Nikollas Ferreira Gonçalves

Protótipo de uma aplicação web para gestão de eventos online

Formiga - MG

2022

Nikollas Ferreira Gonçalves

Protótipo de uma aplicação web para gestão de eventos online

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Me. Roger Santos Ferreira

Formiga - MG

2022

Gonçalves, Nikollas Ferreira
G635p Protótipo de uma Aplicação WEB para Gestão de Eventos Online / Nikollas Ferreira Gonçalves - Formiga : IFMG, 2022.
64p. : il.

Orientador: Prof. MSc. Roger Santos Ferreira
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Laravel. 2. Gerenciamento de Eventos. 3. Next. Js. 4. Aplicação WEB. 5. Engenharia de Software. I. Ferreira, Roger Santos. II. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Msc. Simoni Júlia da Silveira

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Formiga
Diretoria de Ensino
Docência Área Acadêmica de Computação
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

NIKOLLAS FERREIRA GONÇALVES

PROTÓTIPO DE UMA APLICAÇÃO WEB PARA GESTÃO DE EVENTOS ONLINE

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

APROVADO em 25 de novembro de 2022.

BANCA EXAMINADORA

Prof. Roger Santos Ferreira (Orientador)

Prof. Manoel Pereira Junior (IFMG)

Prof. Fernando Paim Lima (IFMG)

Formiga,25 de novembro de 2022.

Documento assinado eletronicamente por **Roger Santos Ferreira**, **Professor**, em 28/11/2022, às 08:15, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Manoel Pereira Junior**, **Professor**, em 28/11/2022, às 08:19, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **Fernando Paim Lima**, **Professor**, em 01/12/2022, às 15:52, conforme art. 1º, III, "b", da Lei 11.419/2006.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **1389789** e o código CRC **5C94023F**.

Dedico este trabalho a minha família, pois é graças ao seu incentivo que hoje pude concluir o meu curso.

Agradecimentos

Agradeço a todos que estiveram presentes nos momentos de dificuldade desta monografia, deste curso, em especial minha família que me apoiou constantemente nesta jornada.

"As máquinas me surpreendem muito frequentemente."
(Alan Turing)

Resumo

Com o aumento da demanda por eventos online nos últimos anos, foram necessários adotar métodos que antes não eram utilizados, como aplicações gerenciadoras de eventos. O presente trabalho apresenta um protótipo de uma aplicação web com o propósito de gerir os eventos, utilizando-se de *frameworks* modernos, como o Next.js para o desenvolvimento do frontend e o Laravel para o backend. A distribuição vertical se deve à aplicação ser dividida logicamente em dois servidores distintos, que melhora o gerenciamento de sua arquitetura. Com estes recursos e seu planejamento com a metodologia unificada, o protótipo final apresenta a implementação por meio das tecnologias citadas, além de exemplificar o uso dos conceitos almejados no desenvolvimento de aplicações web atualmente: design responsivo e fluído, além de demonstrar a divisão de responsabilidades entre as camadas de software.

Palavras-chave: Laravel, Gerenciamento de eventos, Next.js

Abstract

With the increase in demand for online events in recent years, it was necessary to adopt methods that were not used before, such as event management applications. The current work presents a prototype of a web application with the purpose of managing events, using modern frameworks, like Next.js for the frontend development and Laravel for the backend. The vertical distribution is due to the application being logically divided into two distinct servers, which improves the management of its architecture. With these resources and it's planning with the unified methodology, the final prototype presents the implementation through the mentioned technologies, in addition to exemplifying the use of the desired concepts in the development of web applications today: responsive and fluid design, in addition to demonstrating the division of responsibilities between software layers.

Keywords: Laravel, Event management, Next.js

Lista de ilustrações

Lista de códigos

Lista de abreviaturas e siglas

HTML	*Hyper Text Markup Language*
CSS	*Cascade Style Sheet*
JS	*JavaScript*
TS	*TypeScript*
PHP	*PHP: Hypertext Preprocessor*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
ORM	*Object Relational Mapper*
SQL	*Structured Query Language*
API	*Application Programming Interface*
SGBD	*Sistema Gerenciador de Banco de Dados*
PU	*Processo Unificado*
CDN	*Content Delivery Network*
IHC	*Interface Humano Computador*

Sumário

1 Introdução

Segundo o SEBRAE (2020), em abril de 2020, a pandemia do covid-19 afetou 98% do setor de eventos, devido a cancelamentos, renegociações, entre outros motivos para evitar aglomerações. Destes eventos afetados, muitos empresários estão se preocupando com aprimorar sua gestão, e conforme o site feiras do Brasil o número de eventos online para 2021 já supera o número presencial devido à alta demanda. Além disso, de acordo com estudo da Associação Brasileira de Promotores de Eventos (Abrape), 51,9% dos eventos previstos para ocorrer neste período foram cancelados ou adiados (BRASIL, 2021).

Para que estes eventos sejam realizados de uma forma mais eficiente, utilizam-se diversas ferramentas online que possibilitam o gerenciamento de eventos remotos e presenciais, entre essas:

- **Even3:** O Even3 é uma plataforma online a qual disponibiliza espaço para que instituições e/ou organizações criem seus próprios eventos e os hospedem, neste também é possível a criação de eventos gratuitos e pagos, além da emissão de certificados. E dito isso, segundo o site do Reclame Aqui, o maior problema deles é sua emissão de certificado, que apresenta problemas por parte da própria aplicação.
- **OCS:** O Open Conference System é um *software* que teve seu desenvolvimento paralisado pela equipe, e que possui seu código aberto onde o usuário pode elaborar o seu próprio site de compartilhamento de conferências acadêmicas. Este processo é realizado manualmente, como a página do site, por exemplo, mas há exceções do que vem fornecido pelo sistema, o cadastro de participantes, por exemplo.
- **Sympla:** O Sympla é uma plataforma, similar ao Even3, que se propõe a gestão completa e otimizada através de dezenas de ferramentas simples de usar, com eventos gratuitos e pagos também, além de shows e eventos. Seu maior problema, segundo a sua página no Reclame Aqui, é no quesito de ingressos, principalmente em sua compra e recebimento.
- **EventBrite:** O Eventbrite é uma plataforma global de ingressos de autoatendimento para experiências ao vivo que permite a qualquer pessoa criar, compartilhar, encontrar e participar de eventos, festivais, maratonas e conferências, entre outros. Mas grande parte de seus problemas também se dão na compra e recebimento de ingressos, de acordo com reclamações de sua página no Reclame Aqui.
- **S-EVA:** O S-EVA (Sistema de Eventos Acadêmicos) é um protótipo que implementa a solução de eventos acadêmicos, no qual o seu foco é utilizar um banco de dados

integrado a um *framework* para interface gráfica, além de avaliar trabalhos correlatos e destacar pontos importantes sobre gestão de eventos.

1.1 Justificativa

Aplicar os conceitos aprendidos teoricamente em cursos de tecnologia da informação, como Sistemas de Informação, Engenharia da Computação e principalmente a Ciência da Computação, visando um design fluído, com boas práticas de IHC, a modelagem de banco de dados onde a implementação de uma metodologia de desenvolvimento e projeto para a aplicação proposta por meio de *frameworks* modernos.

1.2 Objetivos

O objetivo geral deste trabalho de conclusão de curso é desenvolver um protótipo de uma aplicação web para gerenciar eventos acadêmicos online, exemplificando o uso de conceitos almejados no desenvolvimento de aplicações web atualmente.

- Modelar a arquitetura do sistema.
- Implementação do modelo de banco de dados.
- Desenvolvimento da camada de *backend* em Laravel.
- Desenvolvimento da camada de *frontend* em Next.js.
- Implementação do funcionamento do gerenciador de eventos.

1.3 Estrutura do trabalho

Este trabalho está dividido em seis capítulos. O capítulo 2 apresenta o referencial teórico. O capítulo 3 expõe a metodologia. No capítulo 4 é retratado o desenvolvimento. No capítulo 5 são apresentados os resultados obtidos. Por fim, no capítulo 6 são feitas as considerações finais e trabalhos futuros, seguidas pelas referências bibliográficas.

2 Fundamentação Teórica

Para entendimento do protótipo proposto que será apresentado, são necessários alguns conceitos. Por se tratar de uma aplicação web, primeiramente é preciso definir os conceitos do planejamento e estruturação, por meio da Engenharia de *Software*, o conceito de *frontend*, sendo a interface da aplicação com o usuário, o *backend*, onde é feito o tratamento das informações com o banco de dados.

2.1 Engenharia de *Software*

A tendência é que sistemas de *software* fiquem cada vez maiores e mais complexos, e isto se deve ao poder computacional crescente a cada ano. E isto faz com que os usuários criem maiores expectativas em relação aos sistemas de *software*, que não apenas devem transmitir informações pela internet de forma rápida e segura, mas também adaptados à necessidade de quem o usa. (SOMMERVILE, 2011).

A Engenharia de *Software* abrange os processos com um conjunto de métodos e diversas ferramentas que possibilitam aos profissionais desenvolverem sistemas de *software* com melhor planejamento (PRESSMAN, 2011). E isto é essencial para decidir qual funcionalidade o sistema terá e poderá ser apresentada ao usuário final, assim como a estruturação do próprio projeto e principalmente estabelecer os limites do sistema, ou seja, até que ponto será desenvolvido baseado no contexto do *software* (SOMMERVILE, 2011).

Para que isso seja planejado em uma aplicação e utilizada corretamente, de acordo com Sommervile (2011, p.126): “Modelos de contexto do sistema e modelos de interação apresentam visões complementares dos relacionamentos entre um sistema e seu ambiente.”

2.1.1 Caso de uso

Segundo Pressman (2011), a UML é uma linguagem padrão para descrever e documentar um projeto de *software* que reúne diversos grupos de notações de modelagem, sendo cada um com sua devida sintaxe e semântica predeterminadas. Cada um desses elementos pode ser extensível, permitindo, deste modo, que sejam adaptáveis às características específicas do projeto a ser desenvolvido. A UML independe de linguagem de programação e dos processos de desenvolvimento do *software*, fazendo com que diferentes abordagens possam ser utilizadas durante o processo (BEZERRA, 2007).

O diagrama UML de caso de uso determina as funcionalidades e características presentes no *software*, sendo uma visão geral do ponto de vista do usuário que descreve como este irá interagir com o sistema, seguindo determinadas ações para realizar um objetivo,

como, por exemplo, efetuar um login. Há uma padronização em seu desenvolvimento, representada pelas seguintes características, segundo Pressman (2011) e Sommervile (2011):

- **Ator:** representado por figuras de um boneco palito, é aquele que irá interagir com o sistema, sendo este externo ao sistema desenvolvido.
- **Caso de uso:** representado por uma elipse no diagrama, estes demonstram as ações que os atores podem realizar dentro do sistema, ou seja, identificam as interações individuais entre o sistema e seus usuários.
- **Sistema:** elemento opcional representado por um retângulo para definir o objetivo de cada parte do diagrama de caso de uso.

2.1.2 Processo unificado

O processo unificado é um modelo incremental e iterativo derivado de trabalhos sobre a UML que ilustra as práticas na especificação, no projeto e prevê a prototipação aliada a entrega incremental do desenvolvimento. o PU é descrito em cima de três perspectivas, que podem ser descritas como dinâmica, que mostra as fases do modelo ao longo do tempo, estática, que mostra as atividades realizadas no processo e prática, que sugere boas práticas a serem usadas durante o processo (SOMMERVILE, 2011).

Segundo Bezerra (2007) existem cinco fases do PU, sendo estas:

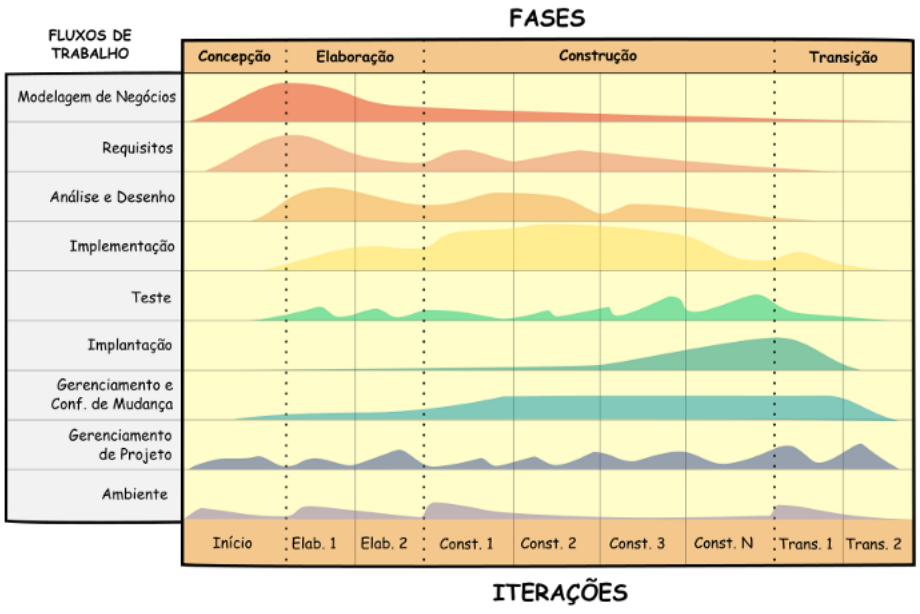
- **Fase de concepção:** identifica as entidades externas que irão interagir com o sistema, descrevendo os requisitos do projeto, que se desenvolve todo o planejamento para ser possível as futuras iterações incrementais do processo. Nesta etapa é desenvolvido, também, a arquitetura provisória do sistema, que será refinada e expandida nos processos subsequentes.
- **Fase de elaboração:** refina e expande, por meio da ampliação da arquitetura do sistema, a fase de concepção, criando o modelo de requisitos e o modelo de implementação, mesmo ainda não oferecendo os recursos necessários para a utilização do sistema. Definem-se as tecnologias que serão utilizadas ao longo do projeto, como frameworks e banco de dados, por exemplo.
- **Fase de construção:** é a fase que será desenvolvido o sistema por meio da programação, ou seja, as iterações desta etapa serão constituídas por fazer uma parte do sistema e a integrar as demais existentes, podendo ser desenvolvidas em paralelo. Tudo isto deve ser feito seguindo as documentações geradas nesta etapa e nas anteriores, para não haver falha na entrega dos requisitos necessários. E ao fim desta fase, deve-se ter um sistema de *software* documentado e funcionando corretamente em

seu ambiente operacional, de modo que já possa ser utilizado em modo de teste pelo usuário.

- **Fase de transição:** é a fase que se entrega o *software* ao usuário final para testes, juntamente aos manuais de utilização e instalação, se for o caso, fazendo com que sejam relatados possíveis defeitos e ajustes necessários. Na conclusão desta etapa, o *software* poderá operar em modo de produção.
- **Fase de produção:** disponibiliza o ambiente operacional ao usuário, monitorando o seu uso contínuo, por meio de suporte ao ambiente operacional, realizando alterações e correções se forem necessários.

No PU, a iteração ocorre de duas formas: entre as cinco fases e entre a própria fase, fazendo com que possam acontecer de maneira concorrente e escalonada. Pode-se retornar a cada uma dessas etapas caso seja necessário complementar e/ou corrigir algum requisito, ou definir uma nova integração, por exemplo, tornando-se incremental a cada etapa. Na figura 1 é demonstrado quatro das cinco etapas do processo unificado, segundo Bezerra (2007), que exibe fluxos de trabalho de um determinado projeto de *software*.

Figura 1 – Processo unificado



Fonte: Dias (2019)

2.2 Aplicação web

Uma aplicação web envolve diversas camadas da programação, lidando desde a interação com o usuário por meio da interface gráfica com o *frontend*, passando pela

manipulação dos dados com o *backend* e o seu armazenamento no banco de dados. Isto, deve ser precedido de um planejamento, explicado na seção 2.1.

2.2.1 *Frontend*

O *frontend* lida com a interface gráfica e apresentação dos dados ao usuário, ou seja, é onde inclui todo o design, estrutura, layout e o conteúdo das páginas que serão exibidas. O *client-side* deve conter boas práticas de IHC, utilizando-se de padrões na interface do usuário (*User-Interface* [1]) e fornecendo uma experiência relevante ao usuário (User-Experience [2]) para que o conteúdo seja disposto e interpretado pelo usuário da aplicação. (SOUTO, 2019)

2.2.1.1 Design responsivo

A união de técnicas de *grids* e imagens fluídos, que se adaptam a proporção da tela que está sendo utilizada, faz que o design responsivo se torne uma boa prática no desenvolvimento do *frontend*. A adaptação pode ser apenas o enquadramento de recursos, dada as proporções de altura e largura de uma tela, assim como a reestruturação do layout de forma que os conteúdos sejam exibidos de maneiras distintas nos mais diferentes dispositivos por meio do *media query*. (MDN contributors, 2022a)

2.2.1.2 Programação reativa

A programação reativa é um paradigma que utilizada de chamadas assíncronas para lidar com atualizações em conteúdos estáticos. Fornece um meio eficiente para atualizações de conteúdos utilizando gatilhos como referência (NOLLE, 2021). Segundo Nolle (2021), há três formas de ativar o gatilho:

- **Evento**: O evento refere-se a uma situação específica que ocorre via chamadas de *software*.
- **Chamada**: quando faz parte do fluxo da aplicação.
- **Mensagem**: unidade de informação que o sistema envia de volta ao usuário ou operador do sistema com informações sobre o status de uma operação, um erro, falha ou outra condição.

[1] É mais uma das ferramentas utilizadas para entregar a boa experiência ao usuário, pois é a parte tangível de um projeto, onde você constrói visualmente o site ou qualquer outra plataforma.

[2] É tudo que envolve o modo como qualquer usuário interage com o mundo ao seu redor. Na verdade, o termo *user experience*(UX) é muito amplo, mas quando falamos de marcas, produtos, sistemas e serviços, é importante entender que UX não envolve apenas o design do produto e seu desenvolvimento. Temos que observar todas as etapas do cliente junto à sua marca, desde o primeiro "encontro" até o pós-uso ou consumo.

Ao digitar em um campo de texto e o que foi escrito ser exibido, com suas edições, na interface da aplicação é um exemplo de programação reativa.

2.2.1.3 Interface e experiência do usuário

A experiência do usuário é o processo para criar produtos que forneçam experiências significativas e relevantes aos usuários (INTERACTION DESIGN FOUNDATION, S.D). De acordo com Norman e Nielsen (S.D), a definição de usabilidade é um atributo de qualidade na interface ao usuário, que seja eficiente de usar e agradável, sendo a experiência deste ainda mais ampla.

A experiência do usuário exemplar é atender as necessidades exatas de quem o utiliza, sem quaisquer complicações ou incômodos, podendo ir além com ofertas de simplicidade e elegância, que trazem alegria ao serem vistos e utilizados (NORMAN; NIELSEN, S.D).

Com o foco no usuário, o produto deve atender as necessidades dele, incluindo sua forma de interação com o sistema, no caso a interface gráfica. Segundo SEBRAE (2021), algumas das métricas para avaliar a experiência do usuário são: usabilidade, taxa de sucesso e taxa de erros.

2.2.2 *Backend*

Segundo a TOTVS (2020), a função do *backend* está relacionada com banco de dados, servidores, estrutura, gerenciamento de conteúdo e segurança. Podendo ser utilizado para o gerenciamento de login, nesta camada provém a proteção e segurança dos dados armazenados e no tráfego das informações da API. Os usuários não interagem diretamente com o *backend*, necessitando de interfaces para realizar tal manipulação (SOUTO, 2019). A escalabilidade e disponibilidade do sistema dependem desta camada, devido à sua gestão das informações contidas nela, sendo que estas podem estar disponíveis mundialmente.

2.2.2.1 API

A API é um conjunto de rotinas e padrões criados por uma aplicação visando a interoperabilidade entre aplicações. Podendo ser feita na camada de *backend*, uma API requer uma funcionalidade requisitada por um usuário no *frontend* para o *backend* processar os dados e os entregar seguindo um padrão por meio do protocolo HTTP[3] (PIRES, 2017).

[3] Protocolo cliente-servidor que permite obtenção de recursos, como arquivos JSON ou documentos HTML. É a base da troca de dados pelas aplicações *web* que suas mensagens são enviadas por meio de requisições e tem como retorno uma resposta (MDN contributors, 2022b).

2.2.3 Framework

Um framework é um conjunto de ferramentas de programação que fornece funcionalidades genéricas que podem ser alteradas conforme o uso específico. Fornecendo um padrão, é possível criar aplicações reunindo estruturas, bibliotecas de código e seu conjunto de ferramentas documentados que estão prontos para o uso do desenvolvedor.

2.2.4 Banco de dados relacional

Segundo Heuser (2009), um banco de dados relacional é composto por tabelas e relações. As tabelas são um conjunto não ordenado de tuplas, que possuem campos com suas propriedades. As relações ligam as tabelas entre si por meio de suas chaves.

2.3 Versionamento

De acordo com a IPSENSE (2020), o versionamento é uma metodologia aplicada por programadores visando controlar e acompanhar o histórico de alterações em um *software*, permitindo diferenciar mudanças realizadas em cada versão. A manutenção do código, ou seja, adicionar ou alterar informações do sistema para atender novas demandas, pode ser realizada através destas ramificações, diferentes instâncias de um mesmo programa, podendo uma instância conter determinadas alterações enquanto a outra se mantém estável sem as modificações (SACRAMENTO, 2021).

3 Materiais e Métodos

Neste capítulo serão apresentados os materiais e métodos utilizados para o desenvolvimento da aplicação. As linguagens de programação, os *framework*s, bibliotecas e suas especificações técnicas são apresentados em 3.1. Como serão utilizados tais materiais e os passos para o desenvolvimento da aplicação são descritos em 3.2.

3.1 Materiais

A seguir, serão descritos os materiais utilizados para o desenvolvimento da aplicação, que envolvem os *framework*s web *front* e *backend* do projeto e suas linguagens de programação, juntamente com a descrição das tecnologias do SGBD e versionamento.

3.1.1 HTML

O *Hiper Text Markup Language* (HTML) é uma linguagem de marcação utilizada para o desenvolvimento de páginas web na qual permite a inserção de conteúdo, estabelece a estrutura básica, assim como a organização de informações de uma página web. Isto o torna o componente básico da web, juntamente ao CSS e Javascript, sendo um conjunto de elementos conectados formando informações das mais variadas, podendo conter palavras, imagens, documentos entre outros tipos dados. Seu funcionamento se dá pela leitura de seus arquivos, que é renderizada para o usuário final por meio de um navegador web. Portanto, é o esqueleto no qual uma página se forma (FLANAGAN, 2013).

3.1.2 CSS

O *Cascading Style Sheets* (CSS) é a linguagem utilizada para estilizar o HTML. Todos os elementos podem ser estilizados, ou seja, é possível modificar detalhes como espaçamentos, cores, margens, fonte, tamanho desta fonte, bordas, efeitos visuais, entre vários outros detalhes para cada um. Desta maneira, é um complemento ao HTML, dando forma e modificação desta estrutura previamente feita (FLANAGAN, 2013).

Segundo Duckett (2011), o CSS pode ser reutilizado em diversas páginas criando padrões visuais a aplicação, assim como utilizado em áreas específicas. Torna as páginas, desde que usado para este princípio, mais responsivas ao organizar o layout e adaptar cada bloco visual para telas com diferentes tamanhos.

3.1.3 Javascript

Segundo Robbins (2018) Javascript é a linguagem de programação da web, interpretada pelos navegadores web. Diversos sites modernos utilizam esta linguagem para manipular o comportamento da página em conjunto ao HTML e CSS. Podendo ser utilizada tanto no *frontend* quanto no *backend* se torna muito versátil para o desenvolvimento destes sistemas. A mudança de dados sem a necessidade de recarregar toda a página permite uma flexibilidade na qual dados podem carregar independentes e de forma assíncrona sem que tenha a necessidade de um usuário final estar constantemente monitorando qualquer alteração (FLANAGAN, 2013).

Existem diversas bibliotecas para a linguagem com funções pré-prontas que visam a agilidade no desenvolvimento, que lidam, por exemplo, com a reatividade na página, como no caso do React.js. Os *framework*s têm incluso nas bibliotecas juntamente as suas regras de implementação, conforme visto no Next.js e sua estruturação das páginas contendo o React.js entre as suas bibliotecas (ROBBINS, 2018).

3.1.4 Typescript

O TypeScript é uma linguagem de programação de código aberto desenvolvido pela Microsoft e é um conjunto de ferramentas e boas práticas de programação para Javascript, o qual adiciona recursos a esta linguagem. Dentre as adições, vale ressaltar a tipagem estática, visto que Javascript não suporta isto (MICROSOFT, 2022). A orientação a objetos também é suportada pelo TypeScript, possibilitando a criação de interfaces para os objetos. Sua utilização pode ser atrelada ao Javascript, ou seja, podem ser utilizados em conjunto para uma maior flexibilidade (CHERNY, 2019).

3.1.5 PHP

O PHP é uma linguagem de *script open-source* de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML. Muito utilizada para o *backend*, esta linguagem tem evoluído rapidamente, sendo suportada por diversos desenvolvedores ao redor do mundo. Além disso, o PHP moderno engloba muitas práticas novas que podem ser desconhecidas para aqueles novos na linguagem ou para aqueles que estão atualizando de suas versões anteriores (LOCKHART, 2015).

3.1.6 Laravel

Framework é um conjunto de ferramentas, recursos e funcionalidades que visa facilitar e agilizar tarefas mais comuns no desenvolvimento de sistemas, sendo criado em uma determinada linguagem de programação. O *framework* escolhido foi o Laravel, feito

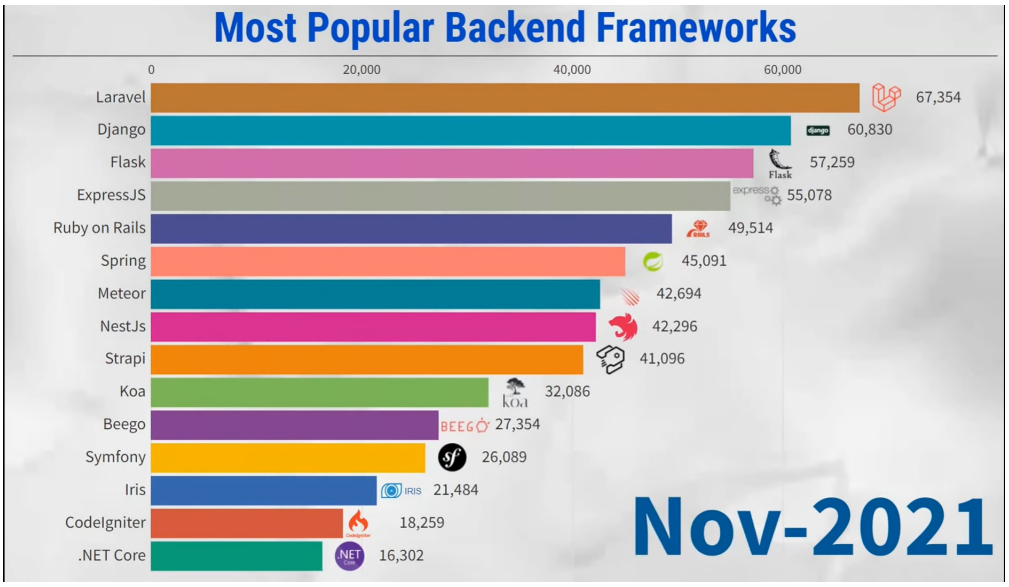
com base no PHP, é gratuito e possui código aberto. Este é utilizado no desenvolvimento de sistemas web e seu objetivo é fornecer código e recursos claros, simples e bonitos que ajudem os desenvolvedores a aprender, iniciar e desenvolver rapidamente e escrever um código simples, claro e duradouro (STAUFFER, 2019). A hospedagem do Laravel é suportada por diversos provedores, incluindo o Heroku, que será detalhado nesta seção.

O Laravel foi baseado no padrão de projeto Model, View, Controller (MVC), que divide a aplicação em três camadas, sendo estas:

- **Modelo:** dita as regras de negócio e os dados da aplicação.
- **Visão:** exibição dos dados para o usuário.
- **Controlador:** interpreta as entradas do usuário para as outras camadas de acordo com o que foi solicitado.

A escolha para o projeto se deve ao fato de que, conforme a figura 2, o Laravel é o *framework backend* mais popular até março de 2021 segundo dados dos repositórios com mais estrelas do GitHub, e isto se deve a diversas características contidas nele de acordo com Stauffer (2019):

Figura 2 – *Frameworks backend* mais populares



Fonte: *Statistcs and data*(Statistics and Data, 2021)

- **Roteamento rápido e simples:** é possível criar rotas com segurança e de forma simples, evitando requisições não autorizadas de maneira simples por meio dos verbos GET, POST, PUT e DELETE. Também é possível criar grupos de rotas com permissões de acesso específicas.

- **ORM Eloquent:** este facilita a interação com o banco de dados utilizando cada tabela como um objeto modelo correspondente. Outra facilidade é a forma de interação com as tabelas, que seriam a recuperação de registros do banco de dados, assim como sua inserção, atualização e exclusão.
- ***Migration*:** As *migration* são como o controle de versão do banco de dados, sendo estas as definições de criação e alteração do *schema*, podendo ser revertidas caso necessário.
- **Transmissão de eventos em tempo real:** utilizando da tecnologia de *WebSockets*, é possível atualizar em tempo real e constantemente a interface do usuário enviando dados sempre que determinado evento ocorra.

3.1.7 SQL

A linguagem SQL é utilizada para executar tarefas em tabelas de banco de dados. Ou seja, com os comandos de inserção, deleção, atualização e leitura de dados é possível manipular as informações por meio de comandos de consulta com diversas informações contidas (HEUSER, 2009). Suas principais características são:

- ***Query*:** é possível solicitar informações específicas de um banco de dados.
- **Manipulação de dados:** adicionar, excluir, alterar, ordenar, requisitar além outras operações para os registros contidos no banco de dados.
- **Acesso de controle de dados:** fornece técnicas de segurança para proteger dados, limitando quais usuários podem controlar e limitar o que podem fazer com os registros contidos no banco de dados.

3.1.8 Git

Controlador de versão é uma ferramenta de software na qual ajuda os desenvolvedores, seja em equipe ou não, a gerenciar alterações no código-fonte, mantendo o registro destas modificações. Então, o GIT é um sistema de controle de versão que possui código aberto, é flexível (possui vários tipos de fluxos de trabalho de desenvolvimento não lineares), é seguro (toda sua estrutura é protegida com um algoritmo de *hash* de criptografia seguro chamado SHA1) e possui um bom desempenho (Fazer o *commit* de novas alterações, *branches*, mesclagem e comparação de versões anteriores, tudo é otimizado para desempenho) (SANTACROCE, 2015).

3.1.9 Bootstrap

Para a estilização do *frontend* foi escolhido o Bootstrap, sendo um *framework* gratuito de HTML, CSS e Javascript no qual visa a criação de sites responsivos, ou seja, se adaptando tanto para telas maiores, como monitores de computador, assim como para telas menores, como os celulares. Além disso, contém bibliotecas prontas de estilização que procuram agilizar o processo de desenvolvimento no CSS e Javascript, assim como a reutilização do código e sua personalização (SPURLOCK, 2013).

Este possui um sistema de *grid*, que resultou em sua responsividade, e dimensiona a aplicação em até doze colunas de acordo com o tamanho de cada dispositivo. Além disso, possui componentes de interface que agilizam o desenvolvimento, visto que já estão implementados de forma que podem ser personalizados, sendo um exemplo a barra de navegação (*navbar*).

3.1.10 React

Para melhorar a interatividade da interface do usuário, foi escolhido o React, que segundo seus próprios criados é "uma biblioteca Javascript declarativa, eficiente e flexível para a criação de interfaces de usuário (UI)". Isso quer dizer que é uma coleção de funcionalidades que podem ser utilizadas e reutilizadas pelo desenvolvedor, sendo seus componentes. Isto permite uma padronização na interface, além de reaproveitamento de código (ZAMMETTI, 2020).

A criação de *views* é facilitada e simples, sendo estas atualizadas e renderizadas apenas para os componentes necessários à medida que os dados forem alterados, sendo estas declarativas fazendo que o código seja mais simples de se depurar.

Sua forma de trabalhar é com componentes encapsulados gerenciadores dos seus próprios estados, que combinados formam uma UI complexa, podendo ser escritos em Javascript ou TypeScript.

3.1.11 Next.js

O Next.js é um *framework* do React, o qual adiciona várias funcionalidades para esta biblioteca, visando acelerar o processo de desenvolvimento. Construído com intuito de melhorar o desempenho da aplicação implementada em React, há uma indexação do conteúdo da página e pelos motores de busca. Ou seja, se torna mais rápido e prático ao desenvolvedor, melhorando a experiência para o usuário final (KONSHIN, 2018).

As principais adições do Next.js em relação ao React segundo a VERCEL (2022) são:

- **Componente de imagem otimizado:** com este componente é possível redimensi-

onar, otimizar e exibir imagens no formato WebP (desde que o navegador suporte) evitando o envio de imagens grandes para dispositivos menores, como os celulares. E tal otimização é feita sob demanda, ou seja, é feita conforme os usuários solicitam, além de serem carregadas na página de forma assíncrona, não penalizando o carregamento do restante.

- **Roteamento por páginas:** dentro do diretório *pages* no projeto é possível, por meio de criação de demais pastas neste, rotas aninhadas, rotas dinâmicas e uma fácil integração entre estas. No primeiro caso, o diretório "pages/blog/index.js" é facilmente acessível digitando apenas /blog no final do endereço, e no segundo caso o caminho "pages/blog/[id].js" é acessível por qualquer valor no endereço após o caminho padrão, o blog, como, por exemplo: exemplo.com/blog/1.

- **Pré-renderização:** por padrão, o *framework* gera, para cada página, um HTML com um arquivo JSON de dados ao invés de ser tudo realizado pelo Javascript via client-side. E isto faz com que um motor de busca tenha melhor desempenho, dada a geração pré-renderizada, ou seja, preparado para melhorar os resultados de um SEO[1]. Portanto, cada página HTML terá o mínimo de código Javascript necessário, fazendo com que torne inteiramente dinâmica, por meio de sua execução somente quando necessário.

Dada a pré-renderização do Next.js, há duas formas as quais ele pode realizar tal recurso, podendo estas serem utilizadas em conjunto dada as necessidades de cada página do projeto. Uma das formas que pode ser utilizada é por meio da *Static Generation*, que recupera informação, caso seja necessário, de forma estática no momento de construção e gera uma página HTML que será colocada em *cache* no CDN. Isto faz com que o usuário final terá de esperar apenas o tempo de download e construção do HTML e não de todo conteúdo, previamente feito em Javascript. Em rotas dinâmicas, podem ser construídas com *cache* de um determinado período, ou seja, a cada um minuto esta página pode ser novamente alterada sua construção caso haja alguma requisição nesta.

A *Server-side Rendering*, que diferentemente da geração estática, a página HTML é gerada a cada requisição. Como a página não pode gerar um *cache* no CDN, será mais lenta, porém estará sempre atualizada no momento da requisição. Entretanto, por mais que seja gerada em tempo de execução, o processamento é similar ao anterior, visto que a página é renderizada no próprio servidor, enviando-se apenas o HTML para o cliente final, com as funções em Javascript necessárias, fazendo com que evite processamento no navegador do usuário, e fique apenas no servidor (VERCEL, 2022).

[1] Search Engine Optimization é utilizado para criar uma estratégia que irá aumentar a posição de um site no ranking nos resultados dos motores de busca. Quanto maior a classificação, mais tráfego orgânico para uma página terá.

A Vercel, criadora do Next.js, suporta todos os recursos da linguagem em sua hospedagem (mais detalhes a frente).

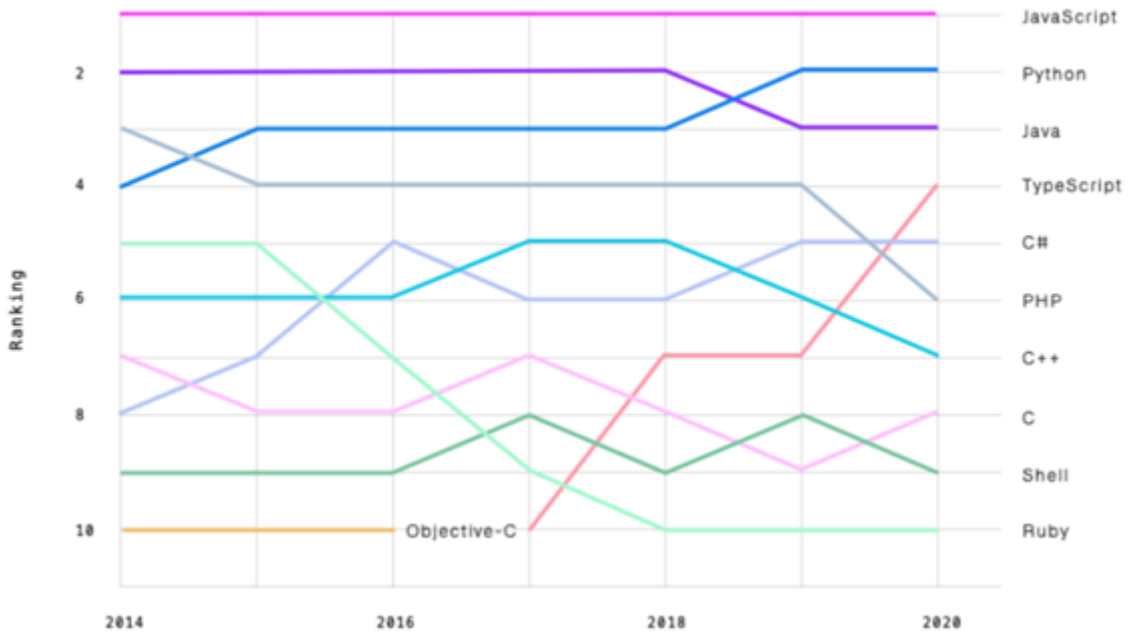
3.1.12 MySQL

Para o banco de dados, foi escolhido o MySQL, é um SGBD relacional de código aberto usado na maioria das aplicações gratuitas para gerir suas bases de dados que utiliza a linguagem SQL. Este possui um alto desempenho, sendo possível verificar uma abundância de dados em pouco tempo, gerando também a escalabilidade deste sistema (HEUSER, 2009).

3.1.13 GitHub

O GitHub é uma plataforma online onde são criados e hospedados repositórios GIT de projetos. Este sistema, visa facilitar o controle de versões de um software ou aplicação, funcionando em um formato de rede social para desenvolvedores (SANTACROCE, 2015). Além disso, é amplamente utilizada com mais de 56 milhões de desenvolvedores em 2020 conforme mostra a figura 3. Dito isso, esta plataforma será utilizada para manter as versões da aplicação no *frontend* e no *backend*.

Figura 3 – Utilização por linguagem do GitHub



Fonte: GitHub stats

3.1.14 Gitflow

Gitflow Workflow é um design de fluxo de trabalho Git que foi publicado e popularizado pela primeira vez por Vincent Driessen no nvie[2]. Este define um modelo de ramificação rigoroso, projetado com base no lançamento do projeto, oferecendo uma estrutura robusta para gerenciar projetos maiores. O Gitflow é eficiente para projetos que possuem um ciclo de lançamento agendado, visto que não adiciona novos conceitos ao fluxo de trabalho, mas sim define funções específicas, as ramificações e quando devem interagir (SANTACROCE, 2015). Ou seja, em conjunto com o versionamento do GitHub, será mais fácil localizar problemas, caso haja, assim como uma padronização em cada *commit* é também possível a verificação de que foi feito de uma forma mais simples e rápida.

3.1.15 Vercel

A Vercel é uma plataforma que hospeda aplicações web *frontend* gratuitamente por meio de um CDN[3] global. Com seus 50 domínios disponíveis e 100 GB de largura de banda, juntamente com sua criptografia SSL, fazendo a criptografia necessária dos dados para garantir a segurança. Além disso, há uma integração diretamente com o GitHub, que a cada *commit* na*branch* predeterminada é gerado um *deploy*, que atualizará o site com as últimas alterações. Sua escalabilidade é, segundo sua própria documentação, infinita, sendo que cada camada de sua infraestrutura aumenta e/ou diminui dinamicamente conforme o sistema e funções de armazenamento de *cache* necessários (VERCEL, 2022).

Internamente na plataforma da Vercel, junto ao o seu *framework* Next.js, há uma otimização de imagem, que será carregada do servidor segundo a área de visualização do dispositivo, guardando-as em *cache*. Esta otimização reduz o uso de banda, minimizando o tráfego de rede de acordo com o que for necessário para o usuário, melhorando então a experiência geral de quem visita à aplicação.

3.1.16 Firebase

Para o armazenamento das imagens, será utilizado o Firebase Cloud Storage. Este oferece uma forma prática de salvar e recuperar arquivos, incluindo uma das utilizações das imagens, gerando uma URL própria para uso. Contendo seu próprio sistema de regras de segurança e acesso como medida protetiva de seus arquivos, permite que apenas clientes autenticados se conectem. Com sua integração diretamente com o Javascript, apenas a URL no banco de dados será salva, portanto, indexada diretamente nos servidores da Google (FIREBASE, 2022).

[2] O nvie é um site pessoal de Vincent Driessen com seus pensamentos em formato de blog.

[3] *Central Delivery Network* (Rede de Distribuição de Conteúdo) é uma rede de servidores que armazenam réplicas do conteúdo em memória *cache* do site e as entrega aos usuários visitantes, de acordo com sua localização geográfica de modo que os conecte no servidor mais rápido e mais próximo, fazendo com que se reduza a latência.

3.1.17 Heroku

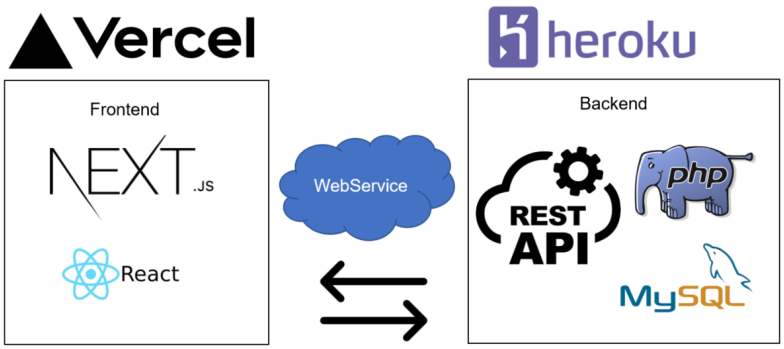
O Heroku é uma plataforma como um serviço, que permite hospedagem, além de configurações e publicações de projetos na nuvem e, diferentemente da Vercel, integra também diversos bancos de dados e possui suporte ao PHP. Com sua integração com o GitHub, seu *deploy* do projeto é feito a cada *commit* registrado no sistema, facilitando a integração. Além disto, esta plataforma é gratuita no seu início, contendo uma máquina virtual com 512 MB, podendo ser escalável à medida que for necessário com seus custos de infraestrutura.

Segundo a plataforma, é monitorado constantemente por múltiplos servidores a cada dependência, fazendo com que as correções sejam aplicadas assim que possível, tendo uma garantia ao usuário e seus servidores. Outro detalhe é que a estabilidade em sua plataforma também é garantida por eles, visto que seus algoritmos internos verificam caso ocorra uma queda ou falha na aplicação em tempo de execução e a restaura como uma nova instância automaticamente, garantindo uma preservação dos serviços executados pelo seu usuário sem a necessidade de verificar constantemente o que está ativo (HEROKU, 2022).

3.1.18 Tecnologias utilizadas

Exemplificando a organização das tecnologias utilizadas, a figura 4 demonstra a utilização e comunicação das tecnologias apresentadas nesta seção. No *frontend* está localizado na camada de visão da aplicação, que o usuário irá interagir, e então o *framework* Next.js, hospedado na Vercel que irá fazer as requisições ao *backend* por meio de um webservice hospedado na Heroku. O *backend* irá processar a requisição da API, que por meio do Laravel, fará as consultas ao banco de dados, retornando uma resposta com os dados requisitados ao *frontend* para serem exibidos ao usuário por meio da interface gráfica.

Figura 4 – Organização das tecnologias utilizadas



Fonte: Próprio autor

O fraco acoplamento entre as tecnologias demonstra a divisão estruturada entre as funcionalidades. Ou seja, enquanto a Vercel está lidando apenas com a renderização de páginas e compilação do Javascript por meio do Next.js, o servidor da Heroku está disponível apenas para consultas via API construída no Laravel, que pode também servir a outra aplicação, podendo esta ser mobile, por exemplo. E, seguindo esta distribuição vertical, conforme Coulouris et al. (2013) há ganho de desempenho, melhorando-se os tempos de carregamento das páginas.

3.2 Métodos

Para este projeto, o desenvolvimento foi baseado no processo unificado, que as etapas são incrementais e interativas, mesmo sendo definido por etapas (SOMMERVILE, 2011). As etapas são definidas por: concepção, elaboração, construção e transição.

Há cinco atores na aplicação, sendo estes:

- **Visitante:** não possui cadastro no sistema ou não entrou em sua conta ainda, portanto seu processo estará pré-definido de forma que visualize os eventos, mas não poderá participar destes. Além disso, poderá criar uma conta para que possa fazer login como usuário e por fim poderá validar os certificados de terceiros, caso tenha a chave necessária.

- **Usuário:** interage com o sistema de modo que poderá gerenciar seu perfil, efetuando alterações em dados de seu cadastro, como seu telefone. Além disso, poderá participar de eventos, os quais irão gerar certificados que poderão ser convertidos para arquivos e salvos no computador, além de serem enviados por e-mail caso tenham participado do evento. Vale ressaltar que estas ações só poderão ser realizadas após a realização do login. E para ser feito isto, é necessário que seja feito um cadastro na aplicação utilizando-se de dados pessoais tais como: nome completo, e-mail, CPF e telefone para serem gerados os certificados corretamente.

- **Apresentador:** tem como principal função, podendo ou não ter um cadastro na aplicação, ser vinculado a um evento o qual irá apresentar. Ou seja, supondo que seja uma palestra, este será o palestrante e terá acesso ao seu certificado por meio do e-mail cadastrado no evento. Caso já tenha um cadastro, este será vinculado diretamente a conta e poderá ser impresso (sem a necessidade de consulta ao e-mail) diretamente da aplicação.

- **Associado:** tem acesso a tudo que um usuário faz, mas poderá gerir até certo nível a instituição, fazendo a criação de eventos acadêmicos. Para que alguém se torne associado, é necessário que o supervisor o vincule a uma instituição. Sua principal

função é gerir os eventos criados pelo próprio autor, de forma que serão vinculados a este por meio da instituição.

- **Supervisor:** tem a função de criar e gerir a instituição, além de gerir todos os eventos pertencentes a esta.

3.2.1 API

A API para a aplicação foi feita em Laravel com utilização do SGBD MySQL por meio de uma autenticação segura. As consultas e requisições ao banco de dados serão feitas nesta camada. E seu retorno dos dados requisitados serão transmitidos ao módulo de visão via JSON.

Além disso, a autenticação da aplicação é validada por meio do JWT[4], ou seja, para cada requisição que exija segurança, como, por exemplo, a criação de um evento ou alterações na conta do usuário, que dizendo, de forma geral, qualquer alteração solicitada ao SGBD. Isto garantirá que o usuário está devidamente conectado, assim como que sua autenticidade.

3.2.2 Frontend

O *frontend* foi feito em Next.js, na qual irá receber os dados referentes a API e convertê-los para uma interface ao usuário. Esta interface é um site no qual, usuário poderá interagir e realizar todas as ações permitidas. Além disso, este módulo feita de forma que seja simples, fácil, empregando conceitos de usabilidade para uma melhor experiência.

Para evitar fraudes com bots, foi necessário utilizar um sistema que verifique isto, principalmente na criação de usuários, assim como no login destes, sendo de onde vem a ideia do CAPTCHA[5]. A implementação utilizada foi o hCAPTCHA[6], o qual irá proteger o sistema de login da aplicação, dificultando o acesso para máquinas automatizadas.

3.2.3 Versionamento

O versionamento foi feito utilizando o Git, por meio de um repositório no Github. Cada versão foi feita com base nos critérios do GitFlow, seguindo a lógica descrita de

[4] JSON Web Token (JWT) é um padrão aberto (RFC 7519) que define uma maneira compacta e independente para transmitir informações com segurança entre as partes como um objeto JSON. Essas informações podem ser verificadas e confiáveis porque estão assinadas digitalmente.

[5] O CAPTCHA (Teste de Turing público completamente automatizado para distinguir entre computadores e pessoas) é um tipo de medida de segurança conhecido como autenticação por desafio e resposta. O CAPTCHA protege contra spam e descriptografia de senhas com um teste simples que prova que você é um ser humano, não um computador tentando invadir uma conta protegida por senha.

[6] O hCAPTCHA é um serviço de CAPTCHA gratuito e independente que fornece desafios fáceis para humanos, entretanto são complexos para uma máquina, tais como classificação de um objeto dada as imagens.

master e *develop*, sendo a primeira a versão de produção e a outra sendo a versão com as novas funcionalidades e para testes. Cada parte da aplicação terá seu versionamento de acordo com sua funcionalidade, ou seja, a API terá seu versionamento separado do *frontend*.

4 Projeto e Desenvolvimento

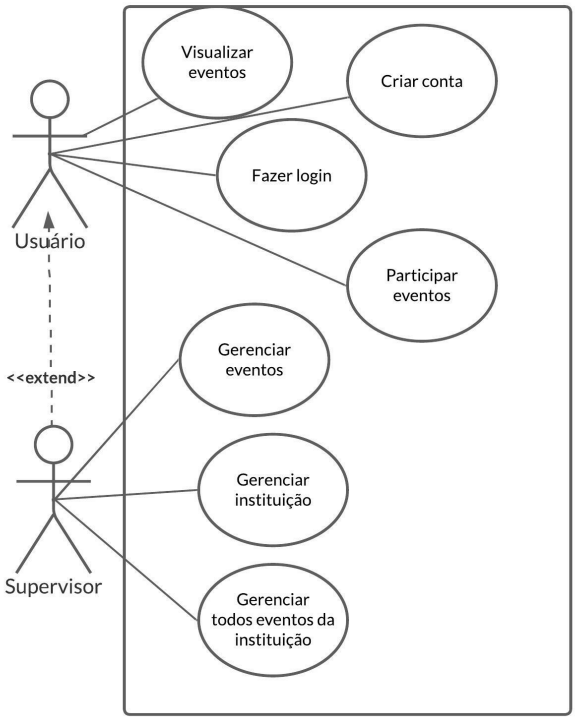
Nesta seção são detalhados os passos e decisões tomadas ao decorrer do período de desenvolvimento deste trabalho. São apresentadas as etapas de implementação e integração com outras soluções utilizadas, assim como a execução e comparação dos resultados obtidos.

4.1 Casos de uso

Inicialmente foi desenvolvido o caso de uso expandido do sistema, baseando-se no que foi demonstrado por Sommervile (2011) do modelo UML, como forma tanto de documentação, assim como planejamento do protótipo. Todas as interações do sistema com seus usuários foram pré-determinadas, assim como a definição e organização de requisitos funcionais no sistema dado seu contexto para modelagem do fluxo básico dos eventos diretamente nestes diagramas. O caso de uso mostra, neste contexto, o escopo da aplicação, trazendo as principais funcionalidades desenvolvidas no protótipo, limitando-se a eventos educacionais online. Os cenários de interação entre os atores no diagrama de caso de uso determinado definem principalmente como deveria ser planejado e até qual ponto pode ser utilizado por cada um deles, respeitando sua hierarquia. A figura 5 demonstra o primeiro caso de uso planejado para o protótipo.

Com base nesse diagrama, e sua devida explicação, foi possível traçar uma rota de desenvolvimento por meio do processo unificado. No início foi pensado em apenas dois atores, sendo estes um administrador e um usuário comum, como mostra a imagem. Nisto, foram desenvolvidas funcionalidades básicas para estes, contendo a primeira iteração, sem os demais.

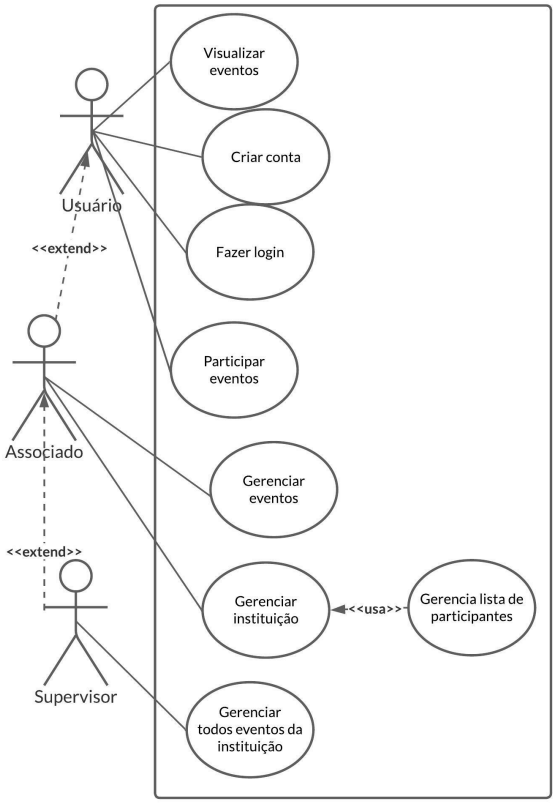
Figura 5 – Primeiro caso de uso



Fonte: Próprio autor

Para a segunda iteração, já há mais atores contando com a inclusão do associado, sendo intermediário entre usuário e supervisor. Além disso, e novas funcionalidades pensadas para a aplicação, sendo estas a gerência da lista de participantes e a gerência dos eventos apenas para determinado usuários (sendo estas relacionadas ao associado) conforme demonstrado na figura 6.

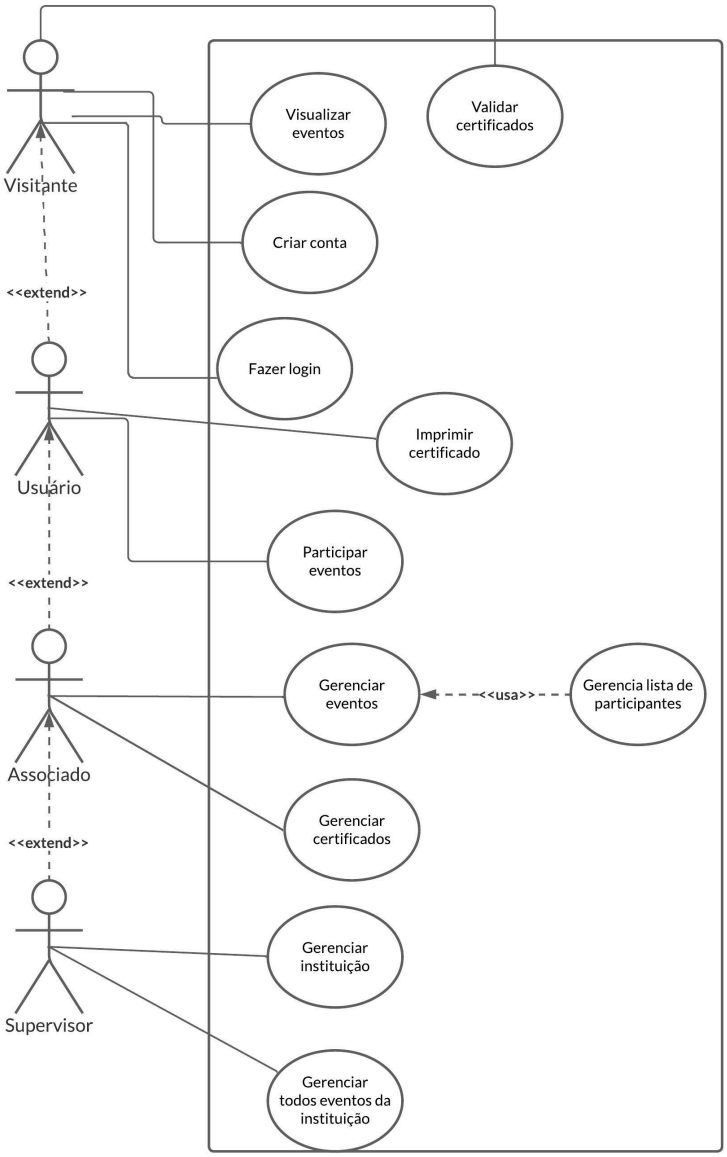
Figura 6 – Segundo caso de uso



Fonte: Próprio autor

Na terceira iteração foram identificadas necessidades de atores que não precisavam necessariamente de um *login*, como o visitante, que poderá apenas visualizar os eventos e validar certificados, operações mais simples e que não necessitam de qualquer permissão especial conforme o que é exibido na figura 7.

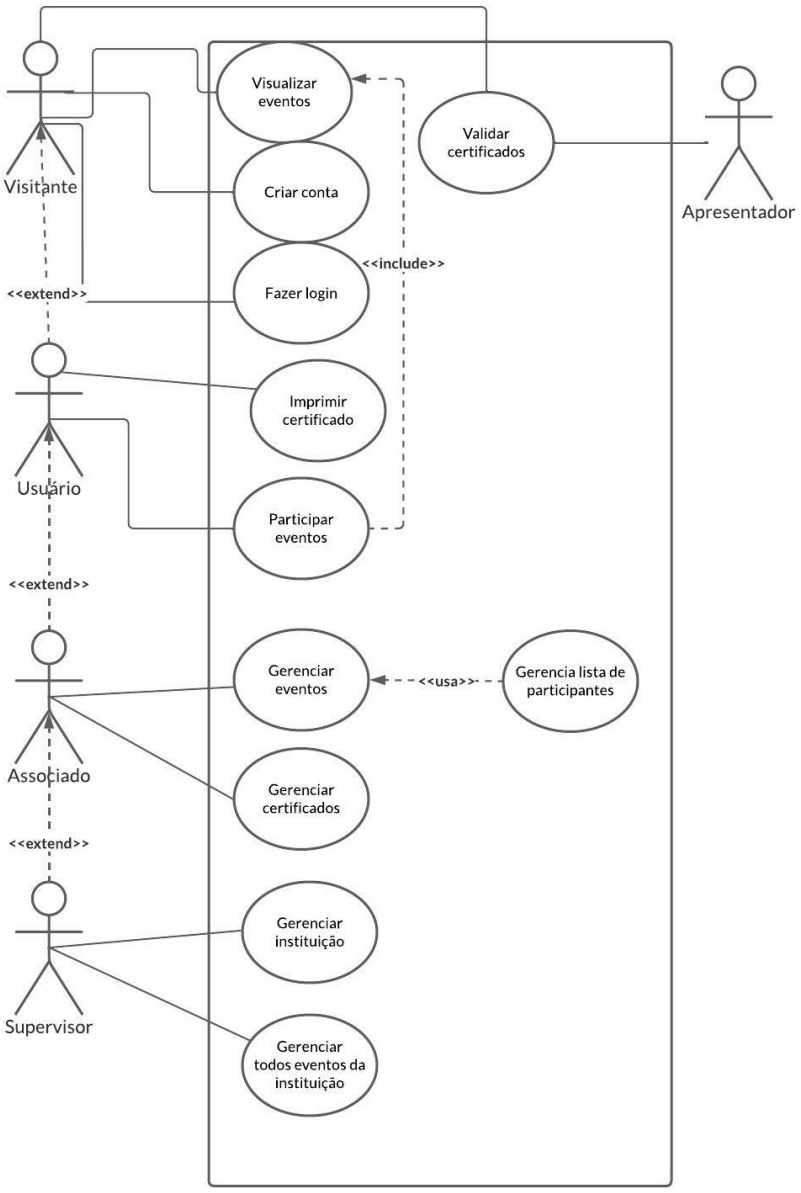
Figura 7 – Terceiro caso de uso



Fonte: Próprio autor

Na quarta e última iteração até o momento o sistema está mais moldado pensando em diversas funcionalidades, inclusive com o caso de uso expandido, tendo em vista melhor as funcionalidades e seus retornos do sistema para o ator determinado conforme a figura 8.

Figura 8 – Quarto caso de uso



Fonte: Próprio autor

4.1.1 Caso de uso expandido

Segundo Wazlawick (2010), o caso de uso expandido corresponde ao aprofundamento da análise de requisitos, ou seja, descreve-se mais detalhadamente os passos para cada caso de uso documentado, conforme demonstrado abaixo:

- CSU01: visualizar eventos
  - Descrição: O sistema apresenta uma lista de eventos e o usuário pode selecionar um evento para visualização mais detalhada.

- CSU02: Criar conta
  - Descrição: O usuário informa o nome, e-mail, telefone, CPF e senha. O sistema valida as informações digitadas e cria a conta.
  - Pós-condições: Um e-mail é enviado ao usuário para verificação.
- CSU03: Fazer login
  - Pré-condições: O usuário deve ter criado a conta anteriormente com o e-mail validado.
  - Descrição: O usuário informa o e-mail e senha. O sistema valida as informações digitadas e realiza o login.
  - Pós-condições: O sistema deixa salvo as informações para que, na próxima entrada do usuário, não seja necessário digitar novamente os dados.
- CSU04: Imprimir certificado
  - Pré-condições: O usuário deve ter participado anteriormente de um evento.
  - Descrição: O participante ou apresentador geram o arquivo PDF do evento selecionado.
- CSU05: Participar eventos
  - Pré-condições: O usuário deve ter efetuado o login e selecionado um evento previamente.
  - Descrição: O participante seleciona as atividades de um determinado evento e participa destas.
  - Pós-condições: O associado ou supervisor podem selecionar quem foi ao evento para geração do certificado.
- CSU06: Gerenciar eventos
  - Pré-condições: O usuário deve ter feito login com permissão de associado ou supervisor e selecionar um evento que ainda não ocorreu nenhuma de suas atividades.
  - Descrição: O usuário pode editar, excluir eventos já cadastrados no sistema. Na edição será possível alterar atividades, descrição e apresentadores.
- CSU07: Gerenciar certificados
  - Pré-condições: O usuário deve ter efetuado login com permissão de associado ou supervisor.

  - Descrição: O usuário pode criar, editar ou excluir modelos padrões de certificado. Para criação é necessário que informe a imagem de fundo, banner e assinatura, um título e nome e cargo de quem irá assinar.
  - Pós-condições: o sistema irá gravar as informações e poderão ser utilizadas para qualquer certificado de evento.

- CSU08: Gerenciar instituição
  - Pré-condições: O usuário deve ter efetuado login com permissão de supervisor.
  - Descrição: É possível alterar dados da instituição como seu nome, o supervisor e gerenciar os associados. Ao alterar o supervisor ou adicionar um associado, o sistema verifica se é um usuário válido.

- CSU09: Gerenciar todos os eventos da instituição
  - Pré-condições: O usuário deve ter efetuado login com permissão de supervisor.
  - Descrição: O supervisor pode alterar qualquer evento da instituição, sendo ou não criado por seu usuário.

- CSU10: Validar certificados
  - Descrição: O usuário informa o código de verificação do certificado.
  - Pós-condições: O sistema retorna se o certificado é válido ou não.

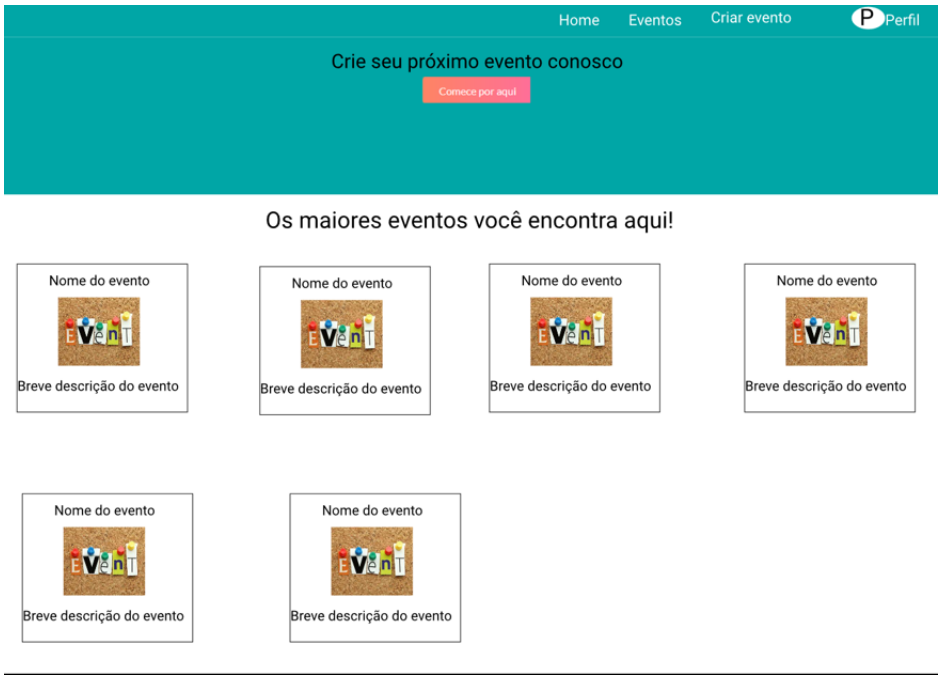
- CSU11: Gerencia lista de participantes
  - Pré-condições: O usuário deve ter efetuado login com permissão de supervisor ou associado e selecionado um evento.
  - Descrição: O usuário pode selecionar quem participou do evento, seja participante ou apresentador para gerar um certificado.
  - Pós-condições: O certificado em formato PDF é enviado ao e-mail dos participantes e apresentadores selecionados.

4.2 Mockups

Dados os casos de uso, será possível criar um mockup das telas baseado nestas informações. Ou seja, a criação de telas por meio do Figma possibilitou a visualização e alteração do layout de uma forma simplificada antes de serem implementadas de fato. Dito isso, todo o mockup foi pensado para seguir padrões na interface, como, por exemplo, a barra de navegação está presente em todas as páginas a qual o usuário poderá navegar (com exceções apenas da criação de conta e login). E isto pode facilitar a navegação, mantendo a similaridade da aplicação para o usuário final.

A figura 9 demonstra como é a ideia da página inicial, que contém uma barra de navegação, um acesso aos eventos, um acesso à criação de eventos e ao perfil. Haverá, no meio da página, um destaque aos últimos eventos cadastrados na plataforma, podendo ser acessados diretamente desta página.

Figura 9 – Mockup da tela inicial

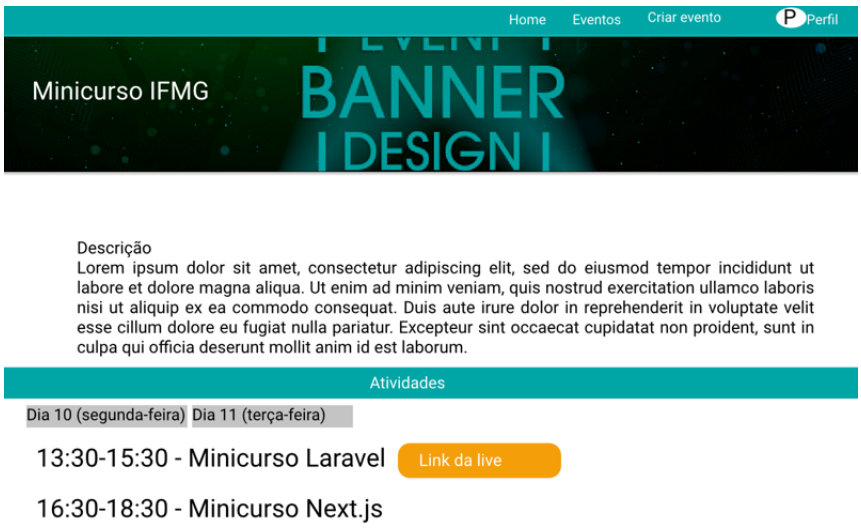


Fonte: Próprio autor

Para a página seguinte, a figura 10 mostra como é a visualização detalhada de um evento, com sua descrição, imagem e suas atividades cadastradas. Estes detalhes são os compostos especificamente para um evento, mas a barra de navegação citada anteriormente na página inicial estará presente para facilitar a navegação entre páginas.

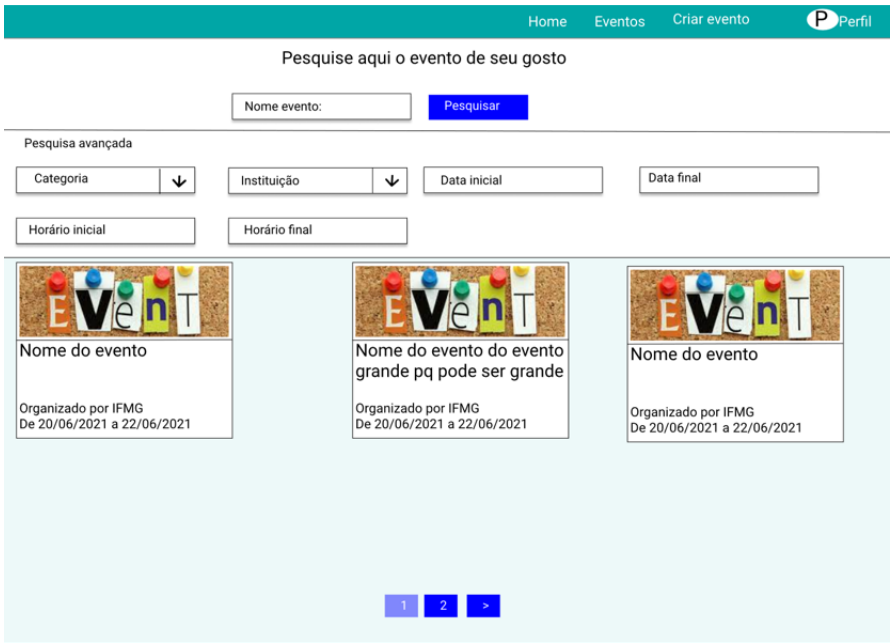
A terceira parte do mockup é a tela de pesquisa de eventos, como mostra a figura 11, que assim como as demais mantém a barra de navegação para facilitar o acesso e possui diversos filtros para os eventos. Estes filtros, definidos por categoria, instituição, data inicial e final e horário inicial e final, foram definidos como os principais para a pesquisa. E ao final da página, percebe-se a paginação, ou seja, caso passe de uma determinada quantidade, serão criadas diversas abas possíveis da mesma pesquisa.

Figura 10 – Mockup da tela do evento



Fonte: Próprio autor

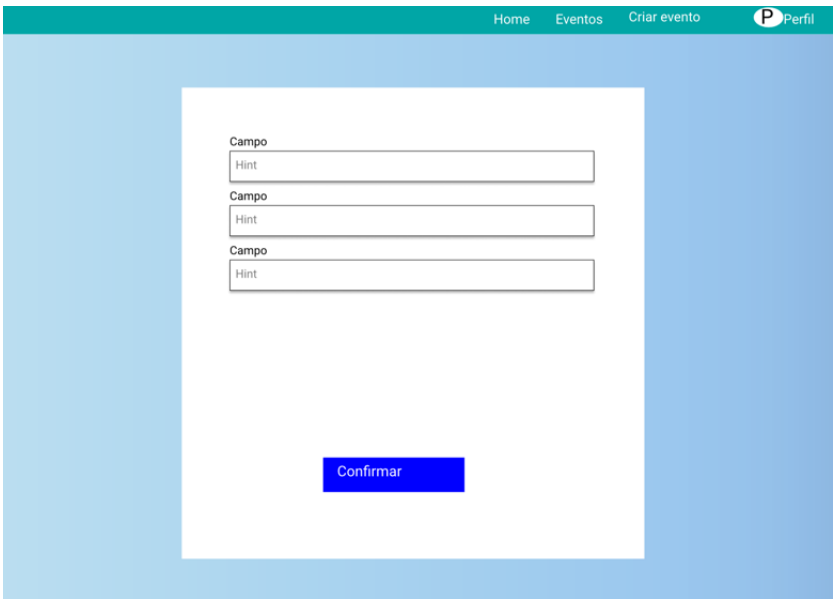
Figura 11 – Mockup da tela de pesquisa



Fonte: Próprio autor

Por fim, a figura 12, mostra como serão todos os formulários do sistema, sendo este utilizado conforme a necessidade da tela, como a criação de eventos ou criação de usuário.

Figura 12 – Mockup da tela de formulário padrão



Fonte: Próprio autor

Com estas informações sobre a responsabilidade e design de cada tela da aplicação, é possível a criação de maneira estruturada, além de padronizada da interface para o usuário.

4.3 Estruturação

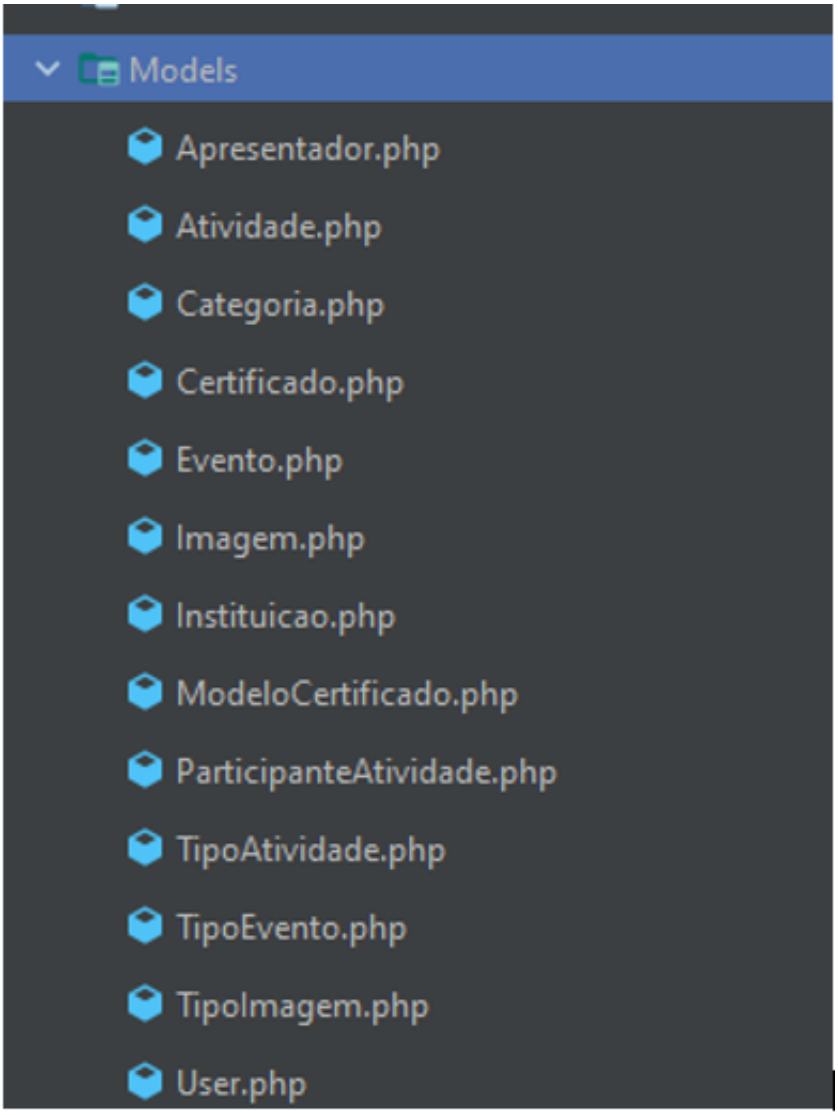
A estruturação segue um modelo de distribuição vertical, o qual prevê a distinção entre divisão de componentes, neste caso a interface com o usuário e a API, em dois servidores distintos, sendo um alocado na Vercel e outro no Heroku. Mesmo que uma aplicação dependa da outra para funcionar integralmente, são serviços distintos e que melhoram de maneira geral o funcionamento do sistema, visto que tais responsabilidades distintas melhoram o tempo de carregamento de acordo com Coulouris et al. (2013). Ou seja, enquanto *backend* persiste os dados e os processa diretamente, respondendo a requisições, o *frontend* fará as requisições recuperando o arquivo JSON gerado e o transformando em uma interface gráfica tangível ao usuário.

4.4 Modelo de banco de dados

Nesta etapa foram criados modelos os quais tiveram embasamento no diagrama de Caso de Uso, assim como nos mockups, visto que devem atender os requisitos predeterminados. Foram criados os modelos conforme figura 13, que determina cada classe do banco de dados feita no Laravel para poder representar os relacionamentos entre as tabelas do

MySQL, além de determinar cada atributo das classes. Estes foram feitos com base na figura 14, que representa o diagrama de entidade e relacionamento do banco de dados.

Figura 13 – Classes de modelo

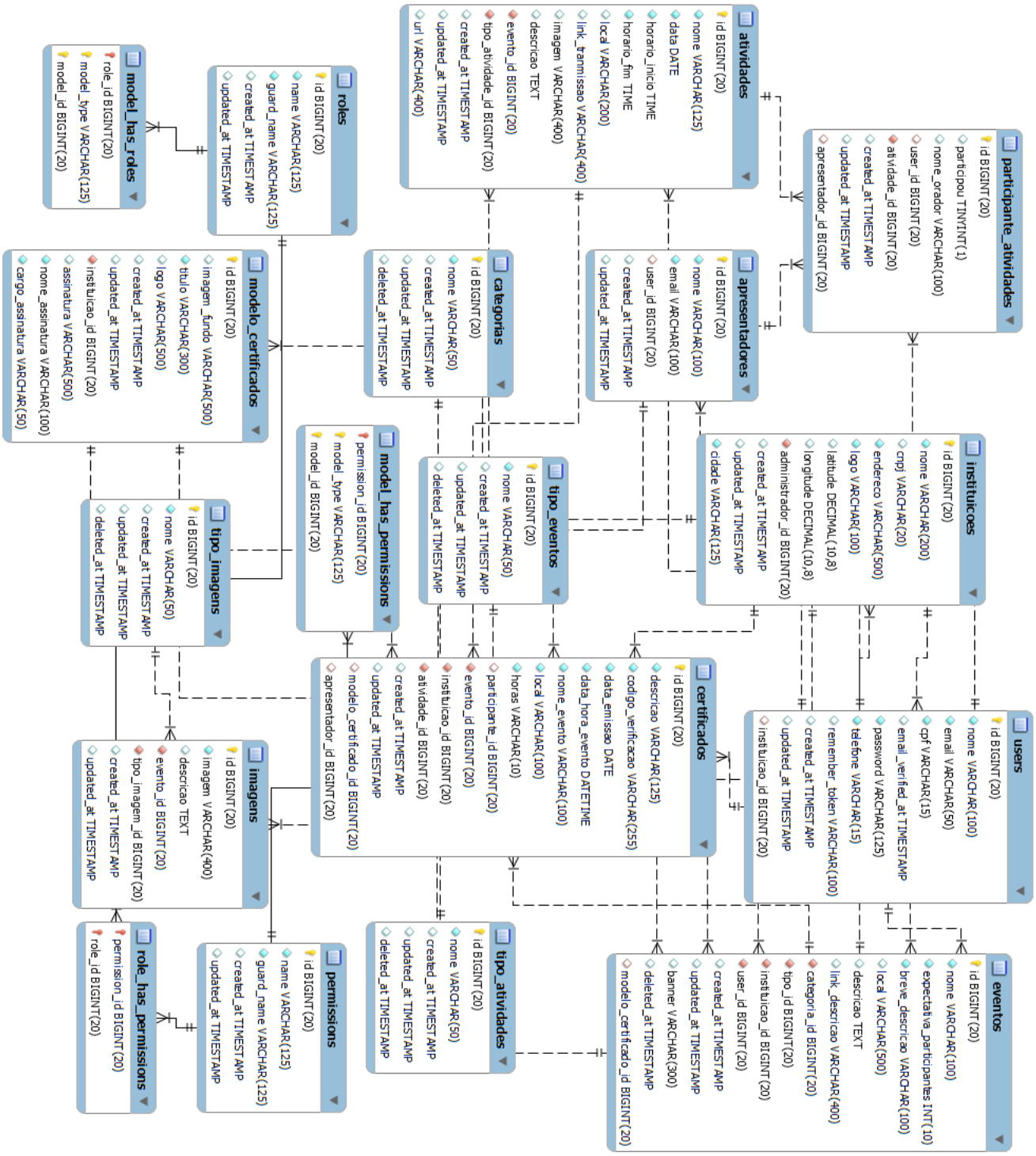


Fonte: Próprio autor

O banco de dados foi planejado criando-se a partir das funcionalidades da aplicação. As tabelas são descritas abaixo:

- ***Users*:** tabela de usuários, com informações básicas do cadastro, contendo nome, e-mail, CPF, senha e telefone. O campo senha é criptografado de modo que tenha segurança ao gravar os dados. Os campos e-mail e CPF devem ser únicos no banco. E há um campo de chave estrangeira, relacionado a instituição, podendo ser nulo, que caso o usuário esteja vinculado a uma instituição, será atribuído o id a ele.

Figura 14 – DER do banco de dados



Fonte: Próprio autor

- **Instituicoes:** tabela que contém as informações referentes as instituições, tais como o nome, CNPJ, endereço e cidade. O campo CNPJ deve ser único. A chave estrangeira administrador_id é relacionada ao usuário que criou ou foi transferida a administração da instituição.

- **Categorias:** tabela criada com intuito de categorizar os eventos de modo que seja facilitada a busca na aplicação.

- **Tipo_Eventos:** tabela criada com intuito de sub-categorizar os eventos de modo que seja facilitada a busca na aplicação

- **Tipo_Atividades:** tabela criada com intuito de categorizar as atividades de modo que seja facilitada a busca na aplicação.

- **Eventos:** tabela que possui os dados do evento, como seu título, expectativa de participantes, banner e uma breve descrição como obrigatórios. Os campos opcionais, como local, descrição podem ser adicionados depois, não sendo obrigatórios. O campo banner é uma URL da imagem armazenada no Firebase. As chaves estrangeiras são definidas pela categoria_id, tipo_id (ambas referentes às categorias e tipos escolhidos pelo usuário), instituicao_id e user_id (ambas referentes ao usuário que criou o evento) e chave modelo_certificado_id (não obrigatória para criação do evento, mas obrigatória para criação dos certificados posteriormente).

- **Modelo_Certificados:** a tabela para os modelos de certificado deve conter os campos de título, nome_assinatura e cargo_assinatura como obrigatórios, sendo textos. Os campos de imagem_fundo, assinatura e logo devem ser preenchidos com as URLs das imagens armazenadas no Firebase. A chave estrangeira instituicao_id se refere a instituição que criou tal modelo.

- **Apresentadores:** os apresentadores podem ser externos à aplicação, então contêm um nome e e-mail. Caso o e-mail seja vinculado a um usuário, será atribuído um id de usuário nesta tabela.

- **Tipo_Imagens:** descreve o tipo de imagem que está no evento, podendo ser banner ou outros.

- **Imagens:** tabela que armazena a URL da imagem contida no Firebase, com as chaves estrangeiras de seu tipo e do evento que foi vinculada.

- **Atividades:** as atividades são descritas pelo seu nome, a data que irá ocorrer, assim como horário de início e término como obrigatório. Descrição e local podem ser inseridos posteriormente. A URL também pode ser inserida posteriormente, mas é necessário que tenha a URL para que a atividade ocorre, sendo que é com base nela

que os participantes entraram. As chaves estrangeiras são: evento_id, referenciando o evento da atividade e a tipo_atividade_id que é sobre o tipo de atividade selecionado.

- **Certificados:** esta tabela descreve todas as informações presentes do evento, atividade, data, e as horas da atividade. Deve conter um modelo de certificado para poder se basear as informações. O certificado pode ser para um participante, que deve ser usuário da aplicação, ou um apresentador, podendo este ser externo a aplicação.

4.5 Controllers

Por meio dos controladores do Laravel, é possível manipular as operações do banco de dados, e para haver uma organização e mantenha o padrão MVC, foram criados controladores específicos para cada modelo, sendo cada um atribuído a responsabilidade do modelo correspondente.

Além disso, por fazerem parte de uma API (explicada com mais detalhes posteriormente), cada operação deve retornar um JSON, com o código de status de retorno do HTTP. Um exemplo possível nos controladores, são suas função que recebem uma requisição com determinados filtros e então, por meio do *eloquent ORM*, realiza a *query* no banco de dados e no fim retorna um JSON com os dados, padronizando, então, a maneira de se receber e retornar os dados.

Dado o exemplo da busca e seus filtros, a validação de dados na inserção e atualização de dados é essencial para não haver dados inseridos de maneira indevida e que haja um padrão nas requisições. Isto tudo é feito diretamente no Laravel, que validará antes mesmo do banco de dados e retornará caso haja algum erro.

4.6 Operações API

A API foi feita no Laravel, respeitando os padrões estabelecidos pela interface REST. Ou seja, por meio de requisições HTTP do tipo GET, POST e DELETE é possível acessar todas as rotas do *backend*. Em diversas destas não será possível acessar sem que haja uma autenticação feita por meio do JWT, que irá validar se há permissão de acesso, e isto determina caso o usuário esteja logado, assim como seu nível de acesso dentro de sua hierarquia.

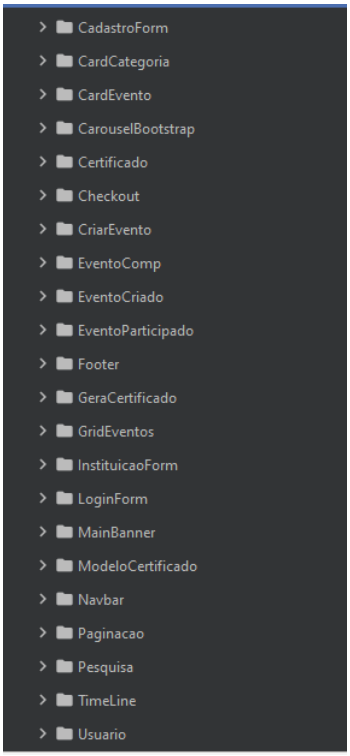
As rotas foram criadas em grupos para atenderem as necessidades e criar-se uma organização para cada requisição. Há agrupamentos para cada módulo da aplicação, sendo aqueles que contém todas as rotas para os modelos, assim como para determinadas funcionalidades do sistema. O grupo de eventos, por exemplo, tem todas as rotas pertencentes a eventos e possui autenticação e validação do que pode ser escrito na URL para poder ser acessada devidamente pelo *frontend*.

4.7 Componentes

Diferentemente do *webservice* criado no Laravel, a estruturação do *frontend* no Next.js é feito por meio de componentes, e por utilizar o Typescript, há a tipagem em variáveis que auxilia o desenvolvedor nos parâmetros a serem passados entre os componentes do projeto.

Por meio dos componentes, mostrados na estrutura da figura 15 é possível a reutilização do código-fonte para diversos layouts, como, por exemplo, o componente NavBar, o qual é utilizado em diversas páginas, tornando-as padronizadas. Com isto, cada componente, também, terá seu estilo por meio de um arquivo de módulo do CSS, que estende os estilos globais, podendo sobrepô-los. Portanto, cada componente terá sua finalidade pré-definida, podendo ser reutilizada caso sirva o propósito.

Figura 15 – Estrutura dos componentes



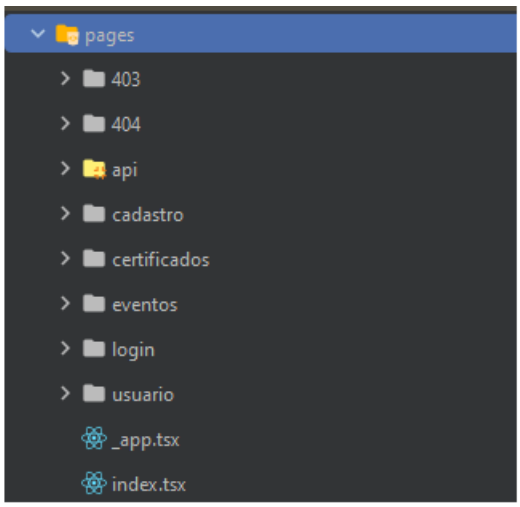
Fonte: Próprio autor

4.8 Rotas Next.js

Em relação às rotas, serão as páginas a serem acessadas pelo usuário. Dito isso, estas deverão validar, caso necessário, se o usuário está logado por meio de uma requisição ao *backend*. Além disso, estas devem requisitar diversas informações para poderem ser exibidas nas páginas e terem os dados necessários preenchidos de cada componente. Conforme

estruturação presente na figura 16, cada página tem seu nome e sua estruturação interna, podendo ter sub-rotas, que serão chamadas dentro da pasta. Esta regra vale para todas as pastas presentes, com exceção das 404 (a qual está destinada a páginas não encontradas, ou seja, caso seja digitada uma *url* dentro do domínio que não seja pertencente a esta lista, irá a chamar, visto que é uma página customizada) e a 403 (que determina um recurso não possível de ser acessado para determinado usuário).

Figura 16 – Estruturação de páginas no Next.js



Fonte: Próprio autor

4.9 Renderização das páginas

O método de *Static Generation* foi utilizado nas páginas que não precisam de qualquer atualização do banco de dados. Isto causa mais rapidez na navegação, devido a sua geração estar envolvida apenas um arquivo de texto HTML, e não precisar esperar a compilação do JavaScript. A página utilizada para criar eventos foi feita com base neste modelo de renderização, visto que não haverão mudanças de dados a medida que o sistema for utilizado (sendo a única exceção ao gerar uma nova versão da aplicação).

Há também a outra implementação de pré-renderização, sendo a *Server-side rendering*, utilizada em páginas, como a de pesquisa dos eventos. A sua utilização se deve ao fato de que esta deve ser geradas constantemente baseada nos filtros que o usuário irá fazer. Por estar hospedada apenas no servidor, a página terá uma lentidão se comparada as que utilizam os métodos estáticos, mas garante que as informações do banco de dados estarão atualizadas para exibição do usuário a cada requisição feita.

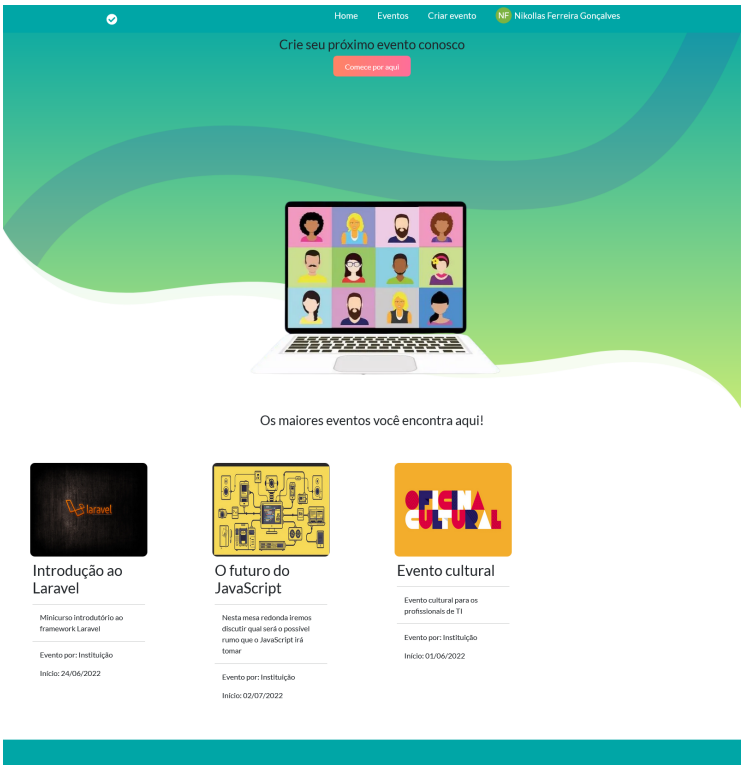
Nas páginas que não precisaram estar constantemente atualizadas, foi utilizado o recurso do Next.js de *Incremental Static Regeneration*, que guarda seus dados como armazenamento temporário (gerando um *cache*). Um exemplo são as páginas de cada evento,

que seguem o requisito de atualização não tão frequente, sendo geradas sob demanda. Ou seja, por todas as páginas serem baseadas nos mesmos componentes, diferindo apenas nos dados requisitados, foi criado um *cache* para cada um, gerando uma página HTML com um JSON de dados, para cada uma das páginas. Isto faz com que o acesso seja mais rápido, e o *cache* é de um minuto, ou seja, a cada um minuto se o evento for editado e caso haja uma requisição, a página será gerada novamente com base no que foi atualizado em cada uma das páginas desta rota dinâmica.

5 Resultados e discussões

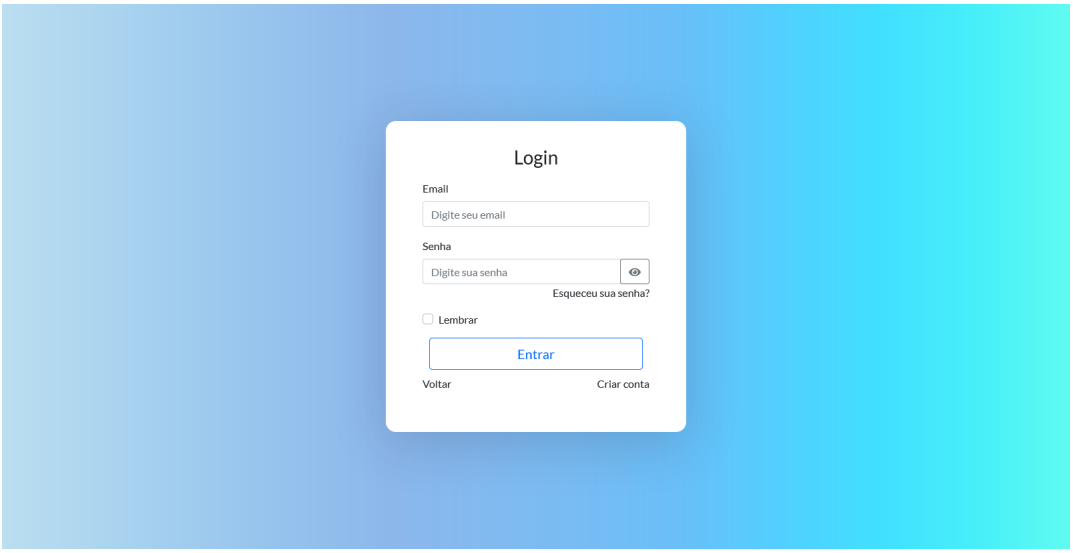
Com a utilização de *frameworks* modernos, foi possível criar uma aplicação web responsiva e fluída. Ao se comparar com o estado inicial, os mockups seguiram seus propósitos e com base neles, foi possível a criação de todas as telas do sistema, conforme visto no capítulo 5. A figura 17 demonstra a tela inicial, que todas as pessoas irão visualizar ao entrar na aplicação. Comparada ao mockup, há uma maior riqueza de detalhes, assim como melhorias que puderam ser ajustadas. Seguindo para as telas de cadastro, como a de Login na figura 18, o fundo foi levemente alterado em sua paleta para corresponder ao que foi gerado anteriormente.

Figura 17 – Página inicial



Fonte: Próprio autor

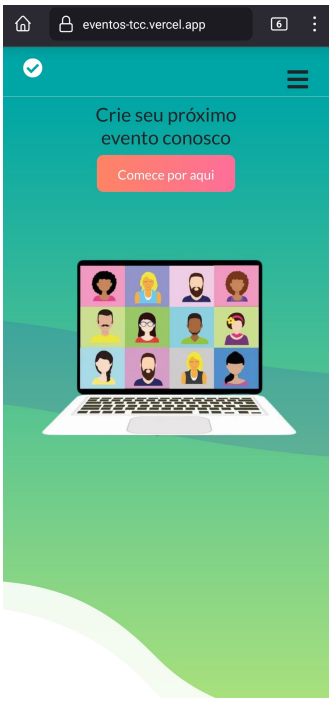
Figura 18 – Página de login



Fonte: Próprio autor

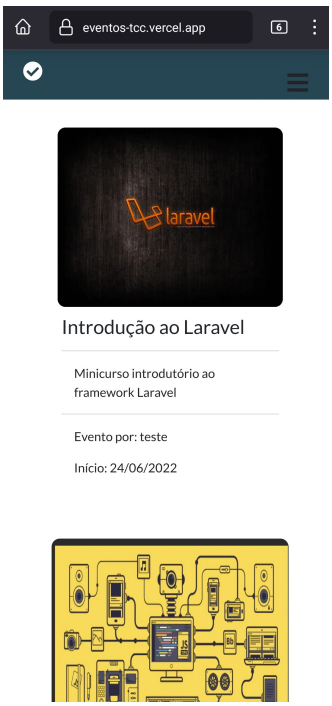
Utilizando a página inicial como exemplo, é possível ver a responsividade adaptada a ela por meio das figuras 19 e 20, que se adaptam a um smartphone. A diferença entre o tamanho de tela de um desktop para um smartphone fez com que a *navbar* seja oculta e as imagens dos eventos sejam exibidas apenas uma por linha, dimensionando a imagem para isto.

Figura 19 – Página inicial *mobile*



Fonte: Próprio autor

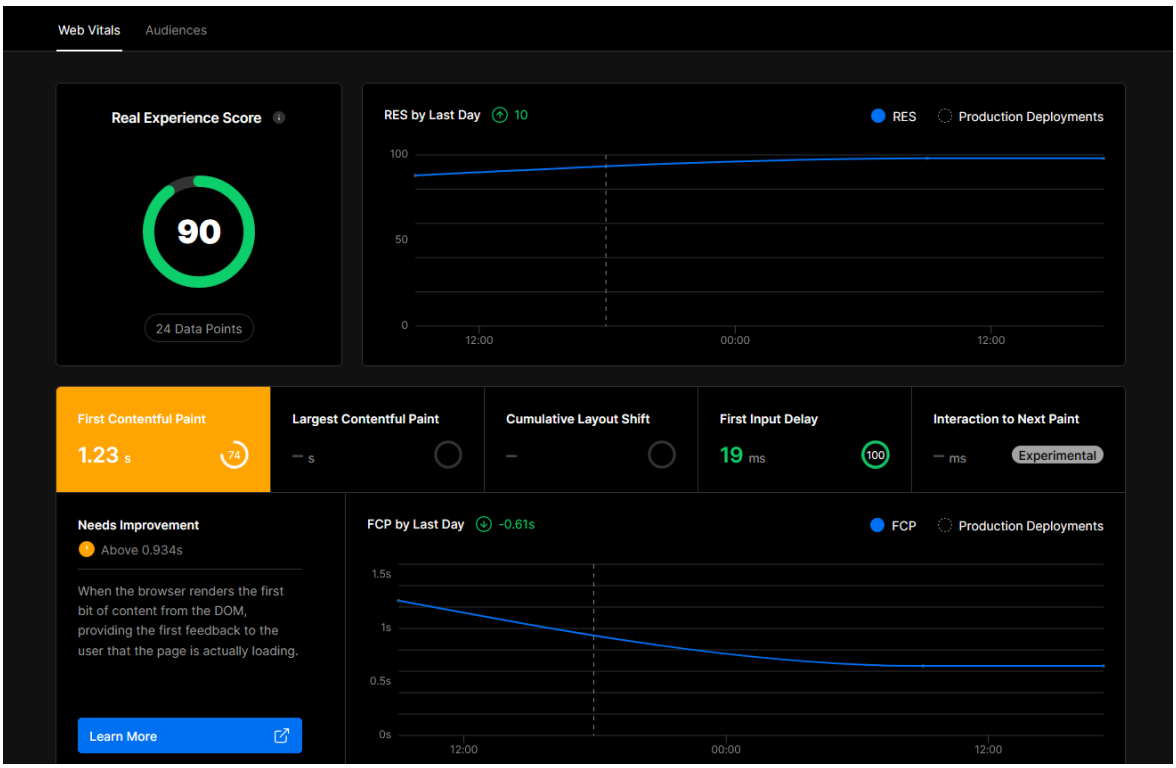
Figura 20 – Continuação da página inicial *mobile*



Fonte: Próprio autor

A fluidez, conforme as estatísticas capturadas pela Vercel, foi obtida por meio de carregamento rápido das páginas juntamente com a otimização de imagens fornecidas pelo framework Next.js. A imagem 21 demonstram os dados capturados e suas estatísticas disponíveis no plano gratuito, com o maior problema sendo a primeira impressão dos dados na tela(*First Contentful Paint*), que demonstra o primeiro carregamento dos dados no cliente. Este resultado é esperado, devido ao *cache* do framework, armazenado em seu servidor após o primeiro uso, o que decai ao longo do tempo, devido a este fator.

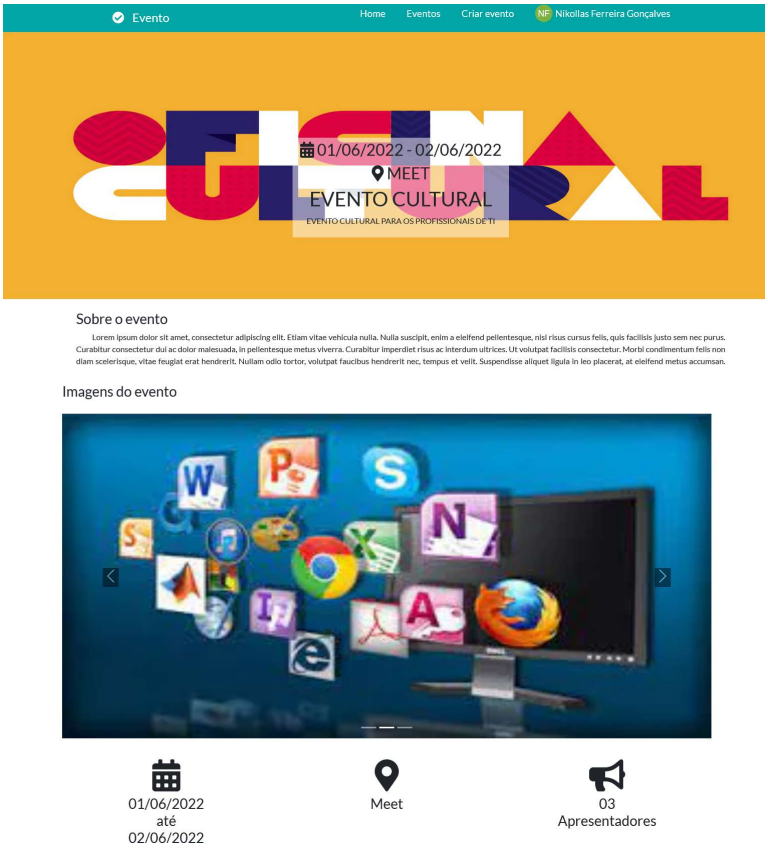
Figura 21 – Estatísticas Vercel



Fonte: Próprio autor

Foi criada uma exibição padrão para cada evento, seguindo o modelo de mockup. Com isto, a aplicação segue um modelo padrão para cada evento, modificando apenas as informações com as imagens contidas em cada um. Na figura 22, é possível ver o detalhamento do evento em si e, com a figura 23 (que estão na mesma página), tem-se completamente os dados para a participação do usuário, com a descrição completa de quem e o que será apresentado no evento.

Figura 22 – Página de evento



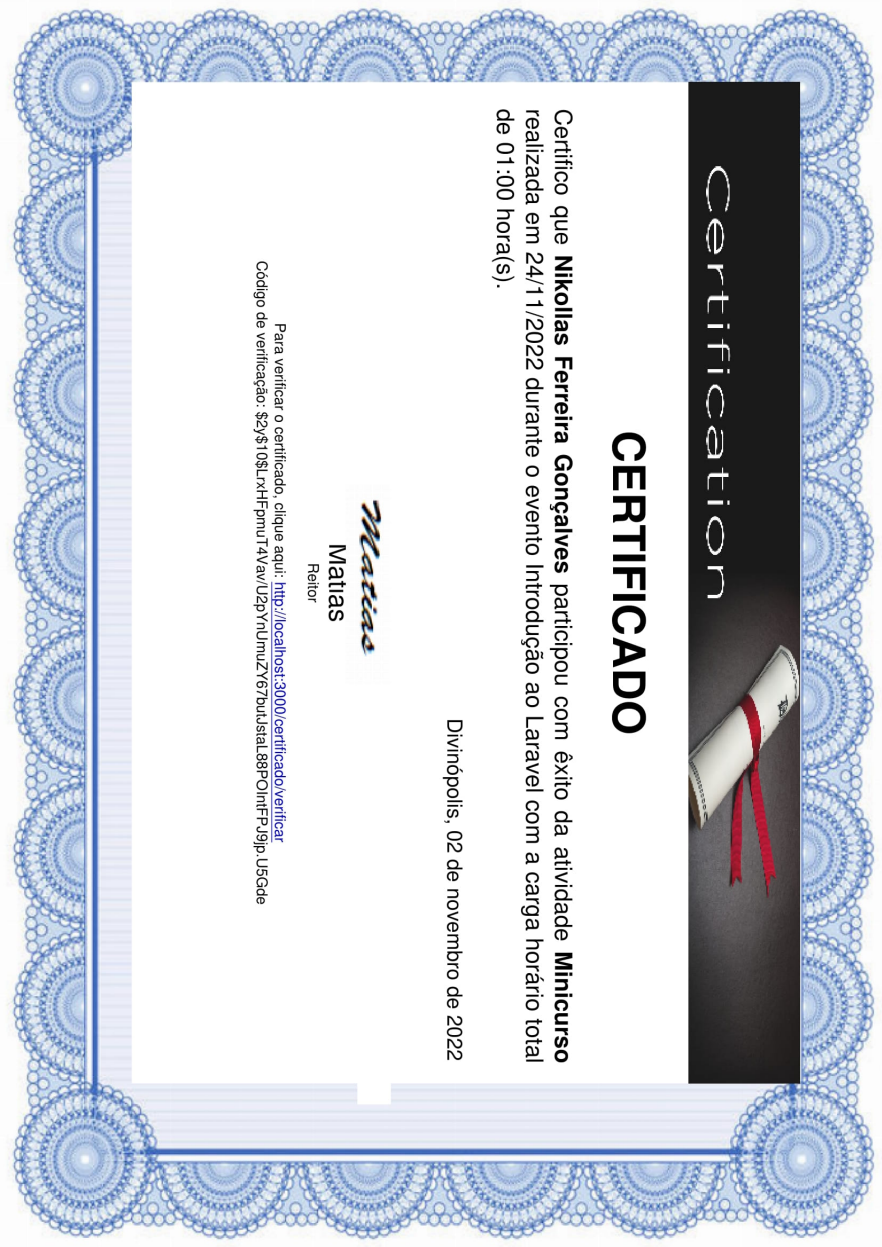
Fonte: Próprio autor

Figura 23 – Atividades

Fonte: Próprio autor

O modelo de certificado criado e que pode ser enviado em formato de PDF para o e-mail do participante, incluindo neste caso o apresentador também, também é gerido pela aplicação. Então, foi necessário criar um validador e enviar juntamente ao PDF o código para validação, como demonstrado na figura 24.

Figura 24 – Exemplo de certificado



Fonte: Próprio autor

A criação do *backend* que foi realizada apresenta um token JWT para autenticação e validação dos dados do usuário, além de realizar toda a troca de informações por meio de sua API REST e envio de e-mails dos certificados. Neste, é proporcionado um ambiente e troca de informações seguros por meio das requisições feitas pelo Next.js. Além disso,

toda a estruturação inicial planejada foi seguida, com exceções de tabelas auxiliares criadas pelo próprio framework Laravel para facilitar gravações de suas estatísticas e informações essenciais ele, ou seja, toda a cardinalidade das tabelas, assim como o projeto das requisições foi seguido conforme planejado anteriormente.

Por se tratar de um framework bem estruturado e organizado, com o Laravel, foi possível criar um padrão para o projeto que apresente funções bem determinadas e cumprindo o requisito de estarem em seu local, como os Controllers tendo as funções de comunicação com o banco de dados, assim como o retorno da requisição por meio de uma resposta em JSON, e as rotas chamando cada uma dessas funções para serem executadas. Portanto, isto facilitou o desenvolvimento, criando-se um padrão pré-determinado e podendo ser expandido ainda mais para projetos em maiores escalas.

O protótipo da aplicação apresenta uma estrutura bem subdividida, contendo o *frontend* e o *backend* estruturados pelos seus frameworks, podendo ser escalável.

A geração dinâmica dos certificados é uma característica relevante da aplicação que padroniza em uma instituição como podem ser seus documentos. Os modelos podem ser replicados ou diferentes para cada evento, podendo tornar padrão ou exclusivo para cada uso. A API permite a possibilidade de a criação de um aplicativo para *smartphones* sem a necessidade de refazer ou reestruturar o *backend*, devido à implementação de tecnologias como o JSON e ser acessível via protocolo HTTP. Os dados na API são protegidos por meio da autenticação fornecida pelo Laravel, ou seja, os dados da aplicação só serão repassados para quem estiver autenticado e com permissão de acesso a determinadas informações por meio do JWT daquele usuário, que tem prazo de validade.

Neste capítulo, foram apresentados os resultados do protótipo desenvolvido. A aplicação é capaz do gerenciamento de eventos, criação e edição dos usuários, assim como geração e validação dos certificados. Pode-se agregar diversas outras funcionalidades a este projeto, dado os frameworks e linguagens utilizados, sendo um protótipo que pode ser viabilizado e implementado em diversas instituições para seu uso no gerenciamento destas tarefas. As funcionalidades estão disponíveis na seguinte URL: <https://eventos-tcc.vercel.app/>, o repositório contendo o *frontend* está disponível em <https://github.com/NikFG/evento_front> e o repositório do *backend* está disponível em: <https://github.com/NikFG/eventos_api>.

6 Conclusão

Neste trabalho, foi desenvolvido e apresentado um protótipo de uma aplicação gerenciadora de eventos acadêmicos gratuitos online, que facilita a criação e a visualização para os cinco perfis de usuários.

O protótipo apresentado é capaz do gerenciamento de usuários, instituições e eventos, assim como a organização por três funções distintas de usuários. Somando essas características, há uma interface intuitiva que visa facilitar todos os processos para seus usuários, sejam apresentadores, visitantes, participantes ou gerenciadores dos eventos (supervisores e associados). Há características similares aos sistemas apresentados na introdução, unindo características relevantes de cada um, e transformando-as num protótipo capaz de gerenciar os eventos cadastrados.

A implantação deste sistema, inicialmente, pode ser feita em qualquer âmbito acadêmico, visto que abrange os mais variados tipos de eventos acadêmicos de maneira remota, podendo substituir uma aplicação utilizada atualmente em uma determinada instituição. Ou seja, a sua geração de certificados, a qual possibilita até mesmo a verificação e envio por e-mail, facilita todo o gerenciamento ao selecionar os participantes e apresentadores que estiveram de fato no evento.

O método de adquirir ingressos poderia diferir, sendo mais similar a um *ecommerce*, contendo carrinho com a possibilidade de selecionar diversas atividades de diversos eventos. Há também funcionalidades que poderiam ser agregadas ao sistema, tirando-o apenas do ramo acadêmico, como impressão de ingressos via qrcode.

Este trabalho poderá contribuir com as instituições que realizam recorrentemente tais eventos acadêmicos. Devido aos *frameworks* e tecnologias utilizados neste projeto, a aplicação funciona de maneira responsiva, podendo ser utilizada, tanto em computadores de uso pessoal, como em dispositivos móveis, sem que haja maiores problemas, como a perda de funcionalidades e quebras no layout.

Para trabalhos futuros, é necessário que tenha integração com APIs de *gateway* de pagamentos para fornecer à aplicação funcionalidades de ingressos pagos, seja via cartão de crédito/débito ou pix, visto que nem sempre um evento pode ser algo gratuito. A aplicação poderá fazer uma reserva mediante a confirmação do pagamento, gerando um identificador único ao ingresso e podendo ser apresentado em formato de *qrcode* ou o texto do código em si, que deve ser validado ao comparecer às atividades compradas.

A solução adotada pode apresentar dificuldades ao escalar para instituições de vários campi por não atender ao requisito hierárquico. Como trabalho futuro, um painel

administrativo poderá ser criado como meio de gerenciar por uma instituição centralizadora da informação, como exemplo: a matriz e filiais de uma instituição acadêmica, de modo que tenham até mais níveis de usuário para cada instituição ou de forma geral.

Alguns eventos hoje não são atualizados sem a necessidade de atualizar a página, o que pode causar números divergentes da realidade de participantes, por exemplo. Uma solução para trabalhos futuros pode ser a implementação por meio de *websockets* que geram atualizações por meio de eventos no banco de dados e atualizam a página apenas no dado escolhido, neste exemplo o número de participantes.

Referências

BEZERRA, E. *Princípios de Análise e Projeto de Sistemas com UML*. Rio de Janeiro: Campus; 2ª Edição, 2007. ISBN 9788535216967. Citado 3 vezes nas páginas 16, 17 e 18.

BRASIL, R. F. D. *Eventos: Presencial x Online*. 2021. Disponível em: <http://www.feirasdobrasil.com.br/revista.asp?area=assuntos&codigo=856>. Acesso em: 5 mai 2021. Citado na página 14.

CHERNY, B. *Programming TypeScript*. Sebastopol: O'Reilly Media, Inc; 1. ed., 2019. ISBN 9781492037651. Citado na página 23.

COULOURIS, G. et al. *Sistemas Distribuídos: conceitos e projetos*. Porto Alegre: Bookman, 2013. ISBN 9788582600535. Citado 2 vezes nas páginas 31 e 43.

DIAS, R. *O Processo Unificado*. 2019. Website. Disponível em: <https://medium.com/contexto-delimitado/o-processo-unificado-d102b1fc9d00>. Acesso em: 10 ago 2022. Citado na página 18.

DUCKETT, J. *HTML & CSS Design and Build Websites*. Indianapolis: John Wiley & Sons, Inc., 2011. ISBN 9781118008188. Citado na página 22.

FIREBASE. *Documentação*. 2022. Firebase. Disponível em: <https://firebase.google.com/docs?hl=pt>. Acesso em: 4 fev 2022. Citado na página 29.

FLANAGAN, D. *JavaScript: O guia definitivo*. [S.l.]: Bookman; 6ª edição, 2013. ISBN 9788565837194. Citado 2 vezes nas páginas 22 e 23.

HEROKU. *Documentation - Getting Started*. 2022. Heroku. Disponível em: <https://devcenter.heroku.com/categories/reference>. Acesso em: 2 fev 2022. Citado na página 30.

HEUSER, C. A. *Projeto de Banco de Dados*. [S.l.]: Bookman; 6ª edição, 2009. ISBN 9788577804528. Citado 3 vezes nas páginas 21, 25 e 28.

INTERACTION DESIGN FOUNDATION. *User Experience (UX) Design*: What is user experience (ux) design? S.D. Website. Disponível em: <https://www.interaction-design.org/literature/topics/ux-design>. Acesso em: 12 ago 2022. Citado na página 20.

IPSENSE. *Entenda a importância do versionamento de software*. 2020. Website. Disponível em: <https://www.ipsense.com.br/blog/entenda-a-importancia-do-versionamento-de-software/>. Acesso em: 6 jun 2022. Citado na página 21.

KONSHIN, K. *Next.js Quick Start Guide: Server-side rendering done right*. Birmingham: Packt Publishing, 2018. ISBN 9781788995849.57388. Citado na página 26.

LOCKHART, J. *Modern PHP*. Sebastopol: O'Reilly Media, Inc, 2015. ISBN 1491905182. Citado na página 23.

MDN contributors. *Design Responsivo*. 2022. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/CSS/CSS_layout/Responsive_Design>. Acesso em: 5 mai 2022. Citado na página 19.

MDN contributors. *Uma visão geral do HTTP*. 2022. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: 5 jun 2022. Citado na página 20.

MICROSOFT. *TypeScript Documentation*. 2022. Website. Disponível em: <https://www.typescriptlang.org/docs/>. Acesso em: 8 jun 2022. Citado na página 23.

NOLLE, T. *What is reactive programming?* 2021. Website. Disponível em: <https://www.techtarget.com/searchapparchitecture/definition/reactive-programming>. Acesso em: 10 ago 2022. Citado na página 19.

NORMAN, D.; NIELSEN, J. *The Definition of User Experience (UX)*. S.D. Website. Disponível em: <https://www.nngroup.com/articles/definition-user-experience/>. Acesso em: 8 set 2022. Citado na página 20.

PIRES, J. *O que é API? REST e RESTful? Conheça as definições e diferenças!* 2017. Website. Disponível em: <https://becode.com.br/o-que-e-api-rest-e-restful/>. Acesso em: 6 jun 2022. Citado na página 20.

PRESSMAN, R. *Engenharia de software*. New York: AMGH Editoras; 7ª Edição, 2011. ISBN 9788580550443. Citado 2 vezes nas páginas 16 e 17.

ROBBINS, J. N. *LEARNING WEB DESIGN*. Sebastopol: O'Reilly Media, Inc., 2018. ISBN 978-1-491-96020-2. Citado na página 23.

SACRAMENTO, G. *Versionamento de código e de software: entenda cada processo*. 2021. Website. Disponível em: <https://blog.somostera.com/data-science/versionamento>. Acesso em: 7 jun 2022. Citado na página 21.

SANTACROCE, F. *Git Essentials*. Birmingham: Packt Publishing, 2015. ISBN 1785287907. Citado 3 vezes nas páginas 25, 28 e 29.

SEBRAE. *Entenda o impacto da pandemia no setor de eventos: Pesquisa realizada pelo Sebrae mostra como a crise tem afetado o segmento. Saiba mais sobre o reflexo da pandemia nas atividades econômicas*. 2020. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/artigos/entenda-o-impacto-da-pandemia-no-setor-de-eventos,424ba538c1be1710VgnVCM1000004c00210aRCRD>. Acesso em: 27 jun 2021. Citado na página 14.

SEBRAE. *TUDO QUE VOCÊ PRECISA SABER SOBRE MÉTRICAS DE UX*. 2021. Website. Disponível em: <https://bibliotecas.sebrae.com.br/chronus/ARQUIVOS_CHRONUS/bds/bds.nsf/6f47c91e556bb2f56e3ab1eb06a26d3a/\protect\T1\textdollarFile/30717.pdf>. Acesso em: 20 set 2022. Citado na página 20.

SOMMERVILE, I. *Engenharia de software*. São Paulo: Pearson Universidades; 9ª Edição, 2011. ISBN 9788579361081. Citado 4 vezes nas páginas 16, 17, 31 e 34.

SOUTO, M. *O que é front-end e back-end?* 2019. Website. Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>. Acesso em: 1 jul 2022. Citado 2 vezes nas páginas 19 e 20.

SPURLOCK, J. *Bootstrap: Responsive Web Development*. Sebastopol: O'Reilly Media, Inc, 2013. ISBN 9781449343910. Citado na página 26.

Statistics and Data. *Most Popular Backend Frameworks – 2012/2021 – New Update*. 2021. Website. Disponível em: <https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2021/>. Acesso em: 4 fev 2022. Citado na página 24.

STAUFFER, M. *Laravel: a Framework for Building Modern PHP Apps*. Sebastopol: O'Reilly Media, Inc; 2nd. ed., 2019. ISBN 9781492041214. Citado na página 24.

TOTVS. *O que é back-end e qual seu papel na programação?* 2020. Website. Disponível em: <https://www.totvs.com/blog/developers/back-end/>. Acesso em: 10 ago 2022. Citado na página 20.

VERCEL. *Documentation - Getting Started*. 2022. Vercelnextjs. Disponível em: <https://nextjs.org/docs>. Acesso em: 1 fev 2022. Citado 3 vezes nas páginas 26, 27 e 29.

WAZLAWICK, R. S. 2010. Website. Disponível em: <https://professorleomir.files.wordpress.com/2012/09/material-de-apoio-casos-de-uso-expandidos1.pdf>. Acesso em: 8 set 2022. Citado na página 38.

ZAMMETTI, F. *Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker*. New York: Apress, 2020. ISBN 9781484257388. Citado na página 26.

Apêndices

Anexos