FutManager: Sistema *web* para gerenciamento de uma equipe de futebol

Rayssa T. Souza, André C. da Silva

[1]Grupo de Pesquisa Mobilidade e Novas Tecnologias de Interação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Ciência e Tecnologia de São Paulo (IFSP)
Avenida Thereza Ana Cecon Breda, N.º 1896,
Vila São Pedro – 13183-250 – Hortolândia – SP – Brasil

rayssa.teixeira@aluno.ifsp.edu.br, andre.constantino@ifsp.edu.br

Abstract. *It is well known that soccer is one of the most widely practiced sports in Brazil and, because of this, there are many schools that teach this sport in depth, one of which is the Futpaz Social Project. In a conversation with the project's team, of which I am a member, I noticed a need to computerize the team's processes, which are done manually. The aim of this work was to develop a web system to computerize the registration of students in the project and their guardians, as well as to control attendance and warnings during classes. For this development, an incremental approach was adopted with six increments resulting in a first executable version of the software.*

Resumo. *Sabe-se que o futebol é um dos esportes mais praticados no Brasil e, devido a isso, existem muitas escolas de ensino aprofundado neste esporte, sendo uma delas o Projeto Social Futpaz. Em conversa com a equipe deste projeto, a qual faço parte, foi possível notar necessidade de informatização de processos da equipe, já que são feitos manualmente. Este trabalho surge com o intuito de desenvolver um sistema web para informatizar o cadastramento de alunos do projeto e seus responsáveis, um controle de presença e advertências durante as aulas ministradas. Para esse desenvolvimento, adotou-se uma abordagem incremental com seis incrementos resultando em uma primeira versão executável do software.*

1. Introdução

Não é novidade que o futebol é um esporte muito apreciado e praticado no Brasil, segundo a revista Superinteressante, 2018, são mais de 30 milhões de brasileiros praticantes desta modalidade esportiva[Fujita 2018]. Devido a isso, existem muitas academias e escolas de treinamento de futebol, desde categorias infantis até as equipes profissionais, e uma dessas escolas é o Projeto Social FutPaz.

O Projeto Social FutPaz [Futpaz 2022], fica situado em um bairro carente da cidade de Piracicaba-SP, e é uma escola especializada no ensino da prática de futebol para crianças entre as idades de 7 a 15 anos (sub 9, sub 11, sub 13 e sub 15). Atualmente, esta escola atende em torno de 80 crianças e pré-adolescentes que buscam um aperfeiçoamento dentro do esporte, ofertando aulas e treinamentos, além de oferecer outros benefícios ao aluno, por mérito do jogador, e a sua família, por meio de sorteios.

Em uma conversa com a equipe responsável por este Projeto Social, a qual sou uma das integrantes, foi possível notar uma necessidade de informatização de alguns processos realizados pela comissão, pois, até o momento, realiza-se manualmente todas as anotações referentes aos alunos e às suas documentações, o que torna o trabalho de armazenamento e organização das informações referentes aos atletas, responsáveis, equipe, campeonatos, amistosos, entre outros, trabalhoso.

A partir desta problemática identificada pela equipe, surge a ideia de desenvolver um sistema *web* responsivo, pois dentre tantas formas de desenvolvimento, esta é uma área ampla e muito utilizada no mercado. Segundo o IBGE [Nery and Britto 2022], no ano de 2021, 90% das pessoas possuem acesso à internet em seus domicílios no Brasil, sendo que 99,5% desse acesso é realizado via celular, tornado assim o desenvolvimento *web* uma opção mais democrática para a criação deste sistema.

Dessa forma, será possível informatizar os processos, atualmente feitos manualmente, auxiliando o trabalho da comissão, tornando digitais o acesso e cadastro das informações, pois poderão ser feitos de qualquer dispositivo que tenha acesso à internet, sendo ele um *smartphone*, *notebook*, *tablet*, etc.

Portanto, este projeto terá como objetivo oferecer um sistema de cadastramento de alunos e seus responsáveis e um controle de presença e advertências durante as aulas ministradas no projeto.

Neste artigo serão abordados diversos tópicos relacionados ao desenvolvimento do sistema FutManager, retratando desde o entendimento do problema, até o desenrolar de uma solução para os impedimentos evidenciados. Como primeira seção é abordado o referencial teórico, onde é possível entender os desdobramentos de alguns conceitos relacionados ao trabalho (Seção 2), em sequência é demonstrado alguns aplicativos semelhantes e correlacionados ao que o FutManager propõe (Seção 3), qual será a metodologia implementada (Seção 4), quais serão os passos de execução do desenvolvimento do protótipo de sistema (Seção 5) e por fim, o artigo se encerra com as conclusões (Seção 6).

2. Referencial teórico

Nesta seção do artigo serão apresentados os conceitos de requisitos funcionais e não funcionais, o conceito de responsividade, arquitetura de *software*, como também os padrões arquiteturais: MVC, Cliente-Servidor e Arquitetura em Camadas. Para o desenvolvimento da aplicação, também será abordado a metodologia incremental.

2.1. Requisitos funcionais e não funcionais

Os requisitos funcionais são os serviços que o sistema deve oferecer e assim mostrar como o sistema deve funcionar/reagir a determinadas entradas, e como ele irá se comportar em algumas situações. Os requisitos funcionais dependem diretamente do tipo de *software* que será desenvolvido [Sommerville 2019].

Já os requisitos não funcionais são as restrições que existem sobre os serviços e as funções do sistema. Esses requisitos normalmente são aplicados a todo o sistema, mas pode ocorrer de funcionar em somente algumas partes do sistema ou em algumas funções. As restrições mais comuns estão relacionadas a disponibilidade (contingência em casos de indisponibilidade), escalabilidade (capacidade de lidar com um aumento de carga), segurança (proteção do sistema), responsividade (capacidade de adaptação de uma aplicação a diferentes tipos de dispositivos), entre outros [Sommerville 2019].

2.2. Responsividade

Dentro do contexto da criação de um sistema *web*, atualmente existe a necessidade do acesso ao site por diferentes dispositivos usados pelos usuários. Independente de qual dispositivo o usuário estará utilizando, este deve ter uma boa experiência de uso com a aplicação.

Definida originalmente por Ethan Marcotte [Marcotte 2010], a responsividade consiste em um *web design* que busque atender as necessidades dos usuários e seus dispositivos, gerando assim uma experiência de visualização ideal, tornando os projetos não somente flexíveis, mas também adaptáveis para qualquer mídia ou dispositivo que o reproduza.

Basicamente o design responsivo busca tornar o *layout* adaptável para as capacidades e dimensões dos dispositivos, sendo assim, o sistema torna-se capaz de mudar sua base de tamanho, além de tornar seu conteúdo mais flexível.

2.3. Arquitetura de software

A arquitetura de *software* é o conjunto de estruturas que são vitais para a criação de um sistema, essas estruturas correspondem por elementos de *software*, que podem ser serviços, componentes, módulos, entre outros [dos Santos 2020]. Ela sempre está presente em todos os *softwares*, e em sua elaboração, leva-se em consideração os requisitos funcionais e não funcionais da aplicação em questão. Normalmente, formula-se a arquitetura de um *software* levando em consideração alguns padrões de arquitetura como o MVC, a arquitetura cliente-servidor ou a arquitetura em camadas.

O padrão de arquitetura MVC (Figura 1) é uma arquitetura que define três componentes: *Model*, *View*, *Controller*. Ela tem uma dinâmica simples, todas as requisições da aplicação são enviadas para o *Controller*, que consegue acessar a camada *Model* e essa processa a requisição, depois direciona para o *View*, onde é exibida a informação [Luciano and Avles 2011].

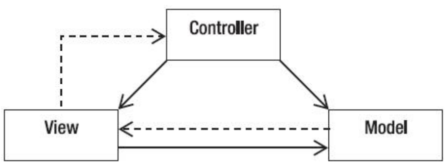

Figura 1. Arquitetura MVC [Higor 2013]

Já o padrão de arquitetura cliente-servidor, consiste em um sistema apresentado com um conjunto de serviços, e os serviços são fornecidos por um servidor. Os clientes são usuários desses serviços e acessam os servidores para usá-los [Sommerville 2019]. Nesta arquitetura, os clientes que utilizam os serviços, têm um contato com o *front-end* do mesmo, enquanto o *back-end* do serviço está nos servidores onde eles estão localizados.

Nesta Figura 2 é possível ver um exemplo da aplicação da arquitetura cliente-servidor, onde uma biblioteca de filmes oferece diversos serviços que estão hospedados

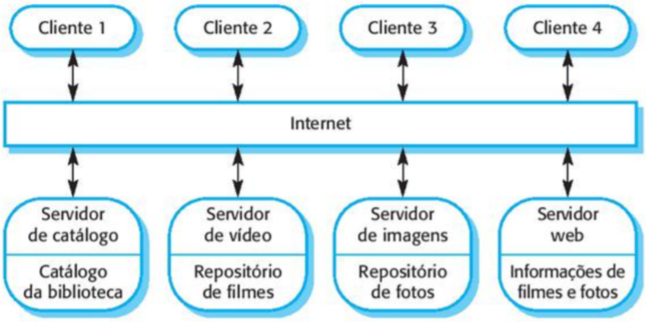

Figura 2. Exemplo de Arquitetura cliente-servidor[Sommerville 2019]

em diferentes servidores, para acessar esses serviços, o cliente faz uma requisição através da internet e os servidores recebem essas requisições e retornam suas informações.

Ainda dentro das arquiteturas de *softwares*, existe a arquitetura em camadas que segundo [Sommerville 2019], o sistema em camadas é organizado de forma que cada camada exerça uma funcionalidade e assim forneça serviços para a camada acima dela, portanto as camadas mais inferiores são aquelas que representam os serviços essenciais e mais utilizados por todo o sistema.

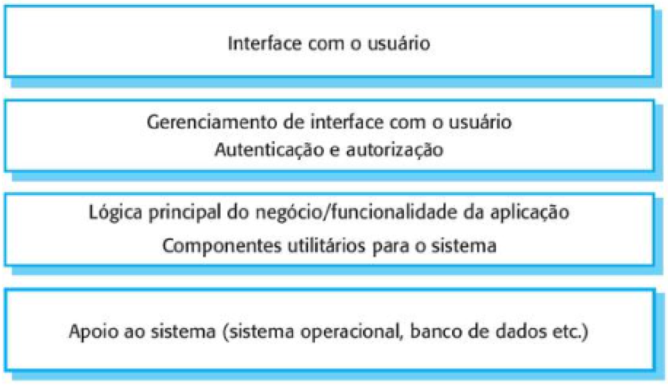

Figura 3. Exemplo genérico da Arquitetura em Camadas [Sommerville 2019]

Na Figura 3 podemos ver uma exemplificação dessa arquitetura, onde temos quatro camadas, sendo a principal camada o apoio ao sistema, que fornece seus serviços para a camada de lógica principal do negócio, que assim como a camada anterior, fornece seus serviços para o gerenciamento de interface, que provisiona serviços para a camada de interface com o usuário.

2.4. Modelo incremental

No desenvolvimento de *software* deve-se adotar um modelo de desenvolvimento. Há diversos modelos propostos na literatura, como o cascata, prototipação, espiral e incremental [Sommerville 2019]. O modelo selecionado para este trabalho foi o modelo incremental devido a suas características explicadas a seguir.

O modelo incremental junta elementos do modelo cascata e modelos mais iterativos. O modelo de processo incremental tem como objetivo apresentar um produto a cada

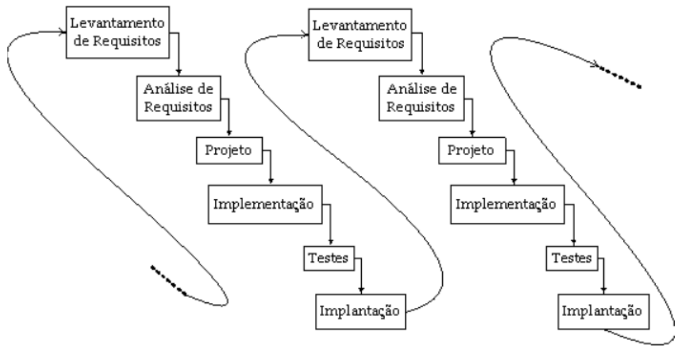

Figura 4. Modelo incremental [Lessa and Lessa Junior 2009]

incremento realizado. Conforme a Figura 4, em cada incremento existem as etapas: levantamento de requisitos, análise de requisitos, projeto, implementação, testes e implantação [Lessa and Lessa Junior 2009].

Nas etapas de levantamento e análise de requisitos são realizados os processos de pesquisa, entendimento e análise de um subconjunto de requisitos funcionais e não funcionais (Subseção 2.1). Essas etapas são importantes para entender as necessidades, problemas e objetivos dos usuários.

Já nas etapas de projeto e implementação, ocorre de fato o desenvolvimento do *software*/aplicativo relacionado aquele subconjunto de requisitos levantado previamente. Também existe a etapa de testes, que consiste em uma dedicação da equipe em realizar avaliações e testes no *software*, para validar se o incremento gerado está segundo os requisitos levantados. E por fim, encontra-se a implantação, que basicamente é a etapa de onde o incremento do *software*, agora pronto, pode ir para o seu ambiente de produção.

3. Trabalhos correlatos

Pesquisando mais sobre a temática, foi possível encontrar trabalhos correlatos que podem auxiliar a compreender o escopo do projeto e os seus requisitos. Sendo eles:

3.1. Copa Fácil

Criado em 2018, o aplicativo *web* e *mobile* Copa Fácil serve para gerenciamento de campeonatos de várias modalidades (Futsal, Futebol, Futebol 7, handebol, basquetebol, vôlei, entre outros). Este sistema contém muitas funcionalidades (Figura 5), desde criar um torneio, criar categorias para o torneio, configurar as equipes, adicionar atletas, criar *rankings* de habilidades, entre outras funções [Fácil 2018].

Tendo mais de um milhão de *downloads*, o aplicativo Copa Fácil utiliza-se muito bem do conceito de responsividade, pois fornece ao usuário final as opções de utilizar o mesmo sistema em diversos aparelhos tecnológicos.

Esse sistema disponibiliza ao usuário uma versão gratuita, que se limita ao cadastro de 100 jogadores por campeonato, ultrapassando essa quantidade, é necessário ativar um dos planos pagos da plataforma. Os valores dos planos podem variar conforme o campeonato, sendo proporcional ao seu tamanho, dessa forma campeonatos pequenos tem um valor menor e campeonatos maiores, um valor maior (Figura 5).

Figura 5. Telas do aplicativo Copa Fácil - Imagens disponibilizadas na Google Play Store [Store 2018]

Figura 6. Planos do Aplicativo Copa Fácil [Fácil 2018]

3.2. SportEasy

Desenvolvido por Albin Egasse e Nizar Melki, no ano de 2012, a SportEasy é uma empresa, que criou um aplicativo, com o mesmo nome, que serve como um gerenciador de uma equipe esportiva, desde o cadastro dos atletas da equipe, a gestão de campeonatos e jogos, comunicação interna, ou seja, entre os membros da equipe, estatísticas do seu time, entre outras funções [Egasse and Melki 2012].

Este sistema foi pensado para a *web* responsivo e *mobile* nativo (Figura 7), no intuito de auxiliar seus 2,5 milhões de usuários.

O aplicativo também disponibiliza diversos esportes para se criar uma equipe, sendo alguns deles: futebol, rugby, basquetebol, voleibol, atletismo, entre outras atividades esportivas.

A aplicação possui dois planos para serem utilizados, uma versão gratuita, onde suas funcionalidades são restritas a: Organização, Comunicação, Espírito de equipe e

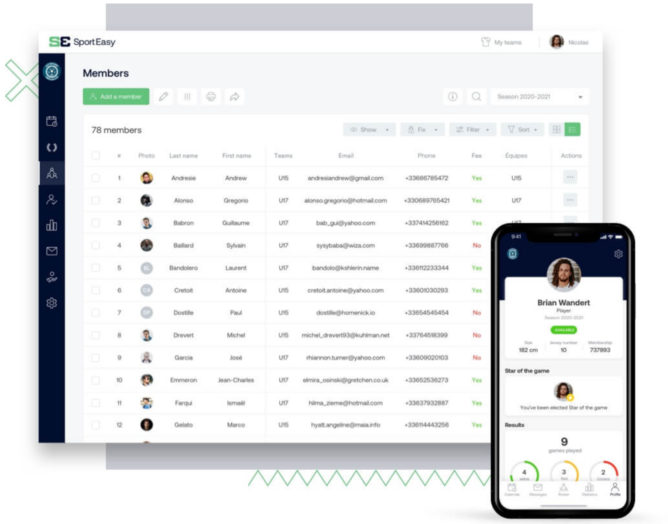

Figura 7. Telas do aplicativo SportEasy para *web* e *mobile* [Egasse and Melki 2012]

Contribuições e Coletas. E uma versão paga, que disponibiliza além das funcionalidades gratuitas: atribuição de tarefas, Estatísticas e Sínteses das presenças (Figura 8).

Características	BASIC	PREMIUM
	Grátis (com publicidade)	9,99€/mês por equipa • 1 mês de teste gratuito • 69€/ano se pagas anualmente (equivalente a 5,75€/mês)
Organização	✓	✓
Comunicação	✓	✓
Espírito de equipa	✓	✓
Contribuições e coletas	✓	✓
Atribuição de tarefas		✓
Nenhuma publicidade clássica		✓
Estatísticas		✓
Síntese das presenças		✓

Figura 8. Planos disponíveis para o SportEasy [Egasse and Melki 2012]

3.3. Ritmo do Esporte

Dentro da mesma temática, ainda temos o Ritmo do Esporte, um aplicativo *mobile* (Figura 9) desenvolvido por Luciano Marin e Nathan Reuter, no ano de 2016, utilizando-se da linguagem de programação JavaScript através do Vue.js e Node.js. Esse aplicativo tem o mesmo intuito do anterior, gerenciar uma equipe, porém seu foco está mais na parte de gestão financeira.

Utilizado por 140 escolas, totalizando aproximadamente 20 mil alunos, este aplicativo apresenta a funcionalidade de acompanhar o que acontece com a sua equipe, gerenciar a parte financeira do clube, controle dos atletas, opção para convocar os jogadores para jogos, criação de avisos e comunicações para as equipes, controle de presenças, entre outros. O *software* Ritmo do Esporte, não é disponível gratuitamente, sendo necessário adquirir um plano para a sua utilização, sendo que o mesmo custa e torno de 200 reais. [Marin and Reuter 2016].

(a) **(b)**

Figura 9. Tela de login e tela dos atletas registrados da aplicação Ritmo do Esporte

Após entender como cada trabalho correlato funciona, temos representado na Tabela 1 um levantamento comparativo das funcionalidades apresentadas nos aplicativos citados anteriormente e as funcionalidades esperadas para a aplicação FutManager.

Tabela 1. Comparação entre os trabalhos correlatos

Funcionalidades	FutManager	Copa Fácil	Sport Easy	Ritmo do Esporte
Gestão de uma equipe de futebol	✓	✗	✓	✓
Gerenciamento de atletas	✓	✓	✓	✓
Gerenciamento de jogos e campeonatos	✓	✓	✓	✓
Controle financeiro da equipe	✗	✗	✗	✓
Comunicação e mensagens	✗	✓	✓	✓
Controle de presença e advertências	✓	✗	✗	✓
Estatísticas da equipe e jogadores	✗	✓	✓	✓
Controle de familiares e responsáveis	✓	✗	✗	✗

Como evidenciado na Tabela 1, existem algumas diferenças entre o aplicativo FutManager e os demais trabalhos correlatos. É possível verificar que todos os sistemas possuem a funcionalidade de gerenciamento de jogos e campeonatos e gestão de atletas. Assim tanto o FutManager, o SportEasy e o Ritmo do Esporte dividem a funcionalidade de gerenciamento de uma equipe de futebol.

Mas, nota-se que o FutManager não possui as funções: controle financeiro (exclusivo do Ritmo do Esporte), comunicação e mensagens e estatísticas da equipe e jogadores (comum aos demais). Porém, também pode-se perceber que o aplicativo a ser desenvolvido neste artigo se destaca, ao fornecer algumas opções diferente dos demais: o controle de presenças e advertências e o controle de familiares e responsáveis.

4. Metodologia

A metodologia utilizada foi o modelo incremental, que consiste na entrega de pequenos incrementos, ou pequenas partes do *software* por vez. Para este desenvolvimento, esta metodologia foi implementada na divisão de incrementos, sendo cada um deles o desenvolvimento de uma ou mais funcionalidades do sistema, dessa forma criando o *front-end*, ou seja, todas as telas do sistema, que também pode ser chamado de camada *view* da arquitetura MVC e também o desenvolvimento do *back-end*, ou seja, a conexão das telas com o banco de dados, também conhecida como os *controllers* e *models* da arquitetura MVC, citada na subseção 2.3 do Referencial teórico.

Dessa forma, o primeiro passo foi fazer um levantamento de requisitos funcionais e não funcionais do *software*, mediante uma reunião com a comissão técnica do Projeto FutPaz, e a partir deles, criar diagramas de caso de uso para registro das funcionalidades do sistema.

Na sequência, com informações coletadas no levantamento de requisitos, foi elaborada a modelagem do banco de dados, com as entidades, atributos e relacionamentos, gerando o diagrama de modelagem de dados lógico. O banco de dados é uma coleção de dados que é organizada e armazenada em um sistema de computador. Normalmente, um banco de dados é controlado por um SGBD (Sistema de Gerenciamento de Banco de Dados) [Oracle 2022], sendo que nesse projeto, o SGBD escolhido é o MySQL.

Partindo para o desenvolvimento dos códigos do sistema, o *software* foi produzido por meio de *frameworks*, que consistem em ferramentas que auxiliam no desenvolvimento do projeto, deixando as tarefas repetitivas para as ferramentas, além de automatizar parte do trabalho do desenvolvedor [Murilo and Bittencourt 2021]. Neste aplicativo foram utilizados os *frameworks*: *React* e Laravel, que utilizam as linguagens de programação *JavaScript* e PHP, respectivamente.

Desenvolvido pela Meta, o *framework React* é uma ferramenta que facilita o desenvolvimento do *front-end web* de uma aplicação, ele segue a ideia de criar um sistema mediante componentes, sendo que um componente é uma parte da interface do usuário (UI) que é totalmente customizável, e que possui suas próprias lógicas e aparência, auxiliando também no desenvolvimento de um sistema responsivo [Meta 2023].

Já o *framework* Laravel foi desenvolvido por uma equipe de nove desenvolvedores de todo o mundo, eles criaram essa ferramenta para ser um facilitador de construções de aplicações *web*, sendo possível desenvolver um sistema de forma *full-stack*, além das facilidades de ter uma camada de abstração de banco de dados, testes unitários, trabalhos agendados, entre outras [Laravel 2023].

Para o desenrolar deste projeto, foi utilizado o *framework React* para a codificação do *front-end* e a ferramenta Laravel será utilizada somente para o desenvolvimento do *back-end*. Também foi utilizado como auxílio para o *front-end* o *framework* de CSS chamado Tailwind, ele é um facilitador da criação de estilos para os componentes, consequentemente, ele simplifica a codificação para deixar o sistema responsivo. Outros recursos usufruídos no desenvolvimento foram o protocolo de segurança OAuth2 e o JWT (JSON Web Tokens), eles foram implementados no *back-end* em Laravel, através de uma biblioteca chamada *Passport*, esses recursos servem para lidar com a emissão de *tokens* de acesso, renovação de *tokens* e revogação de *tokens*.

Essa maneira de desenvolvimento foi utilizada visando projetar o sistema para *web*, que atende-se as arquiteturas de *softwares* citadas na subseção 2.3 do Referencial teórico e que cumpra com os ideais de responsividade (subseção 2.2 do Referencial teórico)

5. Desenvolvimento do trabalho

O primeiro passo para o desenvolvimento do trabalho foi o levantamento de requisitos funcionais e não funcionais, e para isso foi utilizado a técnica de reunião com o cliente, ou seja, com a comissão técnica e administrativa do Projeto Social Futpaz. Nesta reunião estavam presentes o presidente do Projeto, a primeira secretaria e a segunda secretaria. Foi falado sobre as funcionalidades que serão desenvolvidas durante o trabalho, e cada uma foi detalhada e discutida para melhor entendimento do que será produzido.

Também discutimos a possibilidade de o *software* estar disponível para celulares e computadores, e definimos que o aplicativo deve ser desenvolvido para *web*, porém responsivo, podendo ser utilizado por diversos dispositivos, sendo esse um dos requisitos não funcionais. Outro requisito não funcional discutido foi facilidade de utilização, visto que no projeto existem diversos tipos de pessoas, dessa forma o *software* seria criado pensando no conhecimento tecnológico desse público-alvo.

Após o término da reunião foi desenvolvido um diagrama de caso de uso (Figura 10), para melhor visualização dos requisitos. Neste diagrama estão especificados os requisitos funcionais do aplicativo e quais são os atores do sistema (administrador, equipe, responsável e atleta).

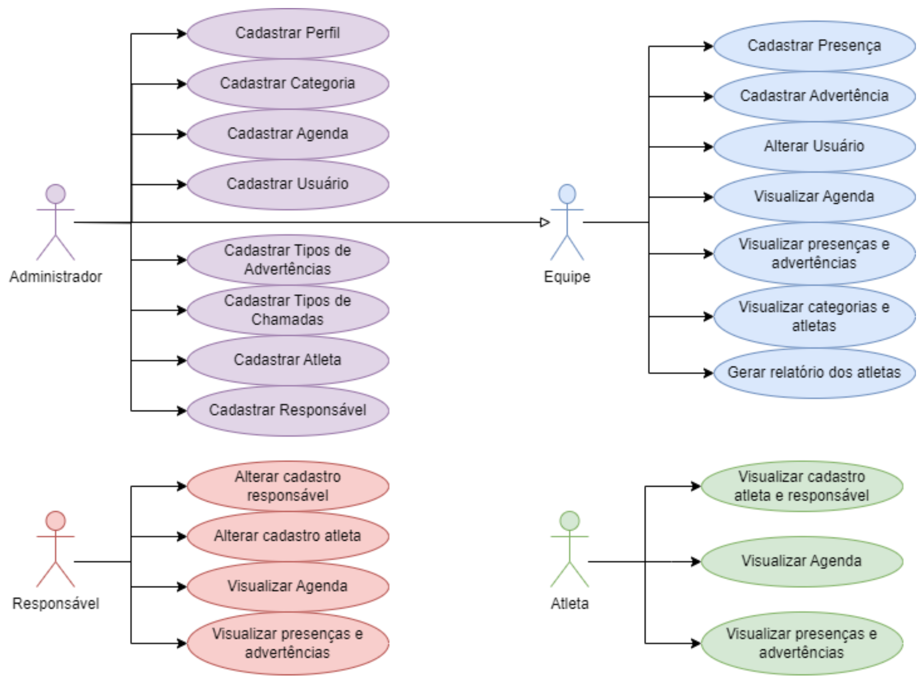

Figura 10. Diagrama de Caso de Uso

No diagrama, fica evidenciado cada funcionalidade que está disponível para os atores do sistema. Os atores administrador e equipe, serão utilizados pela comissão técnica do Projeto Futpaz, eles possuem uma maior gama de funcionalidades, sendo que o ator equipe pode: cadastrar presenças, advertências, alterar usuários, gerar relatórios dos atletas, além de visualizar a agenda, as chamadas (controle de presenças das aulas ministradas), categorias e atletas. Já o administrador possui permissão para as funções do ator equipe mais as opções de: cadastrar perfil, categoria, agenda, usuário, tipos de advertências, tipos de chamadas, atletas e responsáveis.

Olhando para os outros atores, responsável e atleta, estes possuem um pacote de funcionalidades menor, sendo as funcionalidades do atleta: somente visualizar seu cadastro e dos responsáveis, a agenda e as chamadas. Já o responsável, terá estas funcionalidades, mas poderá alterar seu cadastro e o do atleta.

A partir desta conversa também foi possível entender quais são as principais necessidades da ONG, sendo elas o cadastro de atletas e responsáveis, e principalmente, a gestão das presenças e advertências dos atletas. Com esta constatação ficou decidido que seria melhor focar nas funcionalidades que irão suprir essas necessidades, para assim ajudar com a principal demanda do Projeto.

Tendo em vista o que será produzido, foi iniciado a modelagem do banco de dados do sistema. Esta modelagem foi pensada levando em consideração o que foi especificado na reunião, ou seja, foram criadas modelos de tabelas para armazenar os dados dos atletas, responsáveis, chamadas e advertências.

Desta forma, foi elaborado um diagrama de modelagem de dados lógico dividido em duas partes: as tabelas relacionadas aos casos de uso do cadastro de atletas e responsáveis (Figura 11), e as tabelas relacionadas aos casos de uso das chamadas e advertências (Figura 12).

Nas tabelas relacionadas ao cadastro de atletas e responsáveis, temos as entidades: responsaveis, atletas, usuarios, perfis, categorias e atleta_responsaveis.

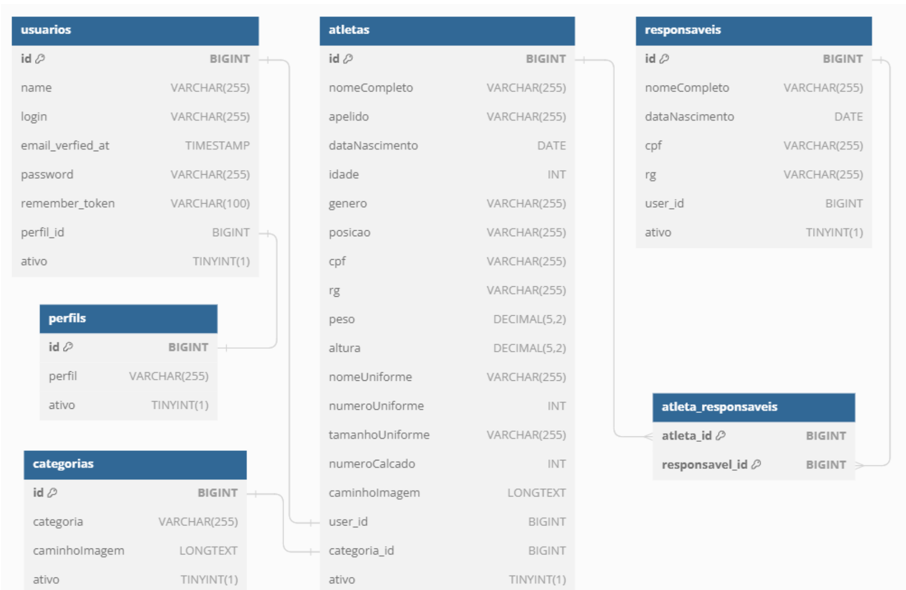

Figura 11. Diagrama de Modelagem de Dados Lógico - Parte 1

Entrando um pouco mais afundo no entendimento de cada tabela, podemos ressaltar, que a entidade perfis será a responsável por armazenar os atores (administrador, equipe, responsável e atleta) que foram citados anteriormente na Figura 10. Já a tabela de categorias será utilizada para guardar as categorias do Projeto Social, na tabela de usuarios, as informações sobre todos os utilizadores do sistema, sejam eles atletas, responsáveis ou da equipe Futpaz. E por último, notam-se as entidades atletas, responsaveis e atleta_responsaveis, nesses locais de armazenamento, se encontrará todas os dados dos jogadores e seus respectivos responsáveis, sendo que a tabela atleta_responsaveis está destinada a vincular os esportistas aos seus responsáveis.

Nas tabelas relacionadas aos casos de uso das chamadas e advertências temos as entidades: chamadas, chamada_tipos, advertencia_tipos, advertencias e presencas.

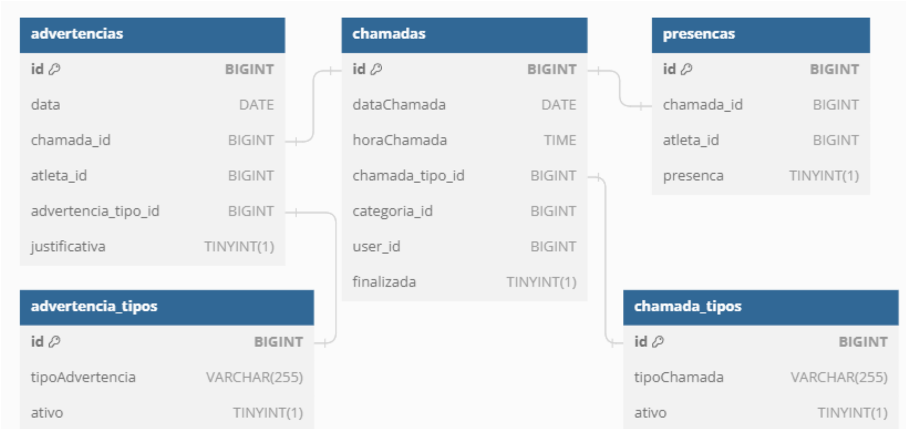

Figura 12. Diagrama de Modelagem de Dados Lógico - Parte 2

Nessas entidades pode-se notar que a tabela chamada será responsável por armazenar as presenças por categoria e data, já na tabela chamada_tipos, é onde será salvo os tipos de chamadas (treino, campeonato, amistoso, evento oficial, etc.). Na tabela presencas, está o vínculo entre a chamada e o atleta. E por fim, existem as tabelas advertencias e advertencia_tipos, onde são relatados as advertências dos atletas por chamada e os tipos de advertências (falta, atraso, vestuário inadequado, entre outros).

Após o término da construção da modelagem de banco de dados foram realizadas as instalações e configuração de ambiente de desenvolvimentos que eram necessárias para o sistema. Para o projeto de *front-end*, foram instalados pacotes e dependências cruciais para a utilização da biblioteca/*framework React* na ferramenta Vite, dessa forma configurando as bibliotecas do Material UI (biblioteca de componentes de código aberto que é implementado design do Google), dotenv (é um pacote para utilizar variáveis de ambiente), router (serve para resolver o problema de rotas de uma página*web*), axios (facilitador para integração entre um projeto *React* e uma API) e o tailwind (um *framework* de CSS que simplifica a criação de estilos para os componentes).

Paralelamente, na outra parte do sistema, o *back-end*, foi iniciado e configurado uma aplicação em Laravel, incluindo os processos de instalação de dependências, configuração do banco de dados, que envolve a conexão com MySQL e iniciar as primeiras *migrations* (uma funcionalidade do *Framework* para criar um versionamento do esquema do banco de dados a partir de classes PHP, e assim permitir que a manipulação das estruturas de tabelas sejam feitas pelo Laravel e não manualmente), assim como também foi instalada a biblioteca do *Passport* para utilização do OAuth2 para gerenciar toda a parte de autenticação do projeto. Essa etapa inicial de configuração foi importante para que o ambiente estivesse nas melhores condições, para facilitar o desenvolvimento subsequente.

Depois da finalização da preparação do ambiente total da aplicação, iniciou-se o desenvolvimento da primeira funcionalidade do sistema: o login.

5.1. Implementação do primeiro incremento

Para o login, o primeiro passo foi a configuração do *back-end* para realizar a conexão com o banco de dados e criar as primeiras tabelas do OAuth, através das *migrations* do Laravel, dessa forma estruturando o padrão de autenticação com *token*. Na sequência foi configurado a primeira rota de API para a conexão do *front-end* com o *back-end*, para ser possível validar as credenciais do usuário que solicitar acesso ao sistema. Ainda dentro dessa etapa, foi criado uma *Factory* (uma classe que cria um modelo de dados para gerar registros fictícios em seu banco de dados) e um *Seeder* (uma classe que utiliza do modelo criado na *Factory* para popular o banco de dados com dados iniciais) para a criação de um usuário inicial do sistema. Finalmente, para garantir o versionamento do *software*, foi configurado um repositório remoto no GitHub, que pode ser acessado a partir do link: `https://github.com/Rayssatsouza/futmanager_backend`.

O segundo passo foi o desenvolvimento do *front-end*, iniciando com a configuração da biblioteca axios para realizar requisições HTTP para a API criada no *back-end* para autenticação, conectando assim as duas partes do sistema. Por conseguinte, foi criada a tela inicial do sistema (Figura 13), com a funcionalidade de autenticação para acesso à plataforma. Nesta funcionalidade, foram aplicadas as configurações de variáveis de ambiente para que as informações principais de conexão possam ser acessadas por qualquer parte do sistema. Assim como no *back-end*, foi criado um repositório para o versionamento do *software* no GitHub (`https://github.com/Rayssatsouza/futmanager_frontend`).

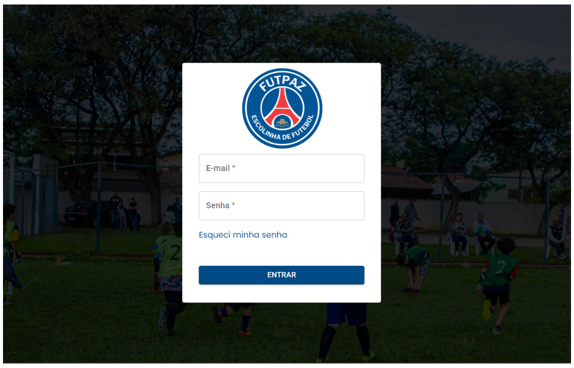

Figura 13. Tela de Login

5.2. Implementação do segundo incremento

O desenvolvimento seguiu com o *front-end*. Foram criados componentes chaves para o sistema, entre eles estão o componente de *NavBar, Header* e *Body* do sistema (Figura 14 apresenta o *layout* definido) para gerar a interface base para todas as páginas *web* que serão desenvolvidas. Em seguida foram configuradas as rotas (*Routes*) para gestão dos diferentes componentes que aparecerão em determinada página *web*, garantindo assim a navegabilidade dentro do sistema.

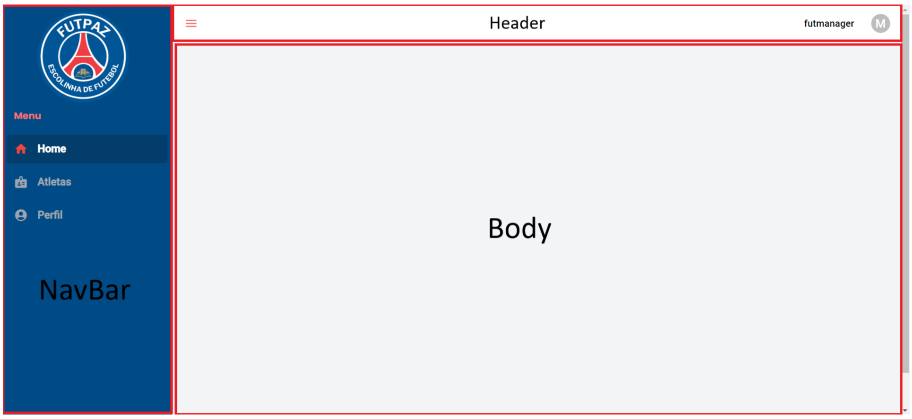

Figura 14. Demonstração de componentes na tela inicial

Construída a base para o desenvolvimento tanto do *front-end* quanto do *back-end*, iniciou-se a criação dos primeiros serviços da API do *back-end*, envolvendo os casos de uso: Cadastrar Perfil e Cadastrar Categoria (apresentados na Figura 10). Para a construção da API para cada um dos itens, foi necessária a criação de migrações (*migrations*), controladores (*controllers*), modelos (*models*) e rotas (*routes*) para cada entidade. Além de realizar alterações nos *seeders* para assegurar que todas as categorias fossem salvas de forma correta no banco de dados.

Já no *front-end*, foram desenvolvidas as telas de cadastro de categorias e perfis, onde o ator Administrador (Figura 10) pode inserir as informações necessárias para essas entidades. Ademais, foi configurado o armazenamento (*local storage*) no navegador para registrar as informações do usuário logado e assim poder gerenciar o *token* de autenticação do sistema, a partir disso, foi possível implementar a exibição do usuário logado no componente da *Header*.

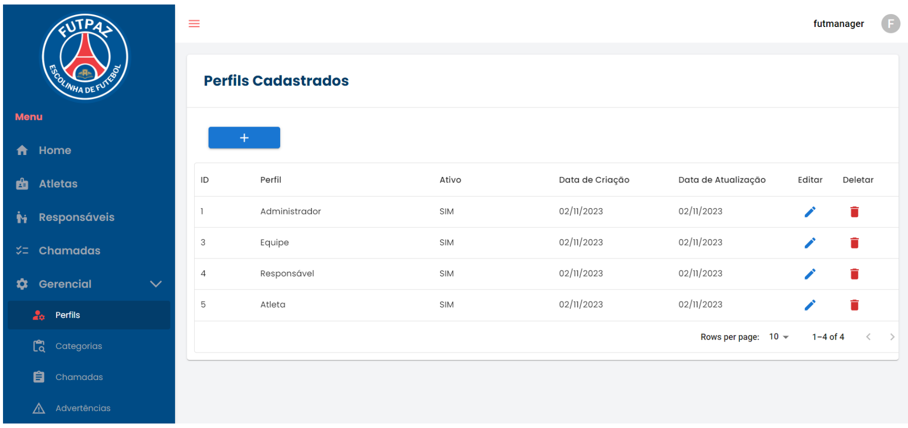

Figura 15. Tela de cadastro de Perfis

Considerando o caso de uso de Cadastro de Perfis, será apresentada uma demonstração da estrutura do *back-end* da aplicação, abrangendo as fases de *migration*, *model*, *controller* e *routes*. Primeiramente, é criada uma *migration* com nome respectivo da tabela (Figura 16), para criar a entidade dentro do banco de dados da aplicação. Nesta classe de versionamento do banco de dados, ficam duas funções: *up* e *down*. Na *up*, fica o que é será criado a partir dessa *migration*, nesse caso a criação da tabela perfis com as suas respectivas colunas (Figura 11). Já a função *down* tem por objetivo desfazer o que é codificado em *up*, para um eventual momento em que seja necessário voltar a execução de uma *migration*, neste exemplo, a função *down* somente faria a exclusão da tabela perfis.

```
return new class extends Migration
{
    /*** Run the migrations.*/
    public function up(): void
    {
        Schema::create("perfils", function (Blueprint $table) {
            $table->id();
            $table->string('perfil');
            $table->boolean('ativo');
            $table->timestamps();
        });
    }

    /*** Reverse the migrations.*/
    public function down(): void
    {
        Schema::dropIfExists('perfils');
    }
};
```

Figura 16. *Migration* da tabela perfis

Em sequência, é codificado a *model* da tabela perfis, basicamente onde fica a regra de negócio daquela entidade. Na *model* se localizam os atributos/colunas daquela classe e também funções para tratar determinado atributo. Neste exemplo (Figura 17), foi declarado o vetor *$fillable* com os atributos da tabela perfis e mais duas funções (*getCreatedAtAttribute* e *getUpdatedAtAttribute*) para tratar o campo de data de criação e data de atualização, e retorná-lo no formato brasileiro, conforme observado na Figura 15.

```
class Perfil extends Model
{
    use HasFactory;

    /** * The attributes that are mass assignable.*
     * @var array<int, string>*/
    0 references
    protected $fillable = [
        'perfil',
        'ativo'
    ];
    0 references | 0 overrides
    public function getCreatedAtAttribute($value)
    {
        return \Carbon\Carbon::parse($value)->format('d/m/Y'); // H:i:s
    }
    0 references | 0 overrides
    public function getUpdatedAtAttribute($value)
    {
        return \Carbon\Carbon::parse($value)->format('d/m/Y'); // H:i:s
    }
}
```

Figura 17. Model da tabela perfis

O próximo passo é a configuração do *controller*, neste arquivo de código ficam as requisições que o *model* irá receber, ou seja, sempre que precisar realizar uma inclusão de um novo item, alteração do item, seleção do item, exclusão do item, ou qualquer outra manipulação do gênero, ela será realizada através de uma função no *controller*. Neste exemplo (Figura 18) envolvendo a *model* perfil, é possível visualizar as funções *get*, *create*, *delete* e *edit* (configurações de um CRUD) para manipular os dados.

Figura 18. *Controller* da tabela perfis

Por fim, nesta demonstração do *back-end*, existe a última etapa, que é a criação de uma rota (*route*). As rotas servem para que o *front-end* consiga realizar requisições para o *back-end* por meio de uma API Rest, asssim as *routes* consistem em um caminho configurado para acionar uma determinada função de um *controller*. Na Figura 19 é possível ver cinco rotas configuradas para a tabela perfis. Cada rota tem qual é o tipo de requisição, o seu caminho, qual o *controller* responsável e qual é a função do *controller* que será executada.

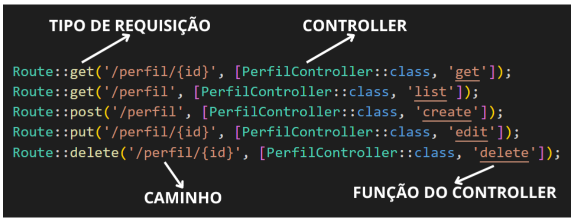

Figura 19. *Routes* da tabela perfis

Essa sequência de ações é importante para criar todas as configurações do *back-end* relacionadas a uma determinada tabela e elas são implementadas a cada nova entidade que é criada no projeto.

5.3. Implementação do terceiro incremento

No terceiro incremento do desenvolvimento do sistema, foi desenvolvidos os outros casos de uso que eram cadastros cruciais para o funcionamento das demais funcionalidades como: Cadastrar os Tipos de Chamada, Cadastrar os Tipos de Advertência e também o Cadastrar Usuários do Sistema (Figura 10).

Inicialmente, o desenvolvimento foi para o *back-end*, onde foram criadas as tabelas chamada_tipos e advertencia_tipos por meio do processo de *migrations*. Com o intuito de gerenciar essas novas entidades, foram desenvolvidas rotas para possibilitar as operações básicas do CRUD (*Create, Read, Update, Delete*). Para cada uma dessas entidades e para a tabela *users*, foram criadas APIs específicas, envolvendo migrações (*migrations*), controladores (*controllers*), modelos (*models*) e rotas (*routes*).

No *front-end*, foram desenvolvidas as telas específicas para a exibição dos tipos de advertências cadastrados e telas para o cadastro desses novos tipos. Do mesmo modo, foram criadas telas destinadas à visualização dos tipos de chamadas cadastrados e ao cadastro de novos tipos de chamadas. Ainda neste incremento, a tela de cadastro de usuários foi implementada, podendo agora cadastrar novos usuários para a aplicação.

Depois da finalização do terceiro incremento, tendo as funcionalidades pilares, ou seja, os cadastros básicos para funcionamento do sistema, foi possível iniciar o desenvolvimento das principais funcionalidades: Cadastrar atleta e responsável e Cadastrar presenças e advertências.

5.4. Implementação do quarto incremento

No quarto incremento deste projeto, o foco voltou-se para os casos de uso que se referem a funcionalidade de cadastro de atletas e responsáveis, para isso foi feita a configuração no *back-end* e a criação de tabelas essenciais para o gerenciamento dos atletas e responsáveis das equipes do Projeto Social. Foram criadas as tabelas atletas, responsaveis, e atleta_responsaveis por meio do processo das *migrations*. Para possibilitar o controle dessas novas entidades, foram desenvolvidas rotas para realizar as operações de inserção, edição, listagem e deleção de dados. Além disso, APIs específicas foram criadas para cada uma dessas entidades, envolvendo as *migrations*, *controllers*, *models* e *routes*.

No *front-end*, o presente incremento resultou na criação de interfaces adicionais para auxiliar na visão das informações de gerenciamento da equipe. Uma tela foi desenvolvida para a visualização das categorias cadastradas, esta mesma tela serve com um primeiro filtro para o acesso a listagem de atletas, permitindo uma organização eficaz das informações. Outra tela foi implementada para exibir os atletas associados a uma categoria selecionada, facilitando a identificação e gestão dos membros de uma determinada categoria. Em sequência, foram criadas telas dedicadas ao cadastro de atletas (Figura 20) e à visualização dos responsáveis cadastrados. Além disso, uma interface específica foi desenvolvidas para vincular o cadastro de responsáveis com os atletas correspondentes.

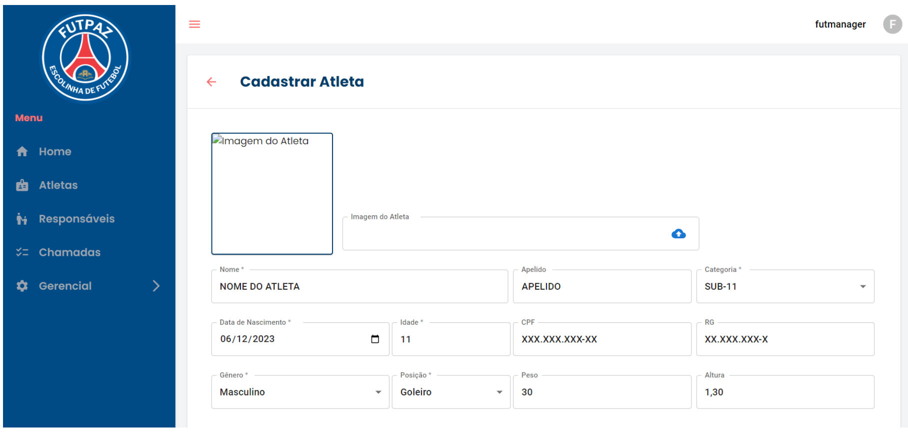

Figura 20. Visualização parcial da tela de cadastro de atletas

5.5. Implementação do quinto incremento

No quinto incremento do desenvolvimento do sistema, a construção se iniciou no *back-end*, com a criação das tabelas relacionadas aos casos de uso (Figura 10): Cadastrar Presença, Cadastrar Advertência e Visualizar presenças e advertências. Levando em consideração a modelagem de dados dessas entidades na Figura 12, foram criadas as tabelas de chamadas, advertencias, e presencas por meio do processo de migração. Para realizar a gestão dessas novas entidades, foram desenvolvidas rotas que permitem as operações essenciais do um *controller* (CRUD). Além disso, APIs específicas foram implementadas para cada uma dessas entidades, abrangendo as *migrations*, *controllers*, *models* e *routes*.

No *front-end*, foi realizada a criação de interfaces adicionais para as funcionalidades propostas. Foi desenvolvida uma tela para a visualização das chamadas cadastradas, gerando uma forma organizada de acompanhar essas informações. Outra tela foi implementada para o cadastro de novas chamadas, onde é possível preencher a data que ocorreu este evento, o horário, qual foi o tipo de evento (informações coletadas do cadastro de tipos de chamada), a categoria e se ela foi finalizada ou não. Além disso, uma tela integrada foi criada para o cadastro de presenças dos atletas, juntamente com o registro de advertências associadas a uma chamada específica, para este registro de advertências existe o preenchimento: data da advertência, o atleta que está a recebendo, tipo de advertência e uma justificativa. Finalmente, uma tela específica foi desenvolvida para a visualização das presenças registradas em uma determinada chamada (Figura 21), proporcionando uma visão de qual evento o atleta compareceu.

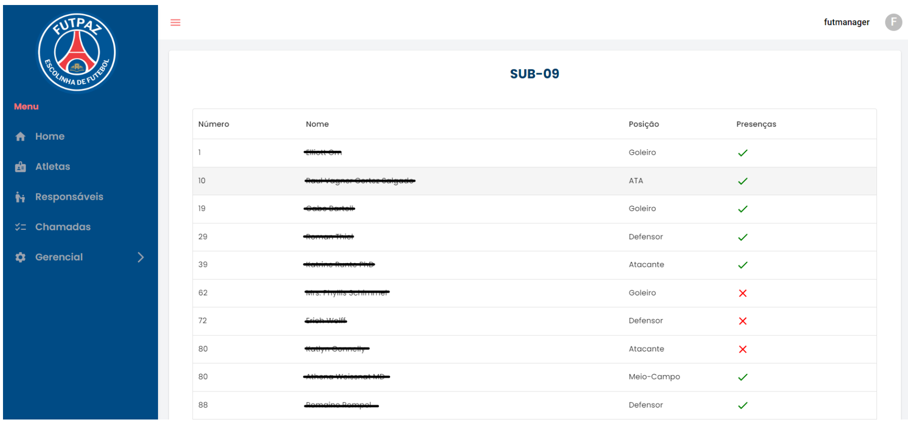

Figura 21. Presenças registradas em uma determinada chamada

5.6. Implementação do sexto incremento

No sexto e último incremento, o foco foi criar as interfaces voltadas para os atores Responsável e Atleta, durante o desenvolvimento dessas telas, foi possível identificar uma forma de unir os dois atores, ou seja, todas as funcionalidades que antes eram aplicadas ao Responsável, também serem aplicadas ao Atleta, e dessa forma cada família do Futpaz teria um acesso por atleta, simplificando a estrutura por trás da aplicação e auxiliando em diminuir o tempo de desenvolvimento.

A partir dessa constatação, foram criadas as telas dedicadas à visualização detalhada das presenças (Figura 22) e advertências em relação a cada atleta individualmente, essas telas dizem respeito ao caso de uso (Figura 10): Visualizar presenças e Advertências , relacionado com ambos os atores (Responsável e Atleta). Além dessa tela, também foi desenvolvido um menu onde é possível editar algumas informações do atleta e do responsável, para suprir as funcionalidades: Alterar Cadastro do Responsável e Alterar Cadastro do Atleta (Figura 10).

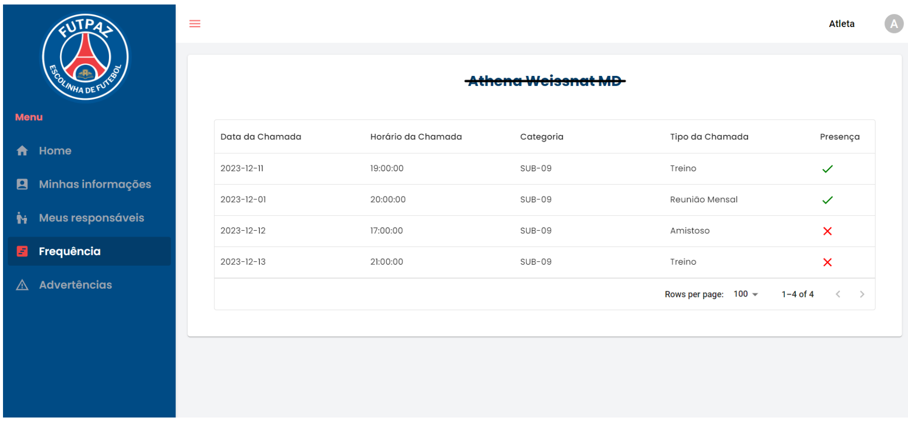

Figura 22. Todas as presenças registradas de um atleta

Outra etapa deste incremento, foi a edição da tela inicial para incluir a visualização das fotos associadas à categoria e aos atletas/membros da equipe, proporcionando uma identificação visual mais completa e personalizada. A configuração de acesso aos menus também foi editada, ajustando-se conforme os perfis de cada usuário. Isso garantiu que as funcionalidades do sistema ficassem disponível de acordo com as permissões atribuídas a cada perfil, conforme especificado anteriormente no Diagrama de Caso de Uso (Figura 10).

Além disso, foi implementada uma funcionalidade muito importante que serve para manter a autenticação do usuário ativa. O sistema foi configurado para atualizar automaticamente o *token* de atualização (*refresh token*) antes de expirar, dessa forma é possível garantir que os usuários possam continuar utilizando o sistema de forma contínua e sem interrupções.

Com o sexto incremento, o desenvolvimento do sistema *web* para gerenciamento de equipe de futebol é concluído.

6. Conclusão

Por fim, este trabalho foi criado na intenção de informatizar processos da comissão técnica do Projeto Social Futpaz, que anteriormente eram feitos manualmente, através de um sistema *web* para gerenciamento de uma equipe de futebol. Neste sistema foram implementados algumas funcionalidade que podem ser resumidas em um cadastramento de alunos do projeto e seus responsáveis, um controle de presença e advertências durante as aulas ministradas. Para realizar essas implementações foi utilizada uma metodologia incremental, que consiste na ideia de realizar o desenvolvimento do projeto em pequenas partes, chamadas incrementos, seguindo este conceito o desenvolvimento deste aplicativo ficou divido em seis incrementos que resultaram em uma primeira versão do *software*.

Para desenvolver esse sistema foram utilizadas as linguagens de programação *PHP* e *Javascript*, por meio de *Frameworks*: o *React* para o *front-end* e o Laravel para o *back-end*. Além da utilização dessas ferramentas, também foram implementadas bibliotecas que auxiliaram no desenvolvimento, sendo os recursos voltados para o *front-end*: a ferramenta Vite, a biblioteca de componentes MUI (*Material UI*), o pacote de variáveis de ambiente dotenv, o facilitador de integração com API axios e o *Framework* de estilização

Tailwind. Já para o *back-end* o recursos utilizado foi a biblioteca *Passport*, que serve para gerenciar a autenticação de um sistema através da tecnologia do OAuth2 (controlador de *Token*).

No decorrer dos incrementos foram desenvolvidas a grande maioria das funcionalidades propostas no escopo original do trabalho (Diagrama de Caso de Uso - Figura 10), sendo essas funcionalidades: cadastrar perfil, cadastrar categoria, cadastrar usuário, cadastrar tipos de advertência, cadastrar tipos de chamadas, cadastrar atleta, cadastrar responsável, cadastrar presença, cadastrar advertência, alterar usuário, visualizar presenças e advertências, visualizar categorias e atletas, alterar cadastro responsável, alterar cadastro atleta.

Porém, conforme citado no início da seção de desenvolvimento, ficou acordado com a comissão técnica a priorização de algumas funcionalidades em detrimento de outras, sendo as funcionalidades priorizadas as desenvolvidas nesse projeto e as demais não implementadas ficam para trabalhos futuros, isso envolve os casos de uso: cadastrar agenda, visualizar agenda e gerar relatórios dos atletas, e também a melhoria nas configurações de responsividades de cada tela do projeto. Outros trabalhos futuros que podem ser implementados seriam a averiguação de segurança do *software*, a usabilidade proposta, o desempenho do mesmo, entre outros trabalhos do gênero.

A proposta deste trabalho foi pensada em correlacionar o um contexto social esportivo, juntamente com as competências desenvolvidas no curso de Análise e Desenvolvimento de Sistemas. Esse conversa entre as áreas veio através de disciplinas e conteúdos vistos no decorrer da graduação, sendo algumas das disciplinas que contribuíram para esse desenvolvimento: Banco de Dados I, Algoritmos e Programação, Desenvolvimento Web, Engenharia de Software, Metodologias Ágeis. Além de outros conhecimentos adquiridos por outros recursos, como: desenvolvimento em *React*, Laravel e entendimento de utilização das diversas bibliotecas utilizadas no trabalho.

Referências

dos Santos, L. O. (2020). Um estilo arquitetural baseado na norma ISO/IEC 30141 para sistemas de internet das coisas. Teste (Doutorado em Sistemas e Computação) - Programa de Pós-Graduação em Sistemas e Computação, Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, 173p.

Egasse, A. and Melki, N. (2012). Sport easy. `https://www.sporteasy.net/pt/home/`. [Online; Acessado em 15 de Março de 2023].

Fujita, L. (2018). Qual é o esporte mais praticado no brasil? `https://super.abril.com.br/mundo-estranho/qual-e-o-esporte-mais-praticado-no-brasil/`. [Online; Acessado em 21 de Setembro de 2022].

Futpaz (2022). @futpazescolinha. `https://www.instagram.com/futpazescolinha/`. [Online; Acessado em 28 de Novembro de 2023].

Fácil, C. (2018). Copa fácil: O jeito mais fácil de gerenciar seu campeonato!. `https://copafacil.com/`. [Online; Acessado em 25 de Setembro de 2023].

Higor (2013). Introdução ao padrão MVC. `https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308`. [Online; Acessado em 18 de Junho de 2023].

Laravel (2023). Laravel: The PHP framework for web artisans. https://laravel.com/. [Online; Acessado em 25 de Setembro de 2023].

Lessa, R. O. and Lessa Junior, E. O. (2009). Modelos de processos de engenharia de software. *Princípios da Engenharia de Software – Universidade do Sul de Santa Catarina (UNISUL)*. https://ead.uepg.br/apl/sigma/assets/editais/PS0059E0080.pdf [Online; Acessado em 08 de Novembro de 2022].

Luciano, J. and Avles, W. J. B. (2011). Padrão de arquitetura MVC: Model-view-controller. *Revista EPeQ Fafibe*, 1(3):102–107.

Marcotte, E. (2010). Web design responsivo. https://alistapart.com/article/responsive-web-design/. [Online; Acessado em 12 de Setembro de 2023].

Marin, L. and Reuter, N. (2016). Ritmo do esporte. http://equipes.ritmodoesporte.com.br/landing-pages/ritmoequipes#funcionalidades. [Online; Acessado em 15 de Março de 2023].

Meta (2023). React: The library for web and native user interfaces. https://react.dev/. [Online; Acessado em 25 de Setembro de 2023].

Murilo, C. and Bittencourt, J. (2021). O que é um framework?. https://www.alura.com.br/artigos/framework-o-que-e-pra-que-serve-essa-ferramenta. [Online; Acessado em 25 de Setembro de 2023].

Nery, C. and Britto, V. (2022). Internet já é acessível em 90,0% dos domicílios do país em 2021. https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/34954-internet-ja-e-acessivel-em-90-0-dos-domicilios-do-pais-em-2021. [Online; Acessado em 26 de Agosto de 2023].

Oracle (2022). O que é um banco de dados. https://www.oracle.com/br/database/what-is-database/.

Sommerville, I. (2019). *Engenharia de software*. Pearson Universidades, 10th edition.

Store, P. (2018). Copa fácil: Organize torneios. https://play.google.com/store/apps/details?id=com.copafacil. [Online; Acessado em 25 de Setembro de 2023].

Documento Digitalizado Público

TCC - Anexo I

Assunto: TCC - Anexo I
Assinado por: Andre Constantino
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 09/03/2024 17:31:31.

Este documento foi armazenado no SUAP em 09/03/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1605930

Código de Autenticação: 380ba8cc40