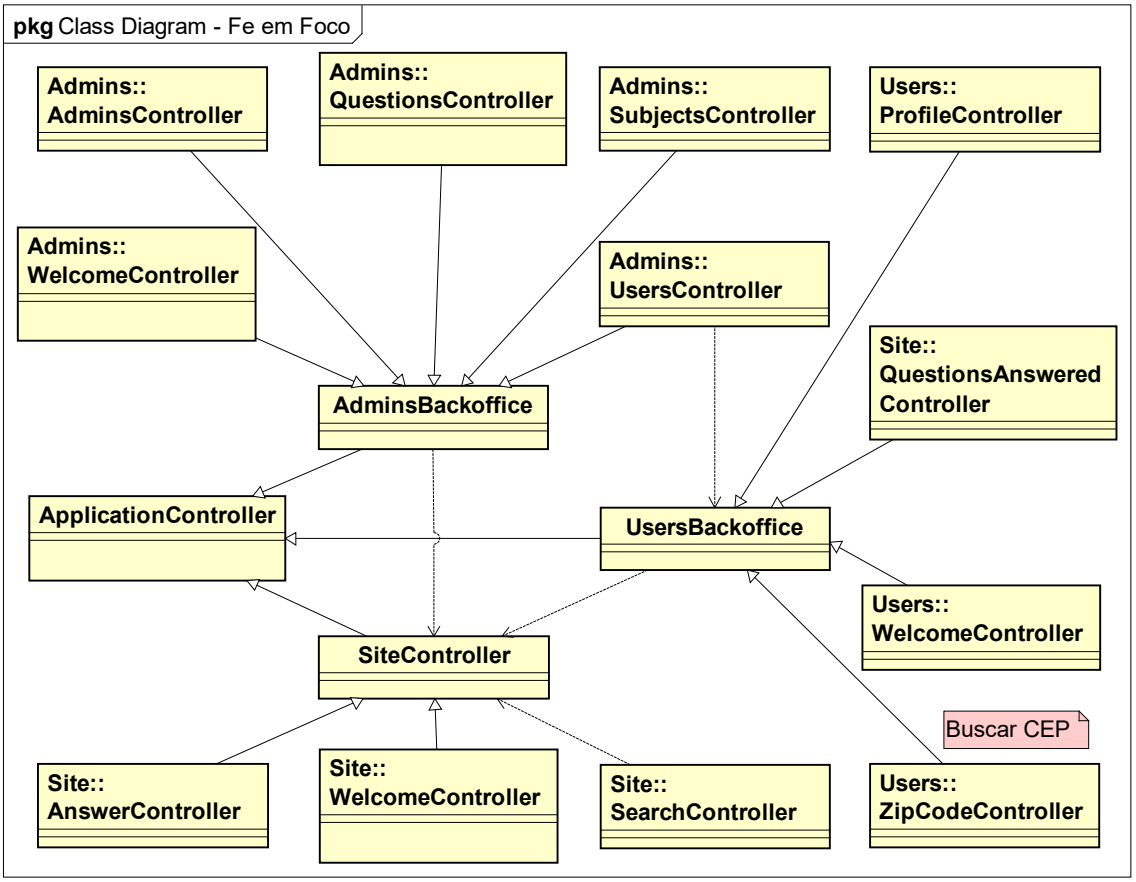
3.4.2 Fase de Planejamento: *Release 2*

A segunda *Release* teve a duração de uma semana, sendo definido que deveriam ser desenvolvidos e entregues o diagrama de classes e a modelagem de entidade e relacionamento (MER) do banco de dados.

A Figura 6 mostra o diagrama de classes da aplicação Fé em Foco, onde, as classes *AdminsBackoffice*, *UsersBackoffice* e *Site* herdarão da classe principal

chamada *ApllicationController*. Da mesma forma, as classes *AdminsBackoffice*, *UsersBackoffice* e *Site* terão suas subclasses, tornando possível a organização da aplicação, baseada na descrição da seção 3.2 e do diagrama de atividades, exposto na Figura 4.

Figura 6 - Diagrama de classe

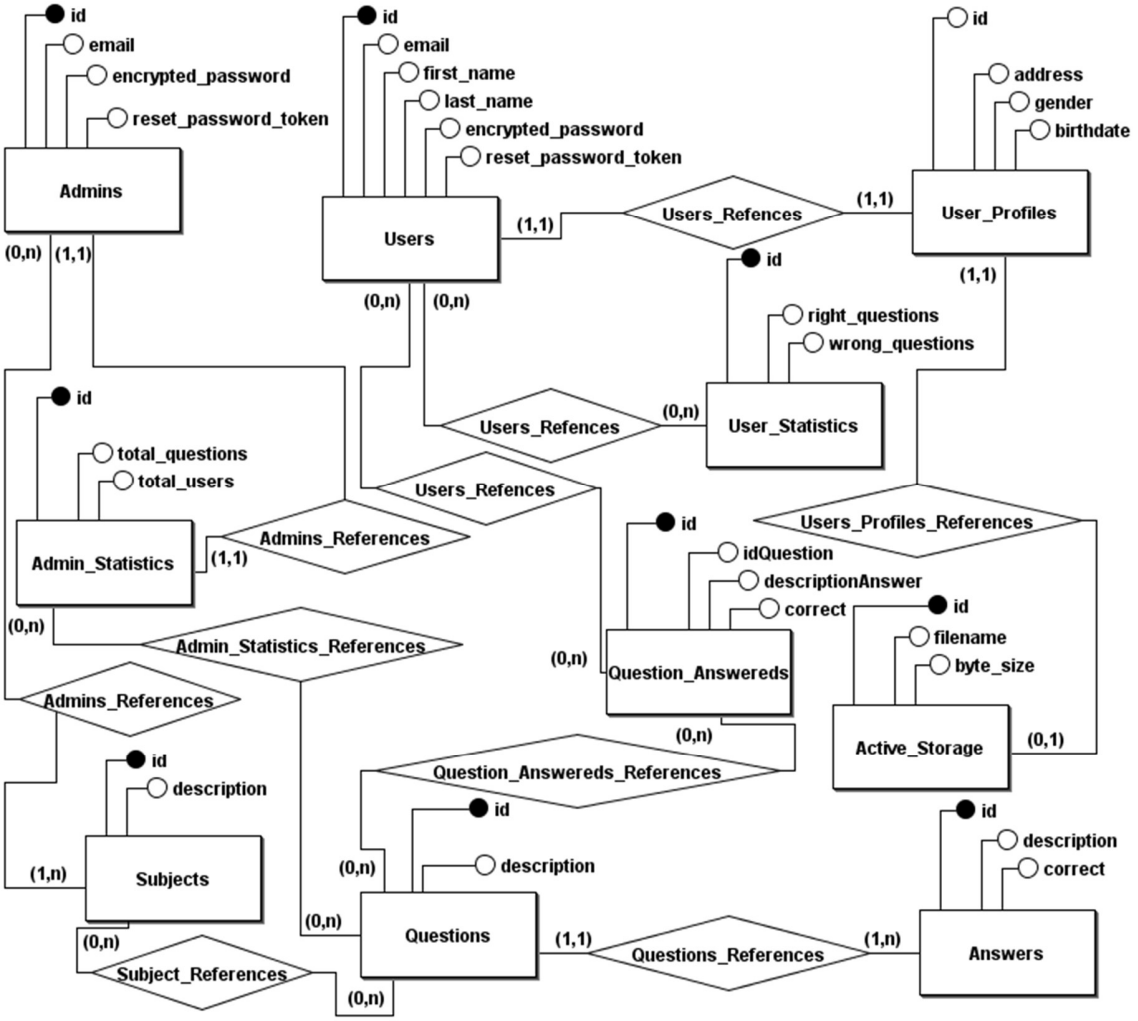


Fonte: elaborado pelo autor (2023)

Quanto ao banco de dados, o *Rails* utiliza o *Object Relational Mapper* (ORM), que é uma técnica de mapeamento de objeto relacional na qual as tabelas do banco de dados são representadas através de classes e os registros de cada tabela são representados como instâncias das classes correspondentes. A grande vantagem da técnica ORM (*Object Relational Mapper*) é que possível utilizar as funções de CRUD (*Create*, *Read*, *Update* e *Delete)* sem escrever instruções SQL (*Structured Query Language)* diretamente e com menos código geral de acesso ao banco de

dados. A Figura 7 mostra a modelagem do banco de dados da aplicação Fé em Foco, com notação entidade-relacionamento, executada no software brModelo. Nela, é possível identificar as entidades e os relacionamentos necessários para o funcionamento da aplicação utilizando o banco de dados *Sqlite3*, escolhido por ser mais prático e acessível. Mas vale ressaltar que nada impede o uso de outros bancos de dados como PostgreSQL, MariaDB ou MySQL, por exemplo. A entidade *Admins* pode se relacionar com as entidades *Admins_Statistics* e *Subjects*. A entidade *Admins_Statistics* pode se relacionar com a entidade *Questions* para construir as estatísticas. A entidade *Subjects* também pode se relacionar com a entidade *Questions*. E a entidade *Questions* pode se relacionar com a entidade *Answers*, para construir as respostas ligadas à pergunta. A entidade *Users* pode se relacionar com as entidades *Users_Profiles*, *Users_Statistics* e *Questions_Answereds*. A entidade *Users_Profiles* se relaciona com a entidade *Active_storage*, com a finalidade de permitir o gerenciamento de foto do perfil do usuário. A entidade *Questions_Answereds* se relaciona com a entidade *Questions* para gerar os relatórios do usuário.

Figura 7 - Modelagem de entidade e relacionamento (MER)



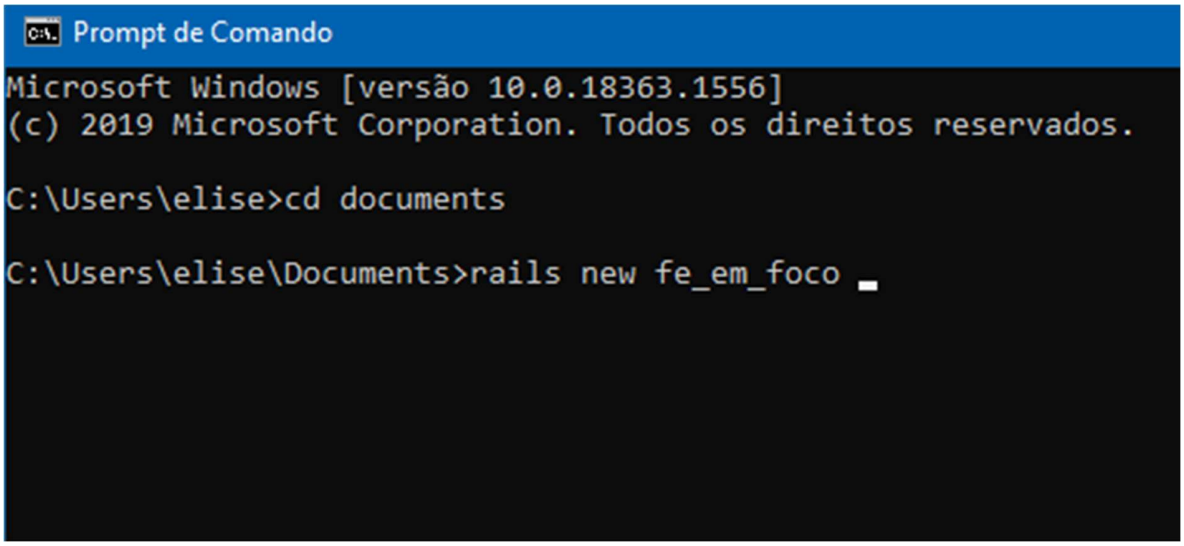
Fonte: elaborado pelo autor (2023)

3.4.3 Fase de Planejamento: *Release 3*

A terceira *Release* teve a duração de uma semana e nela, foi criada a aplicação. No *prompt* de comando do *Windows* 10 (Sistema Operacional utilizado), foi selecionada uma pasta base, e através de um simples comando `rails new fe_em_foco` foi criada toda a estrutura base da aplicação, isto é, todos os diretórios, arquivos de configuração e de dependências que o sistema precisa para ser executado. Uma das grandes vantagens do *Rails* é que com poucos comandos são criadas estruturas complexas que podem ser utilizadas e adaptadas conforme

a necessidade, ganhando bastante tempo de desenvolvimento. Tal facilidade é um diferencial que evidencia as reais vantagens de se utilizar o *Ruby On Rails* para o desenvolvimento de aplicações *web*. O comando `rails new fe_em_foco` é exposto na Figura 8 a seguir:

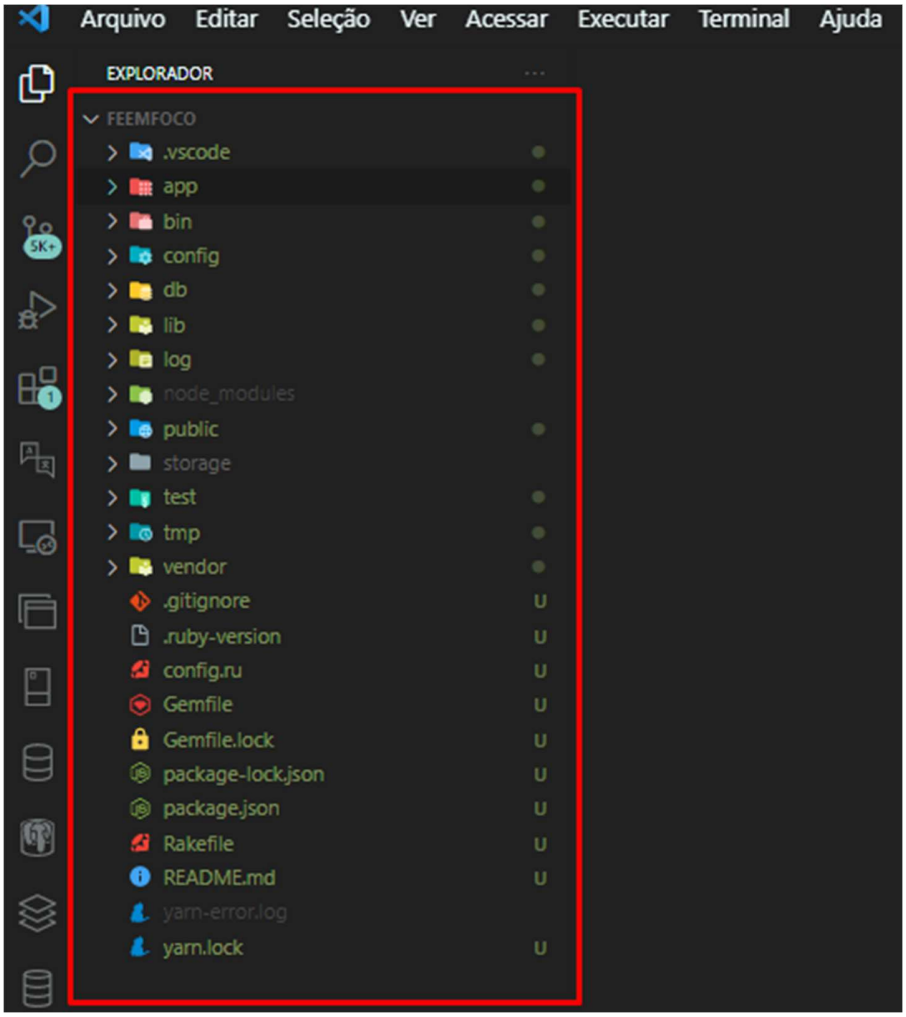
Figura 8 - Criação da aplicação



Fonte: elaborado pelo autor (2023)

Após a criação da estrutura da aplicação, foi aberto o projeto no editor de códigos *VS Code*, para dar início ao desenvolvimento. Na Figura 9 a seguir, pode-se observar na área destacada, toda a estrutura de pastas e arquivos da aplicação Fé em Foco.

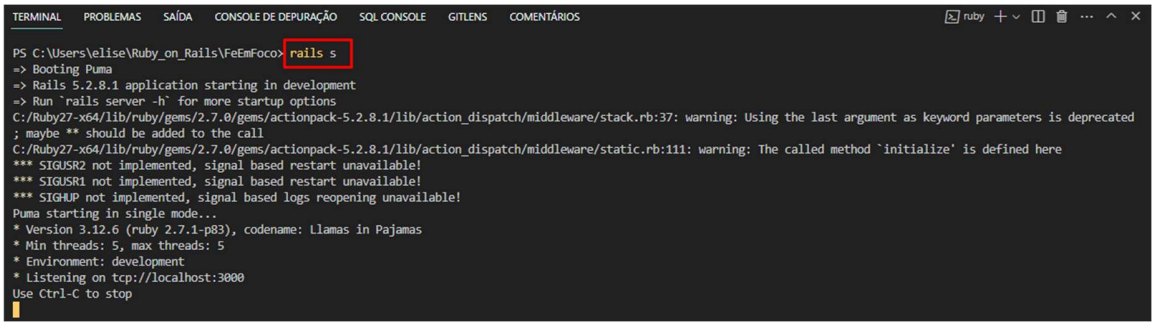
Figura 9 - Estrutura de pastas e arquivos da aplicação



Fonte: elaborado pelo autor (2023)

Uma outra grande vantagem do *Ruby On Rails* é que o *framework* possui o servidor Puma integrado, não precisando instalá-lo à parte. É tão simples, ao ponto de, após a criação do projeto como mencionado anteriormente, apenas abrir um terminal no *VS Code*, apontar para a pasta do projeto e digitar o comando `rails s` que a aplicação já é executada, ficando disponível localmente para acesso no navegador de internet através do endereço `http://localhost:3000/`. A Figura 10 mostra em detalhes o que foi descrito:

Figura 10 - Iniciando a aplicação



Fonte: elaborado pelo autor (2023)

Posteriormente, foram desenvolvidas as funcionalidades básicas da página inicial da aplicação, onde, com o *Rails*, tornou-se possível mesclar código *Ruby* com as demais linguagens utilizando o comando `<%  %>` e, para imprimir na tela, o comando `<%=  %>`. A Figura 11 mostra página inicial da aplicação contendo apenas as funcionalidades básicas da aplicação, com todos os elementos portados do *Bootstrap* 5:

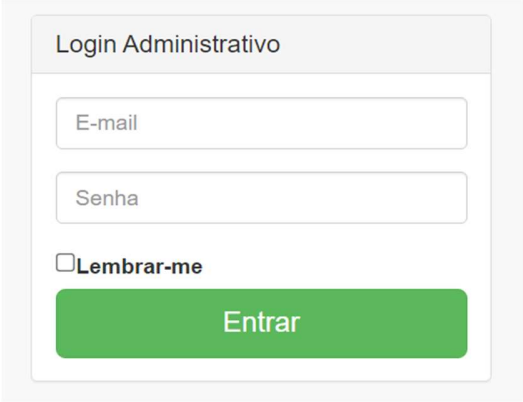
Figura 11 - Página inicial do site

Fonte: elaborado pelo autor (2023)

Nesta *Release* foi criada a estrutura de *logins*. Para isso, foi necessário utilizar a *gem Devise*, que cria toda a estrutura de segurança tanto para o Administrador (Admin) quanto para o Usuário (User) com o comando `rails g devise Admin` e `rails g devise User`. Essa *gem* é um grande diferencial pois ela gera, na parte de segurança, os perfis desejados com *logins*, recuperação

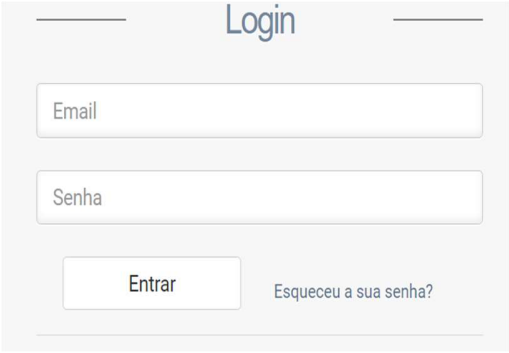
e gerenciamento de senhas, apenas com os comandos mencionados anteriormente. Nota-se mais uma das grandes vantagens dos *Rails*. As figuras 12 e 13 mostram as áreas de *logins* para administrador e usuário, respectivamente, depois de serem customizadas graficamente:

Figura 12 - Login do Administrador



Fonte: elaborado pelo autor (2023)

Figura 13 - Login do Usuário



Fonte: elaborado pelo autor (2023)

3.4.4 Fase de Planejamento: *Release 4*

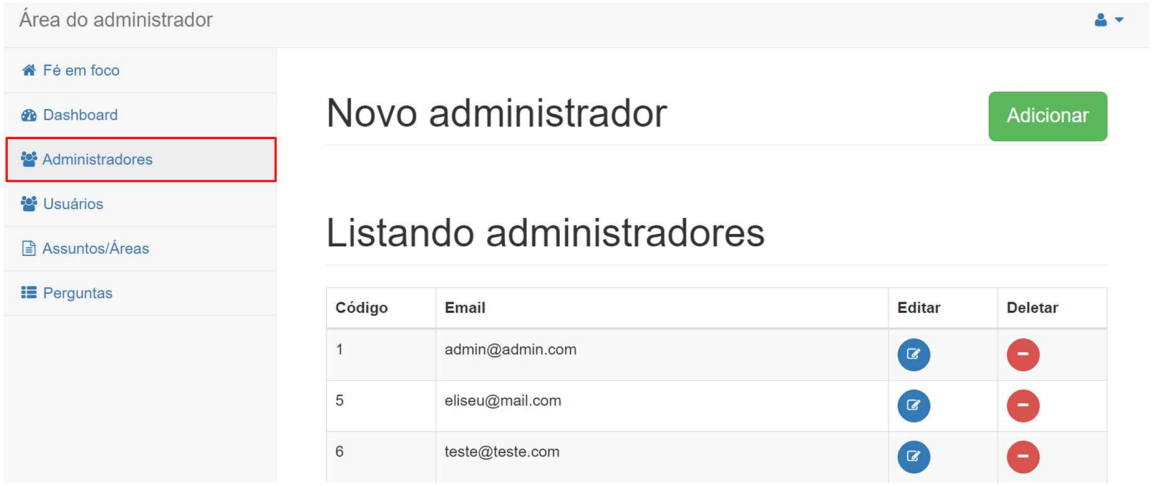
A quarta *Release* teve a duração de duas semanas, devido à complexidade do que foi definido. Nesta *Release* foi desenvolvida a área administrativa, onde o Administrador pode gerenciar a aplicação. Para a área administrativa foi utilizado o *template Bootstrap SB Admin 2* (Start Bootstrap, [s.d.]). Na Figura 14, é mostrada a página inicial do administrador onde é possível acessar na barra lateral, as opções disponíveis, e, na página inicial da aplicação, chamada "*Dashboard*", é possível acompanhar, via *cards*, o total de usuários e o total de perguntas cadastradas no site:

Figura 14 – Página inicial do Administrador

Fonte: elaborado pelo autor (2023)

Após a implementação da página inicial, foi desenvolvida a página de gerência de administradores, chamada "Administradores", bem como suas funcionalidades, onde o administrador principal pode criar, atualizar, visualizar e deletar outros administradores, ou seja, uma estrutura completa de CRUD (*Create, Read, Update, Delete*), que pode ser gerada **facilmente** pelo comando *scaffolding*, conforme a Figura 15 a seguir:

Figura 15 – Página de outros Administradores



Fonte: elaborado pelo autor (2023)

Por fim, nesta *Release*, foi desenvolvida a página de Assuntos e seus recursos, onde o administrador pode criar, visualizar, atualizar e deletar os assuntos para novas perguntas. A seguir, a Figura 16 mostra a página de assuntos:

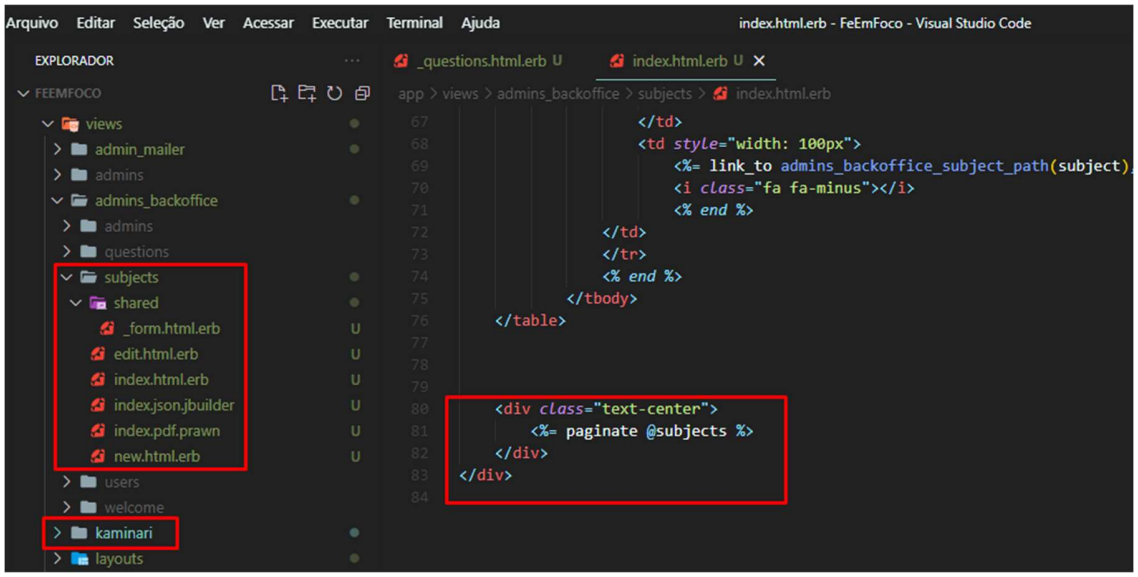
Figura 16 - Página de Assuntos/Áreas

Fonte: elaborado pelo autor (2023)

Novamente, devido aos recursos disponíveis e pelas facilidades da linguagem *Ruby*, com apenas um comando (*scaffolding*) é possível criar uma estrutura completa de CRUD (*Create, Read, Update, Delete*), ficando como trabalho, apenas a customização da página. Na página de assuntos, também foi desenvolvido o recurso de exportar todos os assuntos para um arquivo do tipo PDF (apontado pela seta), caso assim seja necessário.

Uma *gem* bastante útil e que facilita a implementação de paginação, disponível no *Rails* é a *gem Kaminari*. Com ela, é possível criar e usar a paginação de forma bastante facilitada, expondo outra grande vantagem do *Ruby On Rails*. Na Figura 17, é mostrado a implementação da paginação.

Figura 17 - Implementação da gem Kaminari



Fonte: elaborado pelo autor (2023)

Na Figura 18 é mostrado o uso da *gem Kaminari* na página de assuntos, mas o recurso está disponível também em diversas páginas da aplicação.

Figura 18 - Uso da *gem Kaminari* na página de Assuntos

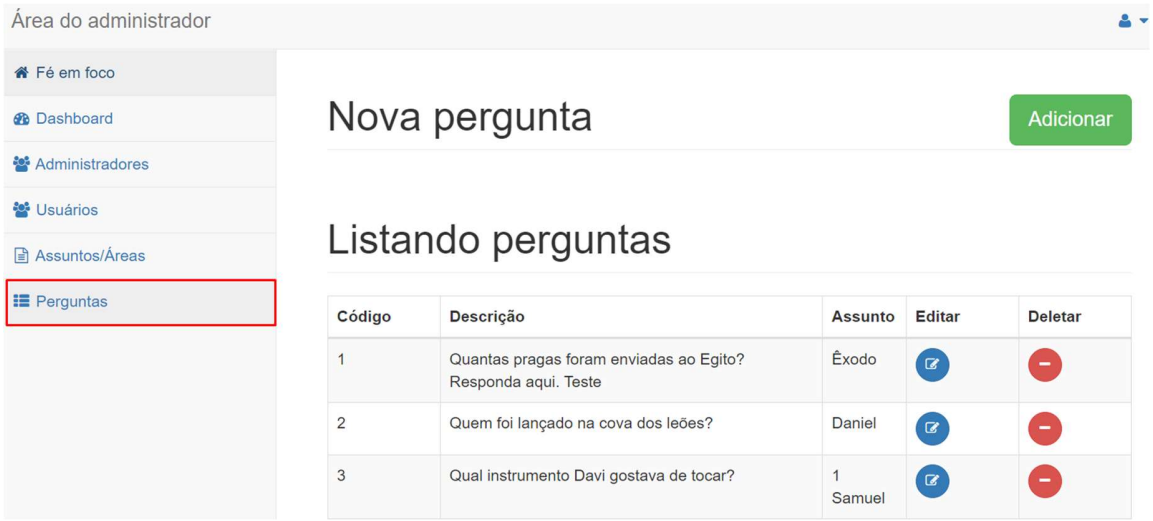
Código	Descrição	Quantidade de questões	Editar	Deletar
1	Velho Testamento	3		
2	Gênesis	2		
3	Êxodo	3		
4	Levítico	0		
5	Números	0		

Fonte: elaborado pelo autor (2023)

3.4.5 Fase de Planejamento: *Release 5*

A Quinta *Release* teve a duração de uma semana. Nesta *Release* foi desenvolvida a página de Perguntas e todos os seus recursos, que como as anteriores, possui uma estrutura completa de CRUD (*Create, Read, Update, Delete*). A Figura 19 mostra a página de perguntas, que possui uma tabela listando as perguntas cadastradas bem como o assunto a que elas pertencem.

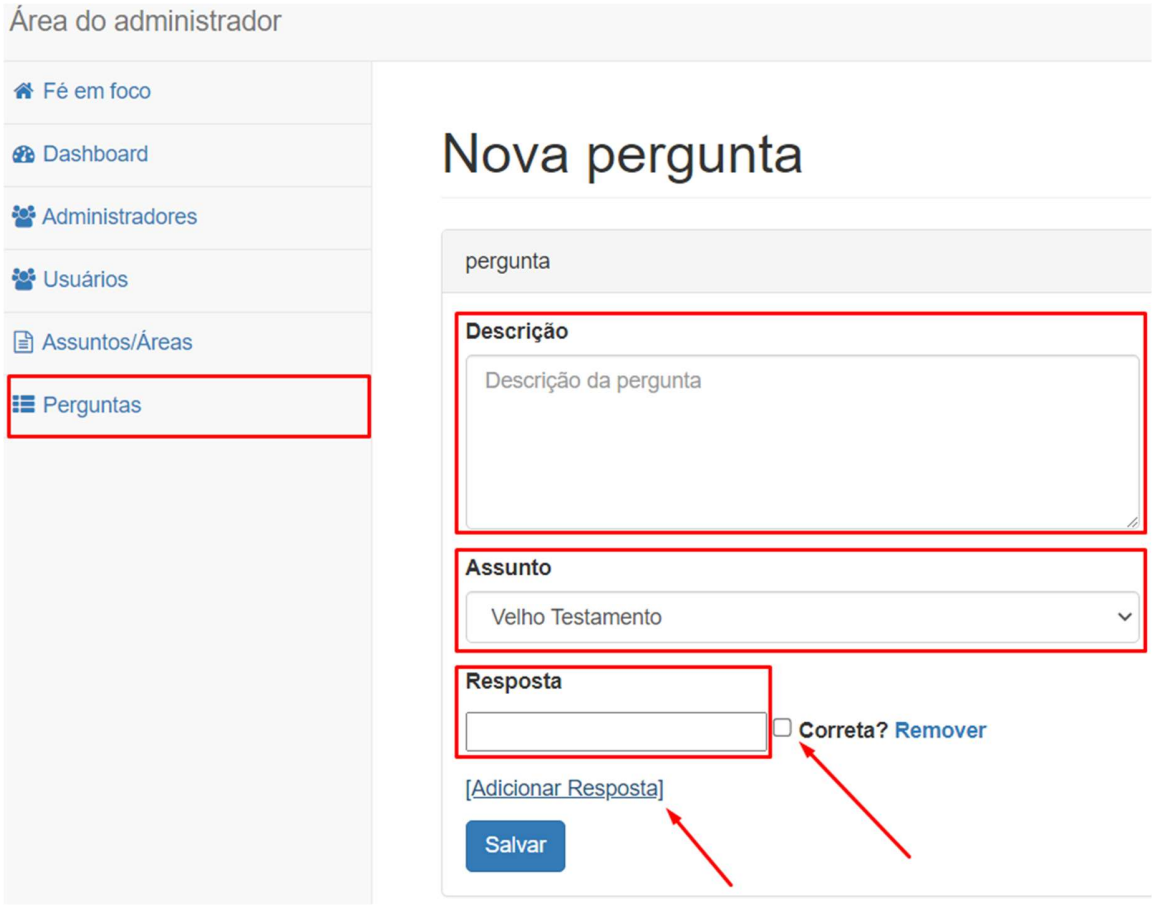
Figura 19 - Página de Perguntas



Fonte: elaborado pelo autor (2023)

Nesta *Release* foi desenvolvida também a área de cadastro das respostas para as perguntas. As respostas são cadastradas juntamente com a pergunta. Uma pergunta possui uma quantidade variável de respostas, sendo possível adicionar uma nova resposta, bastando apenas clicar no campo “Adicionar Resposta”. Logo a frente do campo de resposta possui uma caixa para marcar, caso a resposta seja a correta. Vale ressaltar que todas as perguntas, independente de quantas alternativas possuam, existe apenas uma resposta correta. A Figura 20 exemplifica graficamente o que foi explicado:

Figura 20 - Cadastrando novas perguntas



Fonte: elaborado pelo autor (2023)

3.4.6 Fase de Planejamento: *Release 6*

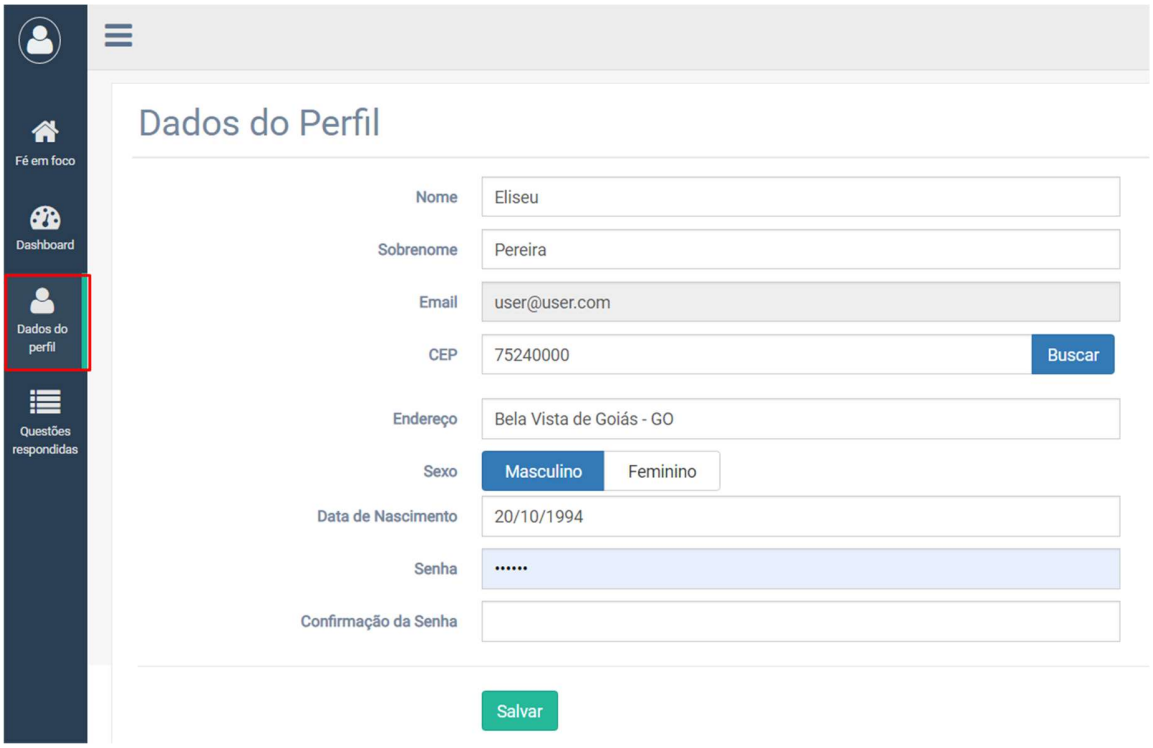
A sexta *Release* teve a duração de duas semanas. Nesta foi desenvolvido o perfil do Usuário sendo utilizado o *template Gentelella Alela! ("Gentelella Alela!", [s.d.])*. Ao fazer *login*, o usuário acessa a página inicial do perfil, podendo acompanhar seu desempenho, saber quantas perguntas já foram respondidas, quantas ele acertou e quantas ele errou. Essa página inicial do usuário se chama "*Dashboard*" e é apresentada na Figura 21 a seguir.

Figura 21 - Página inicial do usuário

Fonte: elaborado pelo autor (2023)

Em seguida, foi desenvolvida a página de Dados do perfil, onde o usuário poderá alterar suas informações pessoais como nome, sobrenome, endereço, sexo, data de nascimento e senha, como pode ser visto na Figura 22:

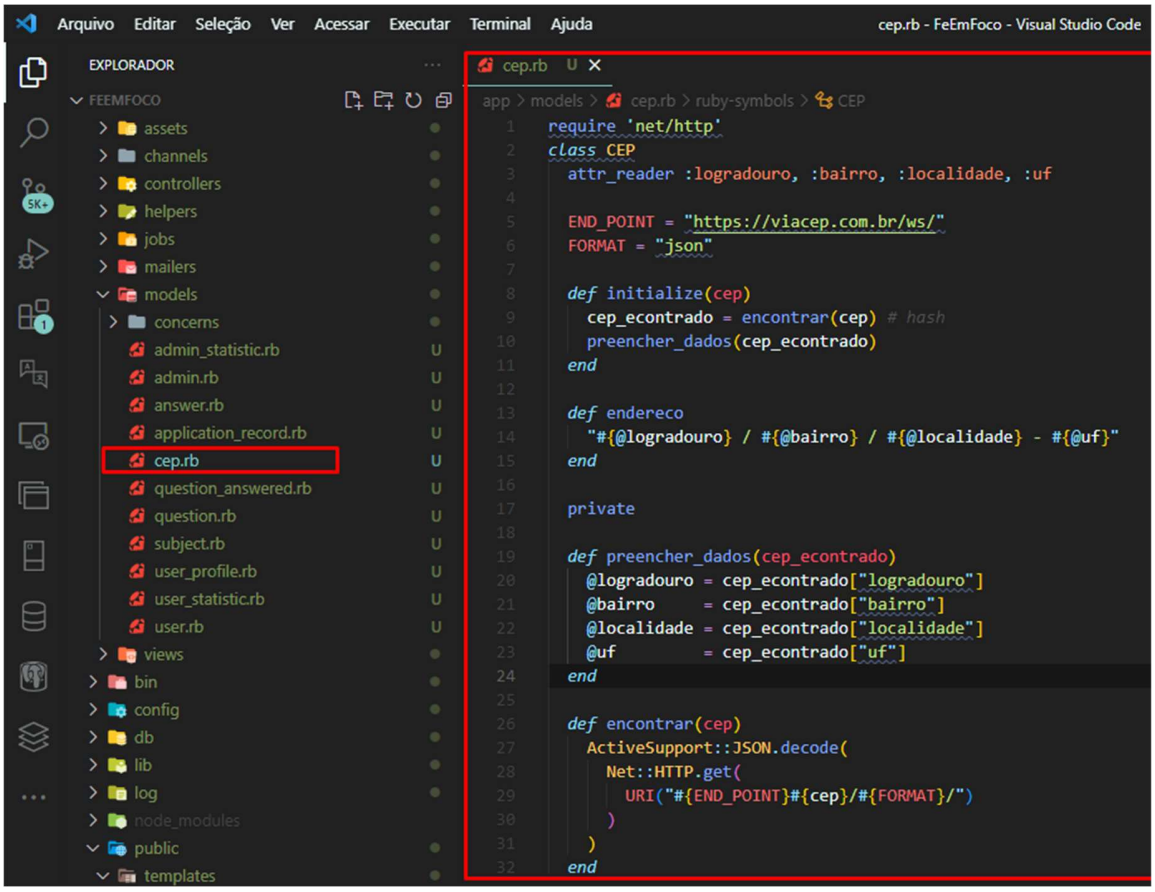
Figura 22 - Página de dados do perfil do usuário



Fonte: elaborado pelo autor (2023)

Ainda, para facilitar o preenchimento do endereço, foi criada uma *API* que busca as informações do CEP informado no *webservice* ViaCEP, bastando informar o código postal no campo de busca, conforme imagem anterior (Figura 22). Caso sejam válidas, as informações relacionadas ao CEP informado são visualizadas. A seguir, na Figura 23, pode-se observar o código que executa este serviço.

Figura 23 - Buscar CEP



Fonte: elaborado pelo autor (2023)

Também, nesta *Release,* foi criada a página de questões respondidas, onde fica visível para o usuário uma tabela detalhada de todas as perguntas respondidas. Nesta tabela, cada linha contém a descrição da questão, a resposta informada, a resposta correta e o resultado. A Figura 24 a seguir mostra em detalhes essa página.

Figura 24 - Página de questões respondidas

Fonte: elaborado pelo autor (2023)

Por fim, ainda nesta *Release,* foi criada a página de usuários na área do Administrador, onde é possível visualizar o nome completo do usuário e seu e-mail. Embora o *link* já existisse desde a implementação da área do Administrador, conforme visto na *Release* 3, somente na *Release* 6 ele de fato está habilitado e funcionando, conforme Figura 25, a seguir:

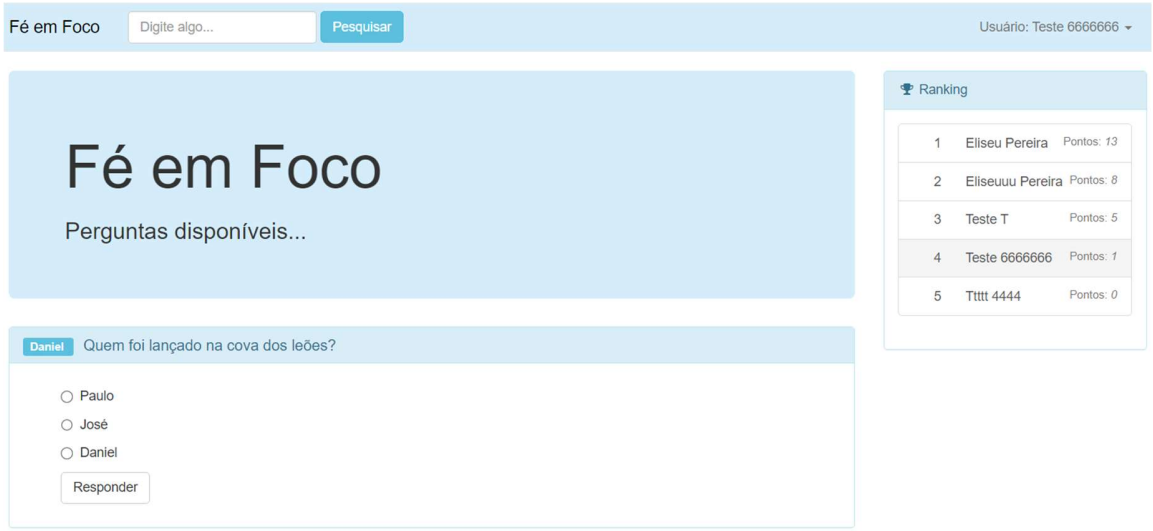
Figura 25 - Página dos administradores para visualizar os usuários

Fonte: elaborado pelo autor (2023)

3.4.7 Fase de Planejamento: *Release 7*

A sétima e última *Release* teve a duração de uma semana. Nesta *Release* foi finalizada a implementação da página inicial e consequentemente a implementação da aplicação. Na página inicial é possível realizar buscas, responder perguntas e acompanha o *ranking* dos cinco usuários que mais pontuaram. Segue a Figura 26, mostrando graficamente como ficou a página inicial do site para o usuário que possui cadastro:

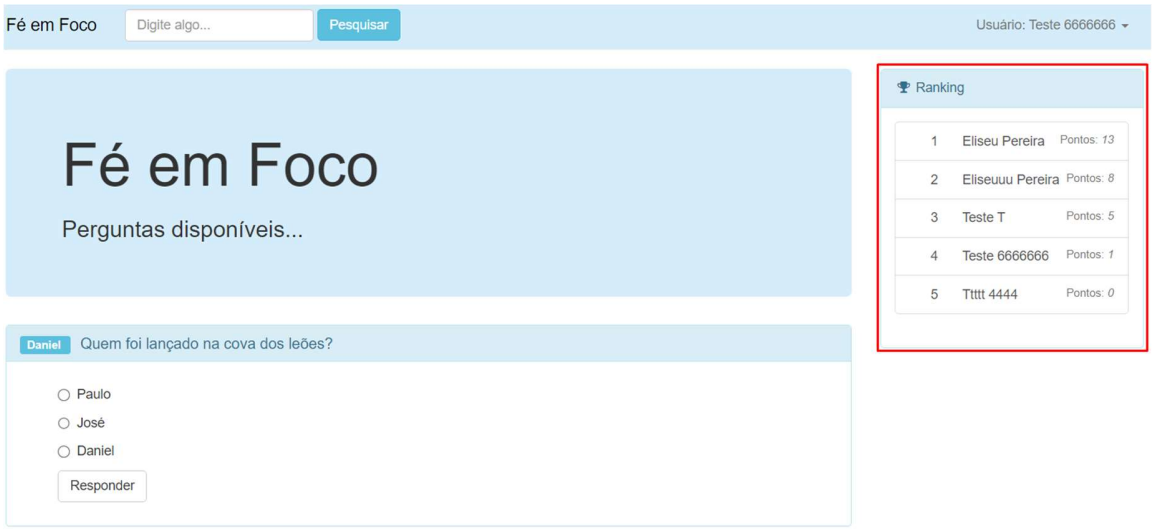
Figura 26 - Página inicial do Site



Fonte: elaborado pelo autor (2023)

Na Figura 27, é mostrado o *ranking*, onde é exposto para todos os usuários cadastrados, os cinco maiores pontuadores da aplicação Fé em Foco:

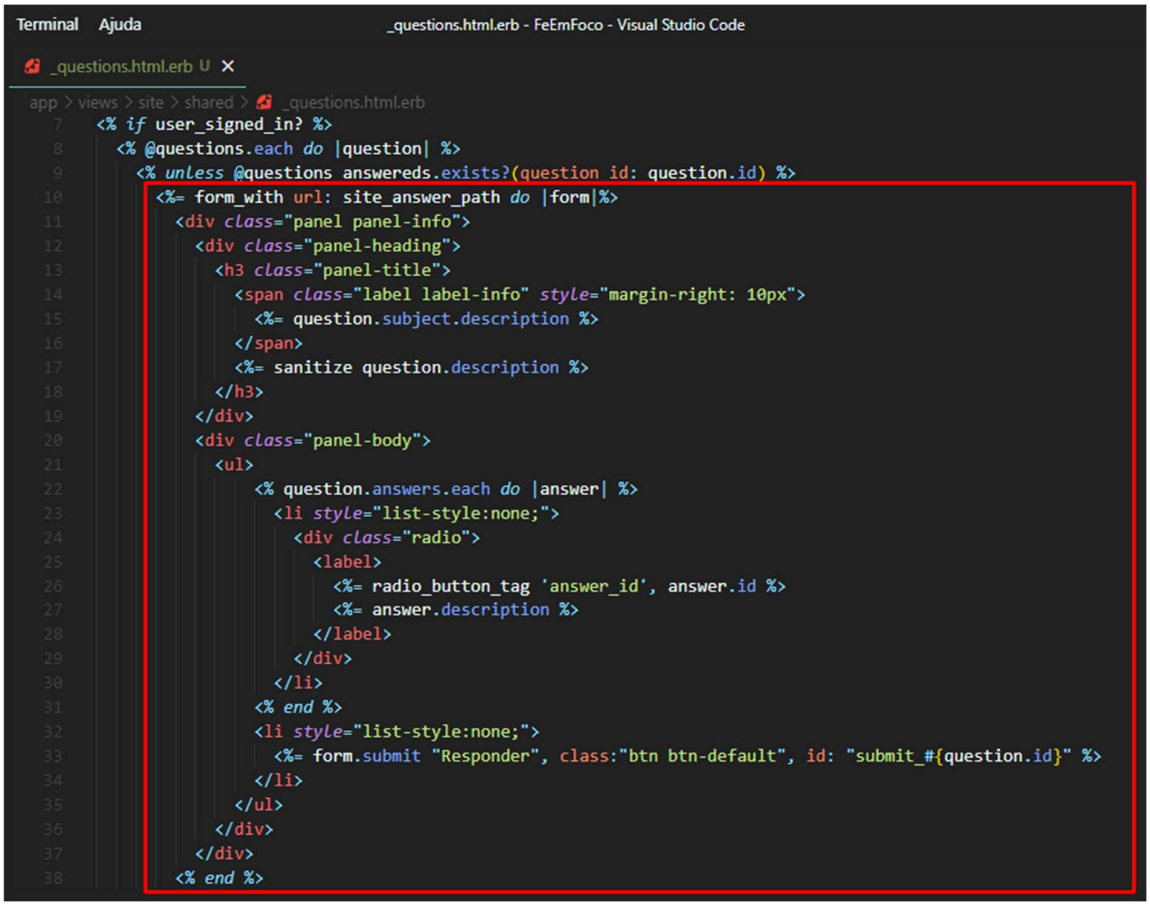
Figura 27 - Ranking visível para usuários cadastrados



Fonte: elaborado pelo autor (2023)

A lista de perguntas da página inicial foi implementada utilizando o recurso do *Rails* chamado `form_with` que lista todas as perguntas e para cada pergunta é disponibilizado um botão para marcar a resposta, outra grande vantagem do *Ruby On Rails*. A seguir, a Figura 28 expõe um trecho do código da página inicial que utiliza o `form_with`:

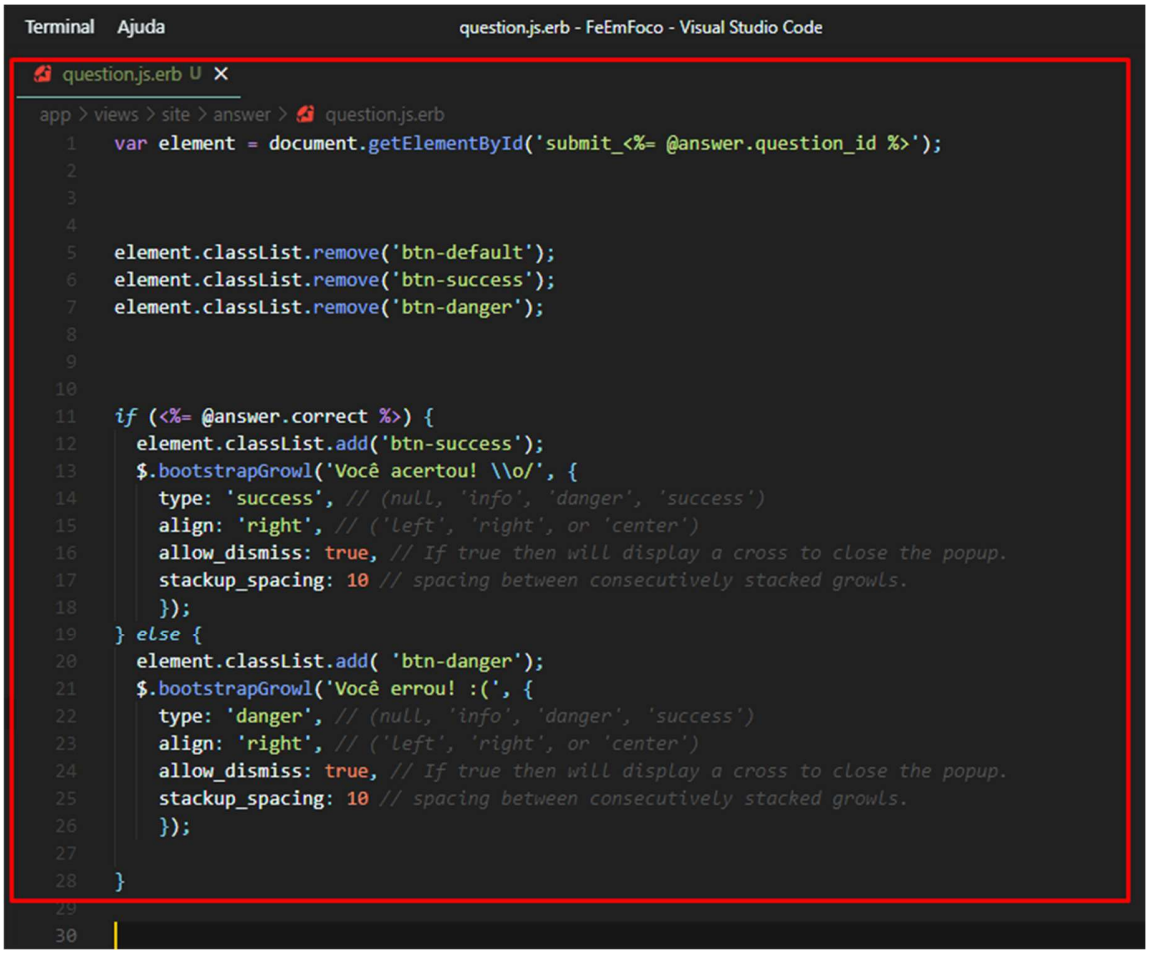
Figura 28 - Código *form_with* da página inicial



Fonte: elaborado pelo autor (2023)

A verificação da resposta é feita via *Javascript*, onde o `form_with` dispara o botão assim que pressionado, chamando o trecho do código em *Javascript* para verificar a resposta antes de gravar na tabela de estatísticas do usuário, no banco de dados. A Figura 29, a seguir é do código de verificação da resposta correta, em *Javascript*:

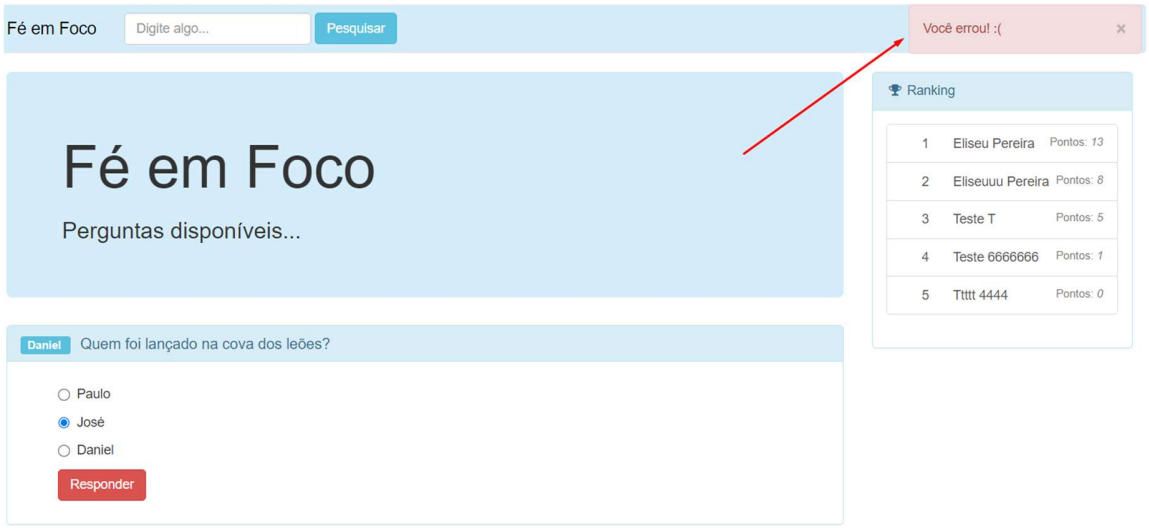
Figura 29 - Código de verificação da resposta



Fonte: elaborado pelo autor (2023)

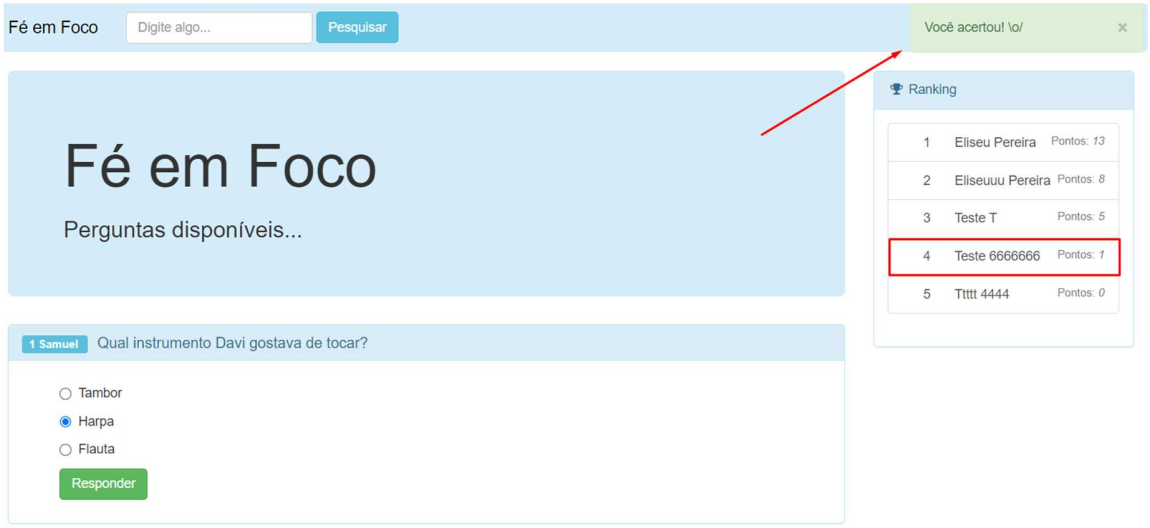
Ao responder uma questão, aparece imediatamente uma mensagem informando se o usuário acertou ou errou a resposta. Também, ao responder, o botão fica vermelho caso a resposta seja errada ou verde, caso a resposta seja certa. A Figura 30 mostra uma resposta errada e a Figura 31 mostra uma resposta certa.

Figura 30 - Errando a resposta



Fonte: elaborado pelo autor (2023)

Figura 31 - Acertando a resposta

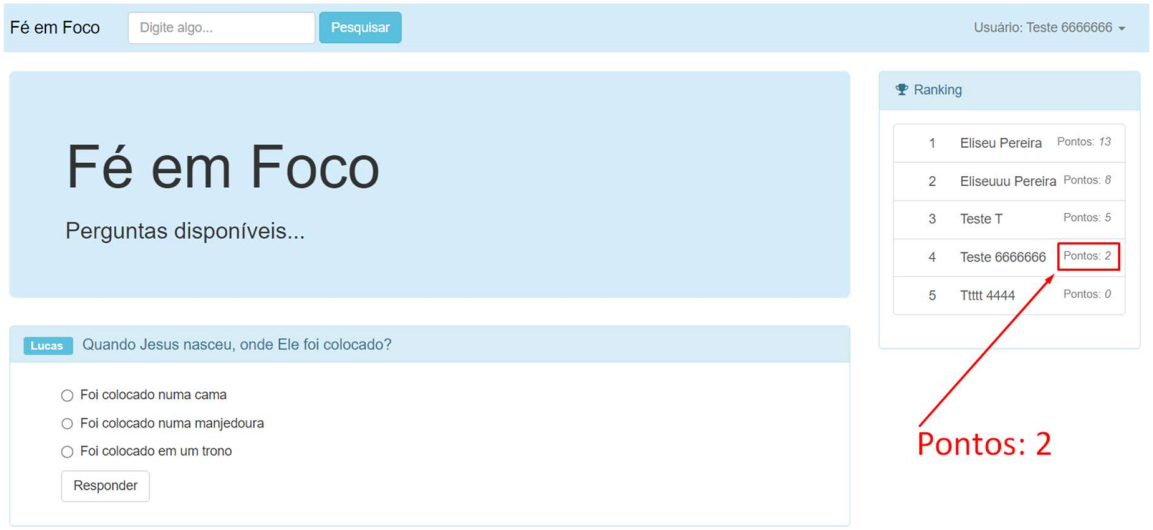


Fonte: elaborado pelo autor (2023)

Ao acertar a resposta, conforme implementado, a pontuação do usuário soma em mais um, podendo se manter ou subir de posição no ranking. Ao analisar a pontuação do usuário número quatro da Figura 31 e após esse usuário acertar a resposta da pergunta, na Figura 32, a seguir, atesta a funcionalidade do ranking,

somando a pontuação em mais um para o usuário que acertou a resposta. Uma vez respondida, a pergunta não fica mais disponível na página pois já fora respondida.

Figura 32 - Atualização de pontos do *ranking*



Fonte: elaborado pelo autor (2023)

Foi necessário um total de sete *Releases* para construir a aplicação, desde a criação do diagrama de atividades até a finalização da última implementação da aplicação. No próximo capítulo serão abordados os resultados esperados e as dificuldades encontradas durante toda a implementação.

4 RESULTADOS

O resultado esperado com esse projeto de construir a aplicação *web* Fé em Foco, estilo *Quiz* de perguntas e respostas, é provar o quão ágil é a construção de aplicações *web* utilizando o *Ruby on Rails* aliado à metodologia XP. Fica comprovado através do estudo de caso que é possível construir aplicações *web* funcionais em um curto espaço de tempo por causa dos recursos disponíveis no *Rails*, da fácil interpretação que o *Ruby* proporciona e da metodologia XP.

A principal dificuldade encontrada durante o desenvolvimento do projeto foi por não trabalhar diariamente com desenvolvimento de software e a limitação de

conhecimento específico a respeito do *Ruby on Rails* e da metodologia XP. Entretanto, devido as facilidades que o *Ruby on Rails* proporciona, ainda sim foi possível desenvolver a aplicação dentro do prazo planejado e com os resultados.

Por fim, os resultados obtidos são expressivos quando comparados às dificuldades encontradas, expondo que, para desenvolvedores *web fullstack*, pequenas e médias empresas que precisam desenvolver *softwares* de forma ágil, o *Ruby on Rails* aliado à metodologia XP é uma ótima opção, principalmente pelas seguintes vantagens:

- Facilidade do *Rails* de criar toda a estrutura base da aplicação, isto é, todos os diretórios, arquivos de configuração e de dependências que o sistema precisa para ser executado com apenas três comandos no terminal (`rails new nome_da_aplicação`).
- Servidor integrado ao *Rails*, que através de apenas dois comandos (`rails s`), torna possível a execução local da aplicação para ser testada e validada no navegador;
- Mesclagem de código Ruby em scripts de outras linguagens com os operadores `<% %>` e `<%= %>` facilitando e simplificando a integração entre linguagens a serem utilizadas conforme a necessidade;
- Técnica de mapeamento de objeto relacional (*Object Relational Mapper*), que torna possível a utilização das funções de CRUD (*Create, Read, Update e Delete*) sem escrever instruções *SQL* (*Structured Query Language*) diretamente e com menos código geral de acesso ao banco de dados.
- *Gems Ruby* que disponibilizam recursos desde a parte de segurança até a parte visual da aplicação, como por exemplo, a *gem Devise* que gera os perfis de acesso, com as devidas restrições e permissões e a *gem Kaminari*, que gera a paginação, conforme consulta simplificada no banco de dados através da técnica de ORM (*Object Relational Mapper*);
- Facilidade do *Rails* de criar estrutura completa de CRUD (*Create, Read, Update, Delete*) com o comando *scaffolding*, não precisando implementar tais funções;

- Construção rápida e organizada da aplicação utilizando as práticas e valores da metodologia ágil *Extreme Programming* (XP).

5 CONCLUSÃO

Um dos meios de minimizar o *déficit* de aplicações que resolvam efetivamente os problemas do dia a dia, principalmente a falta de tempo para desenvolvê-las, foi a construção de *softwares* de desenvolvimento que contenham recursos já implementados (*frameworks*), cujo objetivo é agilizar e facilitar o trabalho árduo dos desenvolvedores. Para tornar esse processo de desenvolvimento ainda mais rápido e organizado, é adotado também uma metodologia ágil de desenvolvimento na construção do *software*.

Diante do exposto, este trabalho demonstra que o *Ruby on Rails* aliado à metodologia *Extreme Programming* (XP) é uma ótima opção para desenvolvimento *web* pois é possível construir aplicações *web* funcionais rapidamente e de forma organizada.

Conclui-se que os recursos proporcionados pelo *framework Ruby on Rails* e as práticas e valores adotados pela metodologia *Extreme Programming* (XP) apresentados evidenciam as reais vantagens do desenvolvimento de aplicações *web* utilizando o *Ruby on Rails* aliado à *Extreme Programming* (XP).

6 TRABALHOS FUTUROS

Para trabalhos futuros sugere-se os pontos a seguir:

- Integração de uma ferramenta que utilize a linguagem de sinais para, no quesito acessibilidade, permitir que usuários mudos e/ou surdos possam utilizar a aplicação com mais facilidade;
- Implementação de mudança de contrastes para facilitar a utilização da aplicação por usuários com algumas dificuldades visuais;
- Implementação da funcionalidade de sinais sonoros durante a interação do usuário com a aplicação;

- Implementação de uma funcionalidade de customização que permita a mudança de cor de acordo com as preferências do usuário;
- Implementar versões da aplicação para *smartphones* (sistema operacional *Android*).

7 REFERÊNCIAS

Cascading Style Sheets. Disponível em: <https://www.w3.org/Style/CSS/>. Acesso em: 17 maio. 2023.

EDUCAÇÃO, R. X. **Desenvolvimento web: TUDO o que você precisa saber!** Disponível em: <https://blog.xpeducacao.com.br/desenvolvimento-web>. Acesso em: 7 maio. 2023.

Start Bootstrap. Disponível em: <https://startbootstrap.com/template-overviews/sb-admin-2/>. Acesso em: 13 maio. 2023.

Gentelella Alela!. Disponível em: <https://colorlib.com/polygon/gentelella/>. Acesso em: 24 maio. 2023.

Gentelella. Disponível em: <https://github.com/ColorlibHQ/gentelella>. Acesso em: 13 maio. 2023.

GIL, A. C. **Como elaborar projetos de pesquisa**, 5. ed. São Paulo: Atlas, 2010.

GIT. **Git**. Disponível em: <https://git-scm.com/>. Acesso em: 17 maio. 2023.

HTML Standard. Disponível em: <https://html.spec.whatwg.org/>. Acesso em: 13 maio. 2023.

IDE do Visual Studio, Editor de Código, Azure DevOps e App Center - Visual Studio. Disponível em: <https://visualstudio.microsoft.com/pt-br/>. Acesso em: 13 maio. 2023.

JavaScript Brasil - WebSite. Disponível em: <https://brasil.js.org/>. Acesso em: 18 maio. 2023. Acesso em: 17 maio. 2023.

JQUERY.ORG, JQUERY F. -. **jQuery 3.3.1 – fixed dependencies in release tag | Official jQuery Blog**. Disponível em: <https://blog.jquery.com/2018/01/20/jquery-3-3-1-fixed-dependencies-in-release-tag/>. Acesso em: 18 maio. 2023.

Linguagem de Programação Ruby. Disponível em: <https://www.ruby-lang.org/pt/>. Acesso em: 13 maio. 2023.

LOSS, Felipe Schroll; DAL PONTE, Maico Silvino. **Desenvolvimento de um aplicativo web para salões de beleza utilizando Ruby on Rails**. Universidade Tecnológica Federal do Paraná. Pato Branco, 2015.

MANHÃES TELES, VINÍCIUS. **UM ESTUDO DE CASO DA ADOÇÃO DAS PRÁTICAS E VALORES DO EXTREME PROGRAMMING** / Vinícius Manhães Teles. Rio de Janeiro: UFRJ / IM / DCC, 2005.

NETO, M. B. DE S. **BR Modelo Web - Ferramenta online para modelagem de banco de dados**. Disponível em: <https://www.brmodeloweb.com/lang/pt-br/index.html>. Acesso em: 17 maio. 2023.

Node.js. Disponível em: <https://nodejs.org/pt-br>. Acesso em: 17 maio. 2023.

npm | build amazing things. Disponível em: <https://www.npmjs.com/>. Acesso em: 17 maio. 2023.

AMARAL, Lucas. **O que são Métodologias Ágeis?** Disponível em: <https://www.tabnews.com.br/LucasAmaral/o-que-sao-metodologias-ageis>. Acesso em: 20 maio. 2023.

OTTO, M. **Bootstrap Blog**. Disponível em: <https://blog.getbootstrap.com/>. Acesso em: 17 maio. 2023.

Powerful and Fast UML Diagramming Software. Disponível em: <https://astah.net/products/astah-uml/>. Acesso em: 17 maio. 2023.

PIRES, Jackson. **Ruby on Rails 5.x - Do início ao fim!** Disponível em: <https://www.udemy.com/course/rubyonrails-5x/>. Acesso em: 7 maio. 2023.

REJMAN, M. **50 Best Ruby On Rails Companies Websites [State For 2022]**. Disponível em: <https://www.ideamotive.co/blog/best-ruby-on-rails-companies-websites>. Acesso em: 13 maio. 2023.

Ruby on Rails. Disponível em: <https://rubyonrails.org/>. Acesso em: 13 maio. 2023.

SANTOS, R. DOS. **O que é CRUD? Por que você precisa criar um CRUD**. Disponível em: <https://www.brasilcode.com.br/o-que-e-crud-por-que-voce-precisa-criar-um-crud/>.

SILVA, G. **O que é arquitetura MVC?** Disponível em: <https://coodesh.com/blog/dicionario/o-que-e-arquitetura-mvc/>. Acesso em: 7 maio. 2023.

Sobre o Ruby. Disponível em: <https://www.ruby-lang.org/pt/about/>. Acesso em: 13 maio. 2023.

Sommerville, Ian. **Engenharia de Software** / Ian Sommerville ; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama. — 9. ed. — São Paulo: Pearson Prentice Hall, 2011.

SOUZA, Lucas. **Ruby - Aprenda a programar na linguagem mais divertida**. Casa do código, 2013 - Disponível em: <http://www.casadocodigo.com.br/products/livro-ruby> Acesso em: 01/05/2023.

Understanding the basics of Ruby on Rails: HTTP, MVC, and Routes. Disponível em: <https://www.freecodecamp.org/news/understanding-the-basics-of-ruby-on-rails-http-mvc-and-routes-359b8d809c7a/>. Acesso em: 13 maio. 2023.

ViaCEP - Webservice CEP e IBGE gratuito. Disponível em: <https://viacep.com.br/>. Acesso em: 24/05/2023.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier, 2014.

WILDT, Daniel *et al*. **EXtreme Programming**: práticas para o dia a dia no Desenvolvimento Ágil de Software. São Paulo: Casa do Código, 2015. E-book.

Yarn. Disponível em: <https://classic.yarnpkg.com/lang/en/>. Acesso em: 13 maio. 2023.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante Eliseu Alves Ribeiro Pereira do Curso de Ciência da Computação, matrícula 2017.1.0028.0078-5, telefone: (62) 9 9699-2964, e-mail eliseualvesribeiropereira@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Desenvolvimento Ágil de Aplicações Web, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 19 de maio de 2023.

Assinatura do autor: Eliseu Alves R. Pereira

Nome completo do autor: Eliseu Alves Ribeiro Pereira

Assinatura do professor–orientador:

Nome completo do professor-orientador: Anibal Santos Lakemura