\CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

JOÃO VITOR GIRARDI

INTERFACE DE USUÁRIO PARA UM SISTEMA DE PRESCRIÇÃO E AVALIAÇÃO DE TREINOS EM ACADEMIAS

RIO DO SUL
2024

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

JOÃO VITOR GIRARDI

INTERFACE DE USUÁRIO PARA UM SISTEMA DE PRESCRIÇÃO E AVALIAÇÃO DE TREINOS EM ACADEMIAS

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Cleber Nardelli

RIO DO SUL
2024

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

JOÃO VITOR GIRARDI

INTERFACE DE USUÁRIO PARA UM SISTEMA DE PRESCRIÇÃO E AVALIAÇÃO DE TREINOS EM ACADEMIAS

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: Esp. Cleber Nardelli

Banca Examinadora:

_____

Professor M.e. Fernando Andrade Bastos

_____

Professor M.e. Marciel de Liz Santos

Rio do Sul, 21 de Novembro de 2024.

Se eu tivesse oito horas para derrubar uma árvore, passaria seis afiando meu machado (Abraham Lincoln).

Dedico esse trabalho a minha família e amigos.

AGRADECIMENTOS

Agradeço primeiramente a minha família por sempre me apoiar em minhas escolhas, a minha namorada por sempre estar presente comigo durante o desenvolvimento deste trabalho me a poiando e dedico a mim mesmo por não ter desistido de finalizar o curso e por não ter sucumbido diante das dificuldades encontradas.

Agradeço também a meu orientador Cleber Nardelli por todo o conhecimento compartilhado durante o desenvolvimento deste trabalho e aos demais professores que fizeram parte da minha formação acadêmica a todos o meu muito obrigado.

RESUMO

O setor de academias vem crescendo com o passar do tempo, com isso cada vez mais pessoas buscam por elas para ter uma vida saudável e com essa crescente demanda é necessário evoluir os sistemas para atender de forma satisfatória, sendo este o principal motivador do presente trabalho apresentando o desenvolvimento de uma interface para um sistema de prescrição e avaliação de treinamento, com foco na digitalização e modernização do processo, substituindo fichas em papel por um sistema integrado. O sistema é composto por um aplicativo web para professores para gerenciamento de treinamentos e um aplicativo mobile para que os alunos acessem rapidamente os programas de treinamento e monitorem seu desempenho. O aplicativo web tem foco na gestão da academia, enquanto o aplicativo móvel oferece aos alunos uma interface apropriada para monitorar mais facilmente seu progresso. O sistema se conecta a uma API que fica responsável camada de back-end, tanto para o aplicativo quando para o sistema web. Para o desenvolvimento fora realizada uma pesquisa aplicada e descritiva, bem como o levantamento de requisitos e revisão de literatura das principais técnicas de desenvolvimento de sistemas. Foram utilizadas tecnologias como React e Next.Js e React Native, seguindo as boas práticas de desenvolvimento e arquitetura REST. O sistema visa otimizar a comunicação entre professores e alunos, contribuindo para a personalização dos treinos e para a motivação contínua dos alunos. A solução proposta busca aumentar a eficiência do acompanhamento e a satisfação dos usuários.

Palavras-Chave: Treino, Interface, React.

ABSTRACT

The fitness industry has been growing steadily over time, with more and more people seeking gyms to lead healthier lives. However, with the increasing demand, we need to evolve our systems to meet this growth. This project presents the development of an interface for a gym training prescription and evaluation system, focusing on digitizing and modernizing the process by replacing paper forms with an integrated system. The system consists of a web application for trainers to manage workouts and a mobile app for students to quickly access training programs and monitor their performance. The web app allows for workout registration and exercise management, while the mobile app offers students daily and weekly workout sessions to easily track their progress. The system connects to an API responsible for user authentication, as well as provisioning and storing data for both the mobile and web applications. The methods used include applied and descriptive research, as well as requirement gathering and a literature review on key system development techniques. Technologies such as React, Next.js, and React Native were employed, following best practices in development and REST architecture. The system aims to optimize communication between trainers and students, contributing to personalized workouts and ongoing motivation for students. The proposed solution seeks to enhance efficiency in tracking progress and overall user satisfaction.

Keywords: Training, Interface, React.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CLI	Command-Line Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
IDOR	Insecure Direct Object Reference
JSON	JavaScript Object Notation
JSX	JavaScript XML
OSI	Open Systems Interconnection
REST	Representational State Transfer
URL	Uniform Resource Locator

SUMÁRIO

1. INTRODUÇÃO

A prática de atividades físicas tem ganhado cada vez mais espaço na rotina dos brasileiros. Segundo informações do site (MEDICINA SA, 2024), o Brasil conta com mais de 32.011 academias ativas. Destes 72,75% pequenos negócios (microempresas) com faturamento anual de até R$ 360 mil. A procura por uma vida mais saudável e ativa tem estimulado o desenvolvimento dessas instituições, que se transformam em um local cada vez mais procurado por indivíduos com objetivos de redução de peso, ganho de massa muscular ou manter a saúde em dia.

Pensando neste cenário, as academias enfrentam um desafio significativo, a gestão eficiente e a criação de treinos personalizados para seus usuários. Este problema se torna ainda mais complexo diante da diversidade de perfis dos alunos, que variam de acordo com seus objetivos e níveis de condicionamento físico. Professores e treinadores necessitam de ferramentas que facilitem a criação de planos de treino personalizados e que permitam um acompanhamento contínuo e eficaz dos alunos.

Hoje o desenvolvimento de treinos se dá por fichas de treinos impressas aonde o professor deve marcar quais treinos devem ser realizados, caso o treino mude ele necessita pegar uma nova ficha e refazer tudo novamente, caso o aluno o professor não se atentem, a ficha pode ser perdida ou desatualizada, deixando o aluno por vários meses com o mesmo treino.

Para atender a essa demanda, surge a proposta de desenvolver a camada de front-end como parte de um sistema web, para consumir *endpoints* produzidos por uma API back-end para os professores e uma aplicação móvel para os alunos. A aplicação web é destinada aos professores gerenciarem os treinos de forma organizada, enquanto a aplicação móvel para os alunos por meio de acesso fácil aos seus planos de treino, contribuindo para uma melhor adesão e acompanhamento das atividades prescritas.

O objetivo principal é otimizar a comunicação entre professores e alunos, além de proporcionar um ambiente mais dinâmico e eficiente para a gestão dos treinos. Com isso, espera-se melhorar a qualidade do acompanhamento dos alunos, aumentar a satisfação dos usuários e promover um ambiente mais colaborativo e motivador dentro das academias.

Este trabalho, além do capítulo introdutório, inclui uma revisão da literatura que examina as tecnologias aplicadas no desenvolvimento do protótipo, dando explicações sobre cada uma. No terceiro capítulo aborda a metodologia da pesquisa, detalhando o processo

utilizado. No quarto capítulo, são apresentados o desenvolvimento da aplicação e a análise do projeto. Por fim, no último capítulo é apresentado as considerações finais.

1.1 PROBLEMA DE PESQUISA

Como operacionalizar de forma simplificada a gestão de treinos para usuários de uma academia, mantendo-os organizados ao longo do tempo?

1.2 OBJETIVOS

1.2.1 Geral

- Desenvolver camada de front-end que atenda às necessidades de prescrição e acompanhamento dos alunos de uma academia.

1.2.2 Específicos

- Identificar as dificuldades operacionais de academia;
- Levantar os requisitos para a criação do sistema;
- Avaliar a melhor tecnologia para o desenvolvimento;
- Desenvolver protótipos das telas com base nos requisitos;
- Desenvolver o aplicativo mobile front end que os alunos utilizaram com base os protótipos;
- Desenvolver a aplicação Web de front-end para gerenciamento da academia com base nos protótipos;
- Desenvolver chamadas para os endpoints de back-end;

1.3 JUSTIFICATIVA

O cuidado com a saúde tem se tornado uma prioridade para muitas pessoas, levando a um aumento na procura por academias. No entanto, muitas dessas instituições ainda utilizam métodos antiquados para o controle de seus espaços e treinos, como planilhas e fichas de treino impressas em papel. Essa falta de modernização pode dificultar o acompanhamento do

profissional de educação física com os alunos da academia e com isso o aluno pode ter resultados negativos quanto a sua performance e objetivos.

Atualmente o controle é realizado de forma muito manual com fichas de treino, em que os professores imprimem e marcam qual exercícios o aluno deve realizar. O aluno por sua vez, precisa procurar sua ficha diariamente e lembrar quais os últimos exercícios que realizou. Sem um sistema, dificulta o acompanhamento ao aluno, se o mesmo está realizando as atividades corretas, ou se está encontrando dificuldades.

Além disso, essas fichas de treino, muitas vezes, ficam desatualizadas. Se o aluno não solicitar um novo treino ao profissional, ele pode acabar mantendo o treino por um longo período. Isso pode levar a uma série de problemas, como a desistência do esporte, a criação de treinos próprios sem a orientação adequada, o que pode resultar em lesões, e, consequentemente, a desistência da prática de exercícios.

Por outro lado, realizar avaliação do aluno também se torna um desafio pois dificulta o acesso dele ao resultado da avaliação, muitas vezes sendo enviada por documentos .pdf. A falta de uma interface onde o aluno e personal possam acompanhar as avaliações e planejar o melhor treino para a melhor evolução do aluno pode levar a uma estagnação do desempenho do aluno, uma plataforma que permita acompanhar e prescrever treinos não só facilitaria o trabalho do professor como também manteria o aluno mais engajado.

A implementação de um sistema moderno de controle e acompanhamento pode ser a solução para esses problemas. Com ele, a academia poderá ter uma proximidade maior com o aluno, o que pode resultar em menos desistências. Atualmente estudos apontam que 65% dos alunos desistem nos primeiros 3 meses. Segundo a (Dynamo, 2024), o ponto crucial é a personalização dos treinos: "Crie planos de treino individualizados que atendam às necessidades e objetivos específicos de cada aluno. Ao verem progresso e alcançarem metas pessoais, os alunos tendem a permanecer mais engajados e motivados.". Com o uso de um sistema informatizado, permitirá aos alunos um melhor acompanhamento dos treinos bem como a evolução, garantindo que eles estejam sempre praticando os exercícios mais adequados para suas necessidades e objetivos. Espera-se com essas inovações que a academia melhore a oferta de serviços, atraindo mais alunos, garantindo satisfação e o bem-estar de todos.

2. REFERENCIAL TEÓRICO

Agora serão apresentados informações e dados sobre os temas referentes ao projeto e principalmente sobre as tecnologias e ferramentas que serão utilizadas para o desenvolvimento.

2.1 ENGENHARIA DE REQUISITOS

Segundo Sommerville (2011, p. 3), “Engenharia de software é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.”.

> Software consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa quanto na virtual, descrevendo a operação e o uso dos programas (Pressman e Maxim, 2021, p.5).

Brookshear (2013, p. 264) afirma que “a aplicação de tecnologias computacionais ao processo de desenvolvimento de software, resultando na chamada engenharia de software auxiliada por computador (CASE), continua a agilizar e a simplificar o processo de desenvolvimento de software.”.

Para Pressman e Maxim (2021) o levantamento de requisitos é um processo que parece simples, mas isto, na verdade é muito difícil. Uma das partes importantes deste processo é entender quais são as metas do negócio, pois elas ajudam a explicar os requisitos para os envolvidos, e podem ser usadas para resolver conflitos.

2.2 HTTP

Conforme nos diz Saudate (2013, p.14) “Este protocolo foi desenvolvido de maneira a ser o mais flexível possível para comportar diversas necessidades diferentes.”

> O protocolo HTTP (*HyperText Transfer Protocol* - Protocolo de Transferência de Hipertexto) data de 1996, época em que os trabalhos conjuntos de Tim Berners-Lee, Roy Fielding e Henrik Frystyk Nielsen levaram à publicação de uma RFC (*Request for Comments*) descrevendo este protocolo. Trata-se de um protocolo de camada de aplicação (segundo o modelo OSI) e, portanto, de relativa facilidade de manipulação em aplicações. (Saudete, 2013, p.13)

Saudate (2013, p.15) ainda complementa com “A versão corrente do HTTP, 1.1, define oficialmente oito métodos - embora o protocolo seja extensível em relação a estes métodos. Hoje, estes oito são: • GET • POST • PUT • DELETE • OPTIONS • HEAD • TRACE • CONNECT”

Os códigos mais importantes e mais utilizados são, os representados no Quadro 1.

Quadro 1 - Códigos mais utilizados HTTP

Status	Nome	Descrição
200	OK	Indica que a operação indicada teve sucesso.
201	Created	Indica que o recurso desejado foi criado com sucesso.
202	Accepted	indica que a solicitação foi recebida e será processada em outro momento. É tipicamente utilizada em requisições assíncronas, que não serão processadas em tempo real.
204	No Content	Usualmente enviado em resposta a uma requisição PUT, POST ou DELETE, onde o servidor pode recusar-se a enviar conteúdo.
206	Partial Content	Utilizado em requisições GET parciais, ou seja, que demandam apenas parte do conteúdo armazenado no servidor (caso muito utilizado em servidores de download).
303	See Other	É utilizado quando a requisição foi processada, mas o servidor não deseja enviar o resultado do processamento. Ao invés disso, o servidor envia a resposta com este código de status e o cabeçalho Location, informando onde a resposta do processamento está.
304	Not Modiied	É utilizado, principalmente, em requisições GET condicionais - quando o cliente deseja ver a resposta apenas se ela tiver sido alterada em relação a uma requisição anterior.
400	Bad Request	É uma resposta genérica para qualquer tipo de erro de processamento cuja responsabilidade é do cliente do serviço.
401	Unauthorized	Utilizado quando o cliente está tentando realizar uma operação sem ter fornecido dados de autenticação (ou a autenticação fornecida for inválida)
403	Forbidden	Utilizado quando o cliente está tentando realizar uma operação sem ter a devida autorização.
404	Not Found	Utilizado quando o recurso solicitado não existe.
409	Conflict	Utilizado quando há conflitos entre dois recursos. Comumente utilizado em resposta a criações de conteúdos que tenham restrições de dados únicos -
500	Internal Server Error	É uma resposta de erro genérica, utilizada quando nenhuma outra se aplica.
503	Service Unavailable	Indica que o servidor está atendendo requisições, mas o serviço em questão não está funcionando corretamente.

Fonte: Elaborado a partir de Saudate (2013)

2.3 REST

Conforme nos explica (Saudate, 2013, p.4) “REST significa *Representational State Transfer* (ou Transferência de Estado Representativo, em tradução livre), e é um estilo de desenvolvimento de web *services* que teve origem na tese de doutorado de Roy Fielding. Este,

por sua vez, é coautor de um dos protocolos mais utilizados no mundo, o HTTP (*Hypertext Transfer Protocol*)".

> Modelo REST foi desenvolvido por Roy Fielding, um dos criadores do protocolo HTTP. Portanto, fica naturalmente evidente, para quem conhece ambos os modelos, as semelhanças entre ambos - sendo que, na realidade, REST é idealmente concebido para uso com o protocolo HTTP. Portanto, para ser um bom usuário de REST, é necessário ter um bom conhecimento do protocolo HTTP (Saudate, 2013, p.13).

Saudate (2013, p.29) diz que "REST é baseado nos conceitos do protocolo HTTP. Além disso, este protocolo é a base para a web como a conhecemos, sendo que a própria navegação nesta pode ser encarada como um uso de REST".

> JSON é uma sigla para Javascript *Object Notation*. É uma linguagem de marcação criada por Douglas Crockford e descrito na RFC 4627, e serve como uma contrapartida a XML. Tem por principal motivação o tamanho reduzido em relação a XML, e acaba tendo uso mais propício em cenários onde largura de banda (ou seja, quantidade de dados que pode ser transmitida em um determinado intervalo de tempo) é um recurso crítico. (Saudete, 2023, p.57).

O autor ainda complementa "Para interagir com as URL's, os métodos HTTP são utilizados. A regra de ouro para esta interação é que URL's são substantivos, e métodos HTTP são verbos. Isto quer dizer que os métodos HTTP são os responsáveis por provocar alterações nos recursos identificados pelas URL's." (Saudate, 2013, p.31).

2.4 AXIOS

Conforme a documentação do AXIOS (2024), Axios é um cliente HTTP baseado em promessas para node.js e navegador. É isomórfico (pode ser executado no navegador e node.js com a mesma base de código). No lado do servidor, ele usa o módulo node.js nativo, enquanto no cliente (navegador) ele usa XMLHttpRequests.

Axios facilita que desenvolvedores façam requisições HTTP, utilizando todos os verbos, como POST, PUT, GET, DELETE etc. Permitindo a manipulação da requisição, seja pela rota, *headers*, *params*, *body*, *interceptors*, etc. Garantindo que todas as necessidades durante o desenvolvimento sejam atendidas por ferramentas presentes no Axios. (AXIOS, 2024).

2.5 HTML (HYPERTEXT MARKUP LANGUAGE)

Conforme nos explica Oliveira (2020, p.17) “A linguagem HTML é a base de criação de qualquer página para a web, porém se trata de uma linguagem composta por elementos estáticos e está fortemente focada na composição e na formatação de documentos.”

> Assim como são necessárias linguagens de programação para o desenvolvimento de programas e aplicativos, também é necessária uma linguagem para criação de páginas de sites. Essa linguagem é denominada HTML (HyperText Markup Language, em português Linguagem de Marcação de Hipertexto). Porém, em vez de comandos para execução de cálculos ou processamento de dados, ela possui marcadores, conhecidos como tags, que são utilizados para formatar um texto, uma imagem ou qualquer outro objeto que faça parte da estrutura de um documento HTML (Alves, 2021, p.7).

Segundo Miletto (2014, p.72) “Uma TAG é uma palavra especifica, definida em HTML, envolta por sinais de ‘menor que’ (<) e ‘maior que’ (>). De um modo geral, as TAGs aparecem em pares, uma indicando o início e a outra indicando o fim da marcação”

Miletto (2014) ainda nos dá algumas TAGs HTML conforme Quadro 1.

Quadro 1 - Tag HTML

TAG	Função
html	Marca o início e o fim da página Web, informando ao navegador que o texto contido no documento está escrito em HTML
head	Marca o início e o fim do cabeçalho, a área onde serão descritos os cabeçalhos e o título da página. Pode ser utilizado, ainda, para declarar scripts em Java- script ou definir formatações CSS.
title	Marca o início e o fim do título da página, que sempre está posicionado na barra superior do browser.
body	Marca o início e o fim do corpo da página, que contém imagens, textos, títulos, links etc.

Fonte: Miletto (2014, p.72)

2.5.1 HTML5

De acordo com Turnel (2010, p.29) “Em HTML5 todos os atributos de manipulador de eventos do HTML4, que tomam a forma onEvent-name, são globais. A HTML5 acrescentou novos manipuladores de eventos para novos eventos.”

Continua Turnel (2010, p.33) “Além de novos elementos e atributos, HTML5 não traz diversos atributos de estilização e formatação que são melhor utilizados com a linguagem CSS. Além disso, não traz elementos que comprometem a usabilidade e a acessibilidade.”, Turnel ainda nos traz alguns elementos HTML4 tiveram seu significado ligeiramente modificados em HTML5, conforme Quadro 2.

Quadro 2 - Quadro de modificações das tags HTML

Elemento	Modificação realizada
<a>	Sem um atributo href, representa um espaço reservado para quando uma ligação for feita de forma alternativa. Ele também pode ter conteúdo de fluxo.
<address>	Além de definir as informações de contato do autor ou titular de um documento, agora é delimitado pelo novo conceito de seccionamento.
<b>	Além de negrito, especifica um espaço de texto a ser estilisticamente deslocado do assunto normal sem transmitir importância extra, como palavras-chave em um resumo do documento.
<hr>	Além de criar uma linha horizontal, representa um elemento no nível de ruptura temática, ou seja, define uma mudança no conteúdo.
<i>	Representa, além de itálico, um intervalo de texto com um assunto alternativo ou humor, ou desvio em relação ao assunto normal, como um termo técnico, uma frase idiomática de uma outra língua, um pensamento, ou outros cujo assunto exija a típica apresentação tipográfica em itálico.
<label>	Além de definir o rotulo de um elemento <input>, agora permite receber o foco no elemento <input> quando pressionada a tecla TAB.
	Representa agora importância ao invés de ênfase.
<head>	Não permite mais o elemento <object> em seu interior.
<menu>	Foi redefinido para ser útil em barras de ferramentas e menus de contexto.
<script>	Pode ser usado para inserir scripts ou para delimitar blocos de dados personalizados.

Fonte: Elaborado a partir de Turnel (2010, p.29)

2.6 CSS

Na visão de Miletto, “As folhas de estilo possibilitam criar estilos personalizados para títulos, listas, imagens etc., além de permitirem a definição de cores, fontes, bordas, alinhamentos, entre outras características vinculadas à aparência das páginas Web.” Miletto (2014, p.80)

Segundo a MDN (mdn web docs, 2024) Assim como o HTML, o CSS não é realmente uma linguagem de programação. Também não é uma linguagem de marcação — é uma linguagem de folhas de estilos. Isso significa que o CSS permite aplicar estilos seletivamente a elementos em documentos HTML. Por exemplo, para selecionar todos os elementos parágrafo de uma página HTML e tornar o texto dentro deles vermelho.

Miletto (2014) Apresenta exemplos de regras de CSS que são descritos no Quadro 3.

Quadro 3 - Regras CSS

Exemplo	Descrição
h1 {font-size: 36pt;}	Todos os cabeçalhos de nível 1 usarão fonte de 36 pontos.
h2 {font-size: 24pt; color: blue;}	Todos os cabeçalhos de nível 2 usarão fonte de 24 pontos e a cor azul.

p {font-family: Times; font-size: 12pt; color: blue; margin-left: 0.5in;}	Todos os parágrafos usarão fonte Times, 12 pontos, na cor azul e recuados meia polegada a partir da margem esquerda da página

Fonte: Miletto (2014, p.80)

2.7 JAVASCRIPT

Na visão de (Oliveira, 2020, p.47) “O Javascript é uma linguagem de programação orientada a objetos, sendo interpretada e executada pelo navegador web (*server-side script*). Apresenta uma sintaxe similar à linguagem Java e tem como objetivo principal oferecer melhor interatividade às páginas.”

Continua Oliveira (Oliveira, 2020, p.50) ainda menciona que “As variáveis em Javascript não possuem um tipo de dados no momento de sua declaração. Elas assumem o tipo de dados dos valores que estão armazenados naquele instante.”

Ainda complementa (Miletto, 2014, p.113) “Os eventos são ações realizadas dentro de uma página HTML. Por esse motivo, são vinculados às TAGs HTML. Dependendo da ação, o evento pode, ou não, ser chamado.”

Mileto (2014) Apresenta alguns eventos que são realizados com o mouse conforme Quadro 4.

Quadro 4 - Eventos com mouse

Propriedade	Ocorre quando
onClick	O usuário clica em um elemento.
onDblClick	O usuário clica duas vezes em um elemento.
onMouseDown	O usuário pressiona o botão do mouse sobre um elemento.
onMouseMove	O ponteiro do mouse está em movimento sobre o elemento.
onMouseOver	O ponteiro do mouse está sobre o elemento.
onMouseOut	O usuário move o ponteiro do mouse para fora do elemento.
onMouseUp	O usuário solta o botão do mouse sobre um elemento.

Fonte: Miletto (2014, p.113)

Mileto (2014) Apresenta alguns eventos que são realizados com o teclado conforme Quadro 5.

Quadro 5 – Eventos com teclado

Propriedade	Ocorre quando
onKeyDown	O usuário está pressionando uma tecla.
onKeyPress	O usuário pressiona uma tecla.
onKeyUp	O usuário solta a tecla (previamente pressionada).

Fonte: Mileto (2014, p.113)

2.8 TYPESCRIPT

"O TypeScript é um pré-processador (*superset*) de códigos JavaScript *open source* desenvolvido e mantido pela Microsoft. Ele foi desenvolvido pelo time do Anders Hejlsberg, arquiteto líder do time TypeScript e desenvolvedor do C#, Delphi e do Turbo Pascal." (Adriano, 2022, p.7)

"O compilador do TypeScript é altamente configurável. Ele nos permite definir o local onde estão os arquivos .ts dentro do nosso projeto, o diretório de destino dos arquivos transpilados, a versão ECMAScript que será utilizada, o nível de restrição do verificador de tipos e até se o compilador deve permitir arquivos JavaScript." (Adriano, 2022, p.10)

A documentação do Typescript ainda nos traz alguns exemplos interessantes sobre Javascript e Typescript:

Typescript (2024) O operador de igualdade do JavaScript (==) converte seus argumentos, levando a comportamento inesperado.

Conforme documentação do Typescript a "Detecção de erros sem execução do código é chamada de verificação estática. Determinar o que é um erro e o que não é baseado nos tipos dos valores sendo operados é chamado de verificação estática de tipos." (Typescript, 2024).

"TypeScript verifica um programa por erros antes de sua execução e faz isso baseado nos tipos dos valores, é um verificador de tipo estático. Por exemplo, o último exemplo acima tem um erro por causa do tipo de obj. Aqui está o erro que o TypeScript encontra:" (Typescript, 2024).

2.8.1 ZOD

Segunda a documentação do Zod(2024), Zod é uma biblioteca de validação e declaração de esquema do TypeScript. Estou usando o termo "esquema" para me referir amplamente a qualquer tipo de dados, de uma simples string a um complexo objeto aninhado. O Zod foi projetado para ser o mais amigável possível ao desenvolvedor. O objetivo é eliminar declarações de tipo duplicadas. Com Zod, você declara um validador *uma vez* e Zod inferirá automaticamente o tipo TypeScript estático. É fácil compor tipos mais simples em estruturas de dados complexas.

2.9 REACT

Os aplicativos React são feitos de componentes. Um componente é uma parte da interface do usuário (interface do usuário) que tem sua própria lógica e aparência. Um componente pode ser tão pequeno quanto um botão ou tão grande quanto uma página inteira. (React, 2024)

Os nomes dos componentes React devem sempre começar com uma letra maiúscula, enquanto as marcas HTML devem ser minúsculas. (React, 2024)

Os componentes do React recebem dados e retornam o que deve aparecer na tela. Você pode passar novos dados para eles em resposta a uma interação, como quando o usuário digita em uma entrada. O React atualizará a tela para corresponder aos novos dados. (React,2024).

2.9.1 React Hook Form

Conforme um artigo publicado no Medium (2023), React Hook Form (RHF) é uma biblioteca poderosa e flexível para criar formulários em React. Ele fornece uma maneira simples e eficiente de lidar com a entrada e validação de formulários, facilitando a criação de formulários dinâmicos e responsivos à entrada do usuário. A validação de formulários é um aspecto essencial da criação de formulários robustos e confiáveis. Ele garante que os dados enviados pelo usuário sejam precisos, completos e no formato correto. Sem validação, os formulários podem ser vulneráveis a erros, problemas de segurança e frustração do usuário. O React Hook Form facilita a adição de validação aos seus formulários usando uma variedade de funções de validação integradas e funções de validação personalizadas.

2.10 NEXT.JS 14

Conforme a documentação do Next.js (2024a) nos informa, o Next.js é uma estrutura React para a criação de aplicativos *Web full-stack*. Use o React *Components* para criar interfaces de usuário e Next.js para recursos e otimizações adicionais. Internamente o Next.js também abstrai e configura automaticamente as ferramentas necessárias para o React, como agregação, compilação e muito mais. Isso permite que você se concentre na criação de seu aplicativo em vez de gastar tempo com a configuração.

Next.js ainda nos traz algumas das suas principais características conforme Quadro 6.

Quadro 6 - Principais características

Característica	Descrição

Roteamento	Um roteador baseado em sistema de arquivos criado sobre componentes de servidor que oferece suporte a layouts, roteamento aninhado, estados de carregamento, tratamento de erros e muito mais.
Renderização	Renderização do lado do cliente e do lado do servidor com componentes de cliente e servidor. Otimizado ainda mais com renderização estática e dinâmica no servidor com Next.js. Streaming no Edge e Node.js *runtimes*.
Busca de dados	Busca de dados simplificada com assíncrono/*await* em Componentes de Servidor e uma API estendida para memorização de solicitação, cache de dados e revalidação.
Otimizações	Otimizações de imagem, fontes e scripts para melhorar os principais sinais vitais da Web e a experiência do usuário do aplicativo.

Fonte: Montado a partir da documentação do (Next.js, 2024ª).

Next.js melhora o desempenho do aplicativo e reduz os custos armazenando em cache o trabalho de renderização e as solicitações de dados. (Next.js, 2024b)

A mesma documentação, afirma por padrão que Next.js armazenará em cache o máximo possível para melhorar o desempenho e reduzir os custos. Isso significa que as rotas são renderizadas estaticamente e as solicitações de dados são armazenadas em cache, a menos que você opte por não participar. (Next.js, 2024b)

O cache dura o tempo de vida de uma solicitação do servidor até que a árvore de componentes do React tenha concluído a renderização. Como a memorização não é compartilhada entre solicitações do servidor e só se aplica durante a renderização, não há necessidade de revalidá-la. (Next.js, 2024b)

Para revalidar dados em um intervalo cronometrado, você pode usar a opção de definir o tempo de vida do cache de um recurso (em segundos), (Next.js, 2024b).

2.11 TAILWIND CSS

O Tailwind CSS funciona verificando todos os seus arquivos HTML, componentes Javascript e quaisquer outros modelos em busca de nomes de classe, gerando os estilos correspondentes e gravando-os em um arquivo CSS estático. (TAILWINDCSS, 2024)

Essa abordagem nos permite implementar um design de componente completamente personalizado sem escrever uma única linha de CSS personalizado. (TAILWINDCSS, 2024)

Seu CSS para de crescer. Usando uma abordagem tradicional, seus arquivos CSS ficam maiores cada vez que você adiciona um novo recurso. Com os utilitários, tudo é reutilizável, então você raramente precisa escrever um novo CSS. (TAILWINDCSS, 2024)

Cada classe de utilitário em Tailwind pode ser aplicada condicionalmente adicionando um modificador ao início do nome da classe que descreve a condição que você deseja segmentar. Por exemplo, para aplicar a classe ao focalizar, use a classe:bg-sky-700 hover:bg-sky-700.

2.12 JSX

Conforme nos explica Marcolino (2021) “O JSX não é uma marcação HTML nem de uma *string*; ele é uma extensão de sintaxe para o JavaScript. É basicamente uma linguagem para a definição da interface do usuário (UI), mas atrelada a funcionalidades e dinamicidades do JavaScript. “ (Marcolino, 2021, p.67)

Aqui o autor nos dá mais alguns detalhes sobre o JSX, “Como as expressões JSX são convertidas em funções JavaScript, podemos utilizar o JSX dentro de blocos if e for, bem como atribuir tais valores em variáveis e utilizá-las como funções”(Marcolino, 2021, p.70).

Em questão de segurança, “O JSX também garante a segurança, prevenindo injeção de conteúdos indesejados e ataques do tipo Cross-site-scripting (XSS), já que tudo é verificado pelo Modelo de Documento de Objetos React (DOM React). Ao final, tudo é convertido em cadeia de caracteres (strings).” (Marcolino, 2021, p.71).

> O XSS é uma vulnerabilidade de aplicações web que permite que código JavaScript sejam injetados, em especial em páginas comuns aos usuários, como páginas de log-in ou iniciais de um site. O ataque ocorre por meio de formulários alterados por scripts JavaScript de modo a redirecionar dados dos usuários para os atacantes. A fim de evitar tais ataques, como os presentes no JSX, é necessário controlar e tratar os dados de entrada e saída tanto de usuários quanto de aplicações. (Marcolino, 2021, p.71)

E ainda continua com (Marcolino, 2021, p.71) “Uma última característica importante do JSX é a capacidade do Babel de compilar objetos representados por meio do React.createElement(). Os trechos de código a seguir, por exemplo, resultam na mesma apresentação”.

“O método createElement() cria e retorna um novo elemento React de tipo específico. No caso do primeiro trecho de código, por utilizar o JSX, o método React.createElement() não é exibido, mas este é chamado para transformar o trecho em um elemento React. Já no seguinte trecho de código, escrito sem JSX, o método é chamado diretamente.” (Marcolino, 2021, p.72).

2.13 REACT NATIVE

Conforme nos explica a documentação (React Native, 2024), ele é uma estrutura de código aberto para criar aplicativos Android e IOS usando o React e os recursos nativos da plataforma de aplicativos. Com o React Native, utilizando-se Javascript para acessar as apis da plataforma, bem como para descrever a aparência e o comportamento da interface do usuário usando componentes do React: pacotes de código reutilizável e encaixável.

A documentação ainda nos traz sobre como funciona a exibição, no desenvolvimento Android e IOS, uma exibição é o bloco de construção básico da interface do usuário: um pequeno elemento retangular na tela que pode ser usado para exibir texto, imagens ou responder à entrada do usuário. mesmo os menores elementos visuais de um aplicativo, como uma linha de texto ou um botão, são tipos de exibições.

Conforme descrito na documentação oficial foi elaborado o Quadro 7 com os principais componentes do React Native.

Quadro 7 – Principais componentes React-Native

Componente de interface do usuário do React Native	**visualização do Android**	**visualização do IOS**	**Descrição**
<View>	<ViewGroup>	<UIView>	Um contêiner que dá suporte ao layout com flexbox, estilo, manipulação de alguns toques e controles de acessibilidade
<Text>	<TextView>	<UITextView>	Exibe, estiliza e aninha cadeias de caracteres de texto e até manipula eventos de toque
<Image>	<ImageView>	<UIImageView>	Exibe diferentes tipos de imagens
<ScrollView>	<ScrollView>	<UIScrollView>	Um contêiner de rolagem genérico que pode conter vários componentes e exibições
<TextInput>	<EditText>	<UITextField>	Permite que o usuário insira texto

Fonte: Elaborado a partir da documentação do (React-Native, 2024).

2.13.1 Expo

Conforme a documentação nos apresenta (EXPO, 2024a), o Expo é um projeto de código aberto que oferece aos desenvolvedores ferramentas poderosas para auxiliar na criação e manutenção de aplicativos React Native em qualquer escala. Por exemplo, pacotes Expo

CLI, Expo Router e Expo SDK. Todas as ferramentas de código aberto da Expo são totalmente gratuitas e possuem a licença do MIT.

O Expo Application Services (EAS) é um conjunto de serviços hospedados que você pode usar com projetos Expo e React Native para:

- Criar, enviar e atualizar seu aplicativo;
- Configure a automação em torno de todos esses processos;
- Colabore com sua equipe;

O EAS resolve um conjunto de problemas que exigem recursos físicos, como servidores de aplicativos e CDNs para fornecer atualizações *over-the-air* e servidores físicos para executar compilações. A EAS tem um plano gratuito generoso que funcionará para muitos projetos de estudantes e hobby.

2.13.1.1 Expo SecureStore

Consultado a documentação do Expo SecureStore tem-se algumas informações de como ela funciona (EXPO, 2024b), é biblioteca que fornece uma maneira de criptografar e armazenar com segurança pares de chave-valor localmente no dispositivo. Cada projeto da Expo tem um sistema de armazenamento separado e não tem acesso ao armazenamento de outros projetos da Expo.

No Android, os valores são armazenados em SharePreference, criptografado com o sistema Keystore do Android, no IOS os valores são armazenados usando os serviços de chaves como *kSecClassGenericPassword*. O iOS tem a opção adicional de poder definir o atributo do valor *kSecAttrAccessible*, que controla quando o valor está disponível para ser buscado.

2.14 GIT

Segundo Ferreira (2021, p.37) “O Git é a principal ferramenta de controle de versão do grupo open source, porque, além de ser uma ferramenta completa, com estrutura e funcionalidades essenciais para realizar o controle de versões de um sistema, é uma ferramenta construída por Linus Torvalds, o criador do Linux.”.

> É importante ressaltar que o Git é uma ferramenta de controle de versão que não se limita somente ao desenvolvimento de software, é utilizado, por exemplo, como um sistema de controle de versões de livro ou qualquer outro arquivo de texto, além de

> realizar controle de arquivos de configurações de servidores, por exemplo. Isso demonstra a grande dinamicidade e o poder que está ferramenta possui. (Ferreira, 2021, p.37).

O autor ainda complementa Ferreira (2021, p.41) “Com o Git e o GitHub, é possível realizar o congelamento de versões, levando ao controle das mudanças realizadas no desenvolvimento de determinado software. Sendo assim, fica fácil de observar todo o projeto, desde o início de seu desenvolvimento até o momento que mudanças são realizadas.”.

Prosseguindo com a explicação de Ferreira (2021, p.41) “Assim, a partir do repositório remoto, é que gestores ou a equipe da qualidade podem se certificar dos itens que compõe determinada versão ou release para testes ou para *deployment*.”

2.15 VERCEL

De acordo com a documentação de software, a Vercel é uma plataforma para desenvolvedores que fornece as ferramentas, os fluxos de trabalho e a infraestrutura necessária para criar e implantar os aplicativos Web mais rapidamente, sem a necessidade de configuração adicional. (Vercel, 2024).

A documentação do software Vercel ainda oferece suporte a estruturas de front-end populares prontas para uso, e sua infraestrutura escalável e segura é distribuída globalmente para fornecer conteúdo de data centers próximos aos usuários para velocidades ideais.

Durante o desenvolvimento, a Vercel fornece ferramentas para colaboração em tempo real em seus projetos, como visualização automática e ambientes de produção, e comentários sobre implantações de visualização.

2.16 VSCODE

Conforme a documentação do VSCODE (Visual Studio Code, 2024), Visual Studio Code é um editor de código-fonte leve, mas poderoso, que é executado em sua área de trabalho e está disponível para Windows, macOS e Linux. Ele vem com suporte embutido para JavaScript, TypeScript e Node.js e tem um rico ecossistema de extensões para outras linguagens e tempos de execução (como C++, C#, Java, Python, PHP, Go, .NET).

A documentação ainda acrescenta (Visual Studio Code, 2024) IntelliSense é um termo geral para vários recursos de edição de código, incluindo: conclusão de código, informações de parâmetro, informações rápidas e listas de membros. Os recursos do IntelliSense às vezes são

chamados por outros nomes, como "conclusão de código", "assistência de conteúdo" e "dica de código".

2.17 SWAGGER

Conforme nos fala a documentação de software o Swagger é um conjunto poderoso, e fácil de usar, de ferramentas de desenvolvimento de API para equipes e indivíduos, permitindo o desenvolvimento em todo o ciclo de vida da API, desde o design e a documentação até o teste e a implantação. (Swagger, 2024).

O Quadro 8 nos dá alguns objetivos dos Swagger.

Quadro 8 - Objetivos do Swagger

Objetivo	Descrição
Livre de dependência	A interface do usuário funciona em qualquer ambiente de desenvolvimento, seja localmente ou na Web
Amigável para humanos	permita que os desenvolvedores finais interajam sem esforço e experimentem cada operação que sua API expõe para facilitar o consumo
Fácil de navegar	Encontre e trabalhe rapidamente com recursos e *endpoints* com documentação categorizada
Estilo totalmente personalizável	Ajuste sua interface do usuário do Swagger da maneira que você quiser com acesso total ao código-fonte

Fonte: Desenvolvido a partir da documentação do (Swagger, 2024)

A documentação ainda nos traz outras informações (Swagger, 2024) o Swagger consiste em uma mistura de ferramentas de código aberto, gratuitas e disponíveis comercialmente, é construído pela SmartBear Software, líder em ferramentas de qualidade de software para equipes. A SmartBear está por trás de alguns dos maiores nomes no espaço de software, incluindo Swagger, SoapUI e QAComplete.

2.18 FIGMA

Conforme nos é explicado na documentação do Figma(2024). Com a prototipagem no Figma, você pode criar vários fluxos para seu protótipo em uma página para visualizar a jornada completa e a experiência de um usuário por meio de seus designs. Um fluxo é a rede de quadros e conexões em uma única página. Um protótipo pode mapear toda a jornada de um usuário pelo seu aplicativo ou site, ou pode se concentrar em um segmento específico dele por meio de seu próprio fluxo. Por exemplo: seu protótipo cobre todas as interações possíveis em um site de

comércio eletrônico. Dentro do protótipo, você tem fluxos para criar uma conta, adicionar itens a um carrinho e finalizar a compra.

3. METODOLOGIA DA PESQUISA

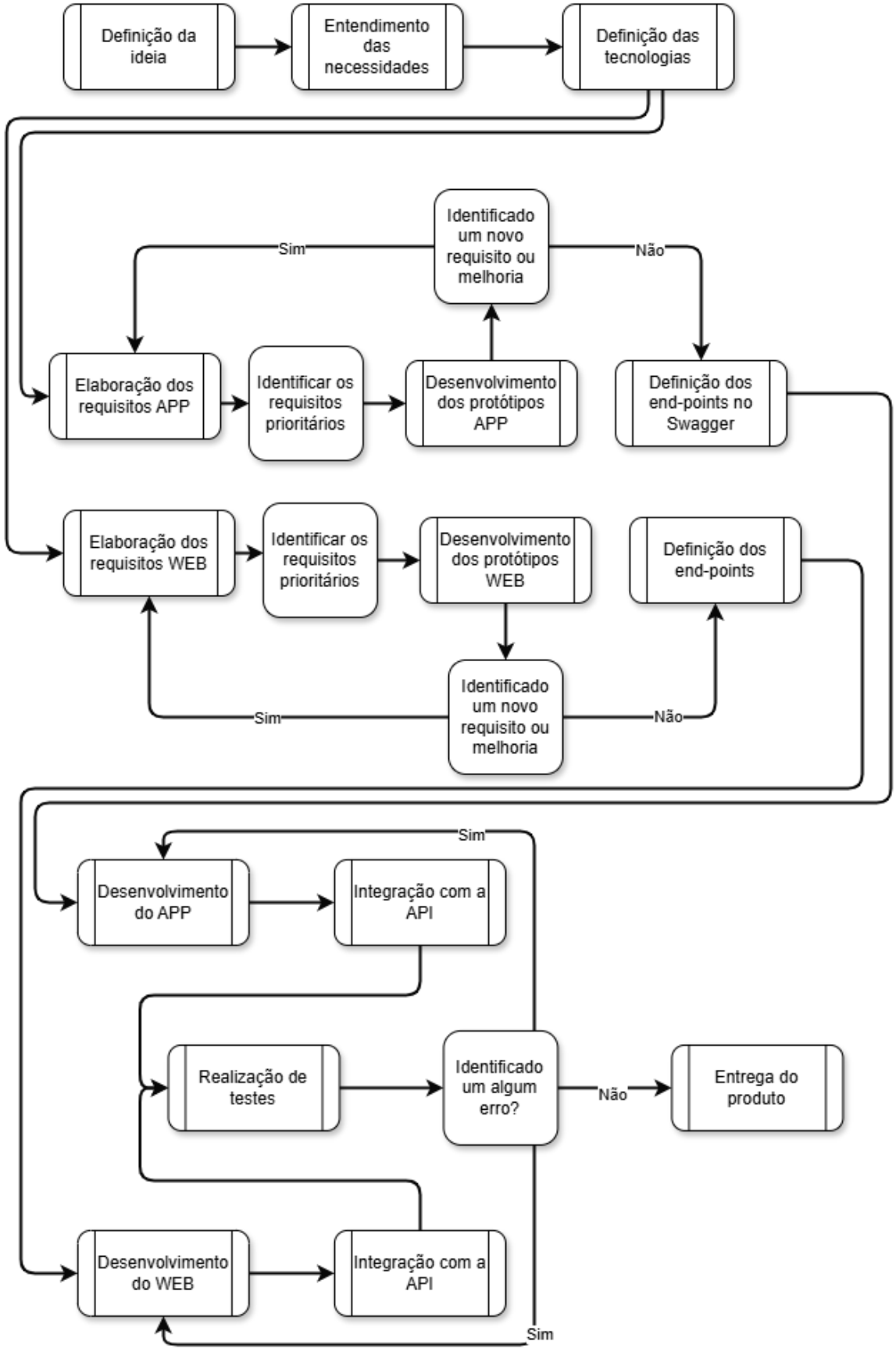
O presente trabalho de conclusão de curso caracteriza-se como pesquisa descritiva e aplicada, o objetivo é analisar e desenvolver uma interface para os usuários de uma academia. O trabalho buscou solucionar os seguintes problemas: Como gerenciar e facilitar a criação de treinos para usuários de uma academia? Como ajudar o aluno a visualizar os treinos a serem realizados? Qual tecnologia supriria as necessidades para o desenvolvimento destas interfaces?

Quanto aos procedimentos realizados, utilizou-se de levantamento documental e de pesquisa de campo qualitativa. A pesquisa foi operacionalizada da seguinte maneira: foram realizadas conversas com a professora responsável pela academia, que ajudou demonstrando como é feito hoje para realizar a prescrição de treino de um aluno e como é feita suas avaliações.

De acordo com o fluxograma apresentado na Figura 1, foram realizadas atividades para entender os princípios e estruturas que precisam ser atingidos para o desenvolvimento do protótipo. Fora realizada a criação de dois protótipos um para o APP e um para o sistema WEB. O APP tem com base de seu desenvolvimento o React Native em conjunto com o EXPO e o WEB o React com Framework NEXT.JS se comunicando com um back-end por requisições HTTPS com transporte de dados no formato JSON, ambos desenvolvimentos usando o Typescript para garantir maior segurança e robustez dos dados.

A escolha de desenvolver duas aplicações distintas se dá pelas necessidades e comportamentos diferentes dos públicos-alvo: professores e alunos. A aplicação WEB será focada em professores, proporcionando uma interface rica e detalhada mais adequada para a prescrição de treinos e acompanhamento dos alunos. Em contraste, o APP móvel será direcionado a alunos, oferecendo uma experiência otimizada para uso em movimento e acesso rápido, com possibilidade de uso offline e de recursos do próprio dispositivo.

Figura 1 – Fluxograma de Elaboração do Trabalho



Fonte: Acervo do autor.

3.1 API DE INTEGRAÇÃO DE SISTEMA WEB PARA ACADEMIA

Este trabalho foi desenvolvido em conjunto com o Trabalho de Conclusão de Curso realizado pelo acadêmico Welson Douglas Gomes de Oliveira, "API DE INTEGRAÇÃO DE SISTEMA WEB PARA ACADEMIA". O projeto visa consumir os endpoints da API, utilizando o formato REST seguindo as boas práticas da arquitetura RESTFULL. A API foi projetada com endpoints específicos, como o cadastro de novos usuários, utilizando os verbos HTTP adequados. Quando um cadastro é bem-sucedido, a API deverá retornar um código de status HTTP 201, indicando que o recurso foi criado com sucesso, assim seguindo as boas práticas imposta por APIs RESTFULL.

3.2 ESTADO DA ARTE

Nesta seção iremos abordar algumas das fermentas que já estão disponíveis no mercado, oferecendo serviços similares ao que o protótipo apresenta, para efeitos de comparação vamos fazer a comparação com as principais características entre NextFit, e Sistema SCA

3.2.1 NextFit

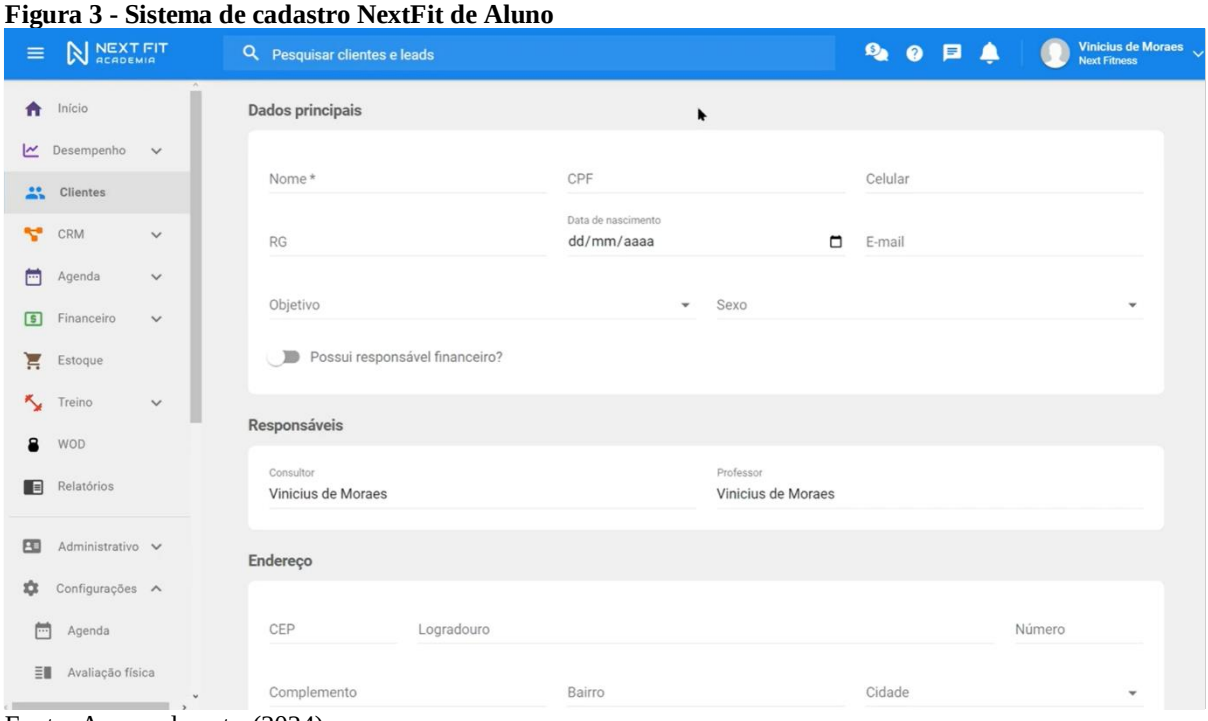
O Next Fit é um sistema de gestão para academias, estúdios e boxes que facilita o controle de atividades como agendamento de aulas, gerenciamento financeiro, comunicação automatizada com alunos e avaliação física. A plataforma oferece módulos para prescrição de treinos, gestão de planos e integração com aplicativos para alunos e instrutores. É um dos sistemas mais populares no Brasil, com suporte em nuvem e funcionalidades para facilitar as atividades diárias.

A seguir lista-se algumas imagens do sistema NextFit. Na , tem-se um exemplo de tela em que o aluno visualiza os treinos. Na Figura 3, tem-se um exemplo de tela de cadastro de novos alunos.

Figura 2 - Exemplo App NextFit

Fonte: App Store (2024).

Figura 3 - Sistema de cadastro NextFit de Aluno



Fonte: Acervo do autor(2024).

3.2.2 Sistema SCA

O Sistema SCA é um software de gestão para academias, clubes e estúdios, desenvolvido pela ProSistemas. O sistema oferece controle financeiro, gestão de acesso, avaliação física, fichas de treinamento e reconhecimento facial. O sistema está disponível em versões online e desktop, oferecendo uma solução completa para a modernização e otimização da gestão de negócios fitness, adaptando-se às necessidades específicas de cada estabelecimento. Possui uma vitrine online aonde as academias podem expor seus produtos para os alunos.

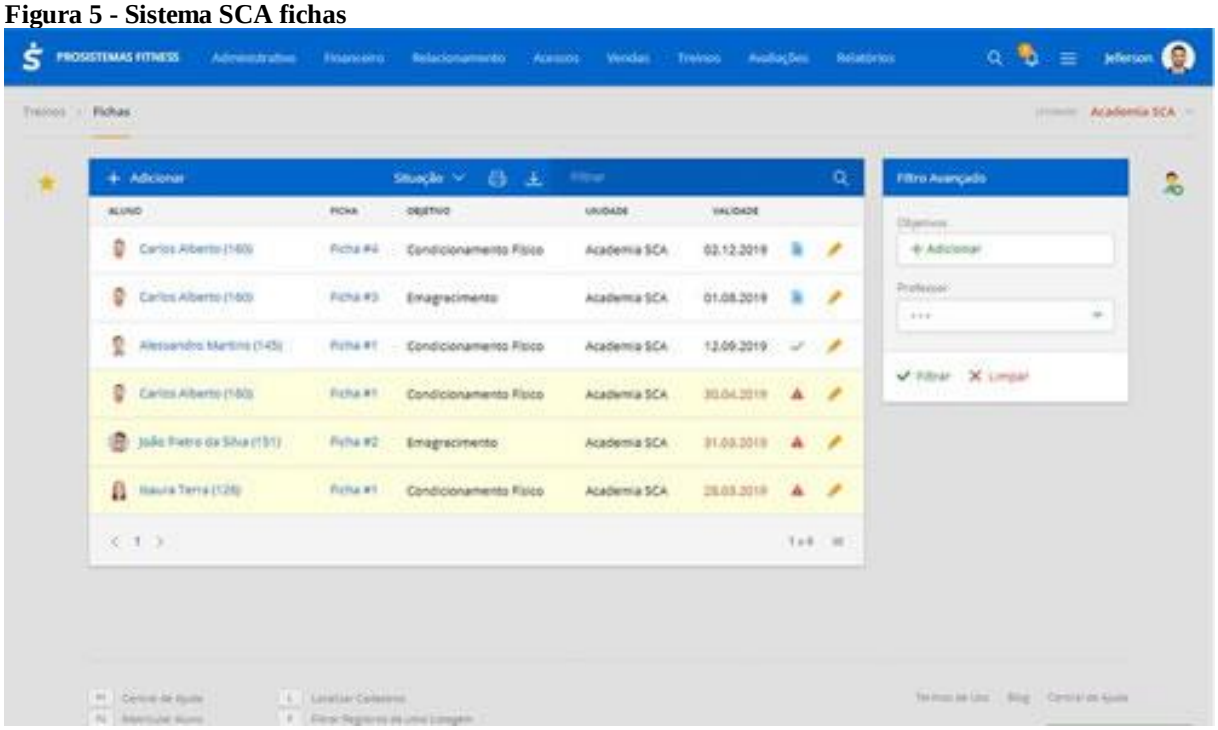
Conforme foi listado anteriormente para o sistema NextFit, disponibiliza-se algumas imagens do Sistema SCA. Na Figura 4 , tem-se uma imagem da tela de menu, onde o usuário é direcionado para as demais telas.

Figura 4 - Sistema SCA APP

Fonte: App Store(2024)

Na Figura 5, tem-se como é uma tela de ficha de treinos do sistema SCA.

Figura 5 - Sistema SCA fichas



Fonte: sistemasca(2024)

3.2.3 Quadro Comparativo

A seguir demonstra-se no Quadro 9, um comparativo com os principais pontos deste trabalho em comparação com as demais aplicações, NextFit e Sistema SCA.

Quadro 9 - Comparativo sistemas

	TC	NextFit	Sistema SCA
Possui aplicativo	X	X	X
Listagem de treinos	X	X	X
Avaliação física	X	X	X
Confirmação da presença por localização	X		
Sistema desktop			X
Sistema web	X	X	X
Integração com catracas		X	X
Controle financeiro		X	X

Fonte: Acervo do autor.

O sistema proposto se diferencia das demais soluções, como o NextFit e o Sistema SCA, ao oferecer funcionalidades específicas focadas diretamente na experiência dos usuários. Enquanto outras plataformas apresentam recursos amplos, como controle financeiro e integração com catracas, o sistema proposto prioriza a personalização e o acompanhamento detalhado de treinos, incluindo confirmação de presença por localização e acesso a informações visuais interativas sobre os exercícios. A utilização de uma plataforma exclusivamente baseada na web e em aplicativos móveis proporciona maior praticidade e acessibilidade.

4. INTERFACE DE USUÁRIO PARA UM SISTEMA DE PRESCRIÇÃO E AVALIAÇÃO DE TREINOS EM ACADEMIAS

Esta sessão foi organizada em duas partes sendo uma Web e outra App Mobile, organizando em cada uma os elementos de desenvolvimento conforme escopo, sendo documentado em separado uma breve descrição de cada parte, lista de requisitos funcionais e protótipos visuais.

As regras de negócio e os requisitos não funcionais, foram agrupados nesta sessão principal.

4.1 REGRAS DE NEGÓCIO

As regras de negócio são responsáveis por indicar quais serão as normas do negócio que deverão ser implementadas, para que a aplicação funcione da maneira correta. No Quadro 10 serão listadas algumas das principais regras de negócio para o desenvolvimento da interface do sistema web.

Quadro 10 - Regras de negócio

Número	Descrição
RN01	Novos professores poderão ser cadastrados apenas por pessoa responsável com poderes para tal.
RN02	Para operacionalizar a aplicação deverá o usuário identificar-se sempre.
RN03	Qualquer um dos professores treinador poderá incluir novos alunos.
RN04	Todo aluno cadastrado deve ter obrigatoriamente ter um professor vinculado a ele.
RN05	Um professor poderá determinar qualquer quantidade e tipo de exercício para os alunos, não há restrições, incluindo a possibilidade de agrupar exercícios.
RN06	O professor deve poder agrupar algum exercício caso ele queira.
RN07	Os professores / instrutores preparam para seus alunos um conjunto de atividades para que eles executem. Essas atividades são distribuídas ao longo de um período com uma meta de treino, como por exemplo perder peso, ganhar massa etc.
RN08	Dispor de opção para remover um ou múltiplos exercícios adicionado.
RN09	Todo treino criado deve ser informado um aluno que vai consumir este treino.
RN10	Treinos não podem ser cadastrados sem data de início e fim.
RN11	Organizar os treinos dos alunos na mesma sequência em que o professor passou.
RN12	Listar todos os treinos dos alunos, caso o aluno não tenha finalizado o treino da semana anterior deve continuar com o treino da semana anterior e não recomeçar um novo na próxima semana.
RN13	Permitir ao aluno marcar o treino como finalizado sem ter que passar por todas as series.
RN14	Não deve permitir realizar o treino C antes de realizar o treino A.
RN15	Os alunos devem conseguir marcar o peso que ele está utilizando naquele exercício.
RN16	Os alunos devem ter acesso a informações detalhadas sobre os exercícios, incluindo imagens ou vídeos, para assegurar a execução correta das atividades.
RN17	O aluno só recebera presença se estiver dentro do ambiente da academia.

Fonte: Acervo do autor.

4.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais são critérios que descrevem características e atributos que não estão diretamente relacionados às funcionalidades específicas da aplicação, mas que são

igualmente importantes a sua qualidade. No Quadro 11 serão listados os requisitos não funcionais.

Quadro 11 - Requisitos não funcionais

Número	Requisito	Descrição
RNF01	Padrão de cores	Deverá usar as paletas de cores “blue”, “Green”, “Red” e “White” do Tailwind.css para todo o sistema.
RNF02	Curva de aprendizagem	Deverá ser de fácil navegação, intuitivo e de fácil interpretação.
RNF03	Manutenibilidade	Sistema deve ser de fácil manutenção, utilizando-se de princípios de codificação e desacoplamento conhecidos como SOLID, Design Patterns.
RNF04	Responsividade	Deverá ser disponibilizado para todos as resoluções de dispositivos, notebooks, desktops ou tablets, sendo excluídos dispositivos moveis como celulares.
RNF05	Desacoplado	Deverá ser desacoplado de qualquer ferramenta específica de alguma nuvem seja (AWS, AZURE ou GOOGLE CLOUD)
RNF06	REST full	Deverá ser completamente desacoplado do back-end recebendo e enviando apenas os dados independentemente de como foi implementado o back-end
RNF07	Acessibilidade	Sistema deve ser acessível através de qualquer computador conectado à internet.
RNF08	Usabilidade	Sistema deve ter uma boa usabilidade para o usuário final e objetivo nas suas tarefas.
RNF09	Armazenamento de dados	Todos os dados armazenados no APP devem ser armazenados de forma segura. Deverá utilizar armazenamento com a biblioteca SecureStore, quando o usuário acessar o app deve verificar se houve mudanças no treino caso não houver não precisa solicitar novamente os dados de treinos.

Fonte: Acervo do autor.

4.3 REQUISITOS FUNCIONAIS

Os requisitos funcionais são declarações detalhadas das funcionalidades e o comportamento específico que o sistema deverá ter para atender às necessidades propostas. Esses requisitos definem as ações, operações e tarefas que o protótipo web deve ser capaz de executar.

No Quadro 12 são descritos os requisitos funcionais para a interface do sistema web.

Quadro 12 - Requisitos Funcionais da Interface Web

Número	Nome	Descrição	RN
RF01	Cadastro de aluno	Sistema deve dispor de uma tela de cadastro de aluno onde deve ser informado, nome, data de nascimento, idade, sexo, CPF, endereço, complemento, bairro, cidade, UF, cep, telefone, celular, e-mail, nível do aluno, nível de atividade, situação, objetivo, observações. Quando for informado o CEP no cadastro sistema deve automaticamente preencher as demais informações de endereço, deixando apenas os campos complemento e número sem preencher. Automaticamente ao preencher o cadastro do aluno o professor se torna o responsável pelos treinos deste aluno.	RN03, RN04

RF02	Cadastro de professor	Sistema deve dispor de uma tela de cadastro de professor onde apenas o administrador do sistema poderá cadastrar um novo professor, deve ser informado, nome, CPF, data de nascimento, idade, sexo, endereço, complemento, bairro, cidade, UF, CEP, telefone, celular, e-mail, situação. Quando for informado o CEP no cadastro sistema deve automaticamente preencher as demais informações de endereço, deixando apenas os campos complemento e número sem preencher.	RN01
RF05	Cadastro de treino	Sistema deve dispor de uma tela para cadastro de treino, onde deve ser informado, nome ou código do aluno, data de início, duração do treino, descrição do treino, adicionar exercício, séries, número de repetições, tempo de descanso, peso, tipo treino A, B... E. Os exercícios podem ser agrupados, selecionando mais de um e clicando na opção de agrupar. Caso o professor tenha adicionado algum exercício de forma errônea tem que haver uma opção de remover o exercício.	RN05, RN06, RN07, RN08, RN09, RN10
RF06	Login	Sistema deve dispor de uma tela de login, onde o usuário poderá fazer login com código ou e-mail e senha.	RN02

Fonte: Acervo do autor.

No Quadro 14 tem-se os requisitos para o desenvolvimento da interface do aplicativo.

Quadro 13 - Requisitos de interface do Aplicativo

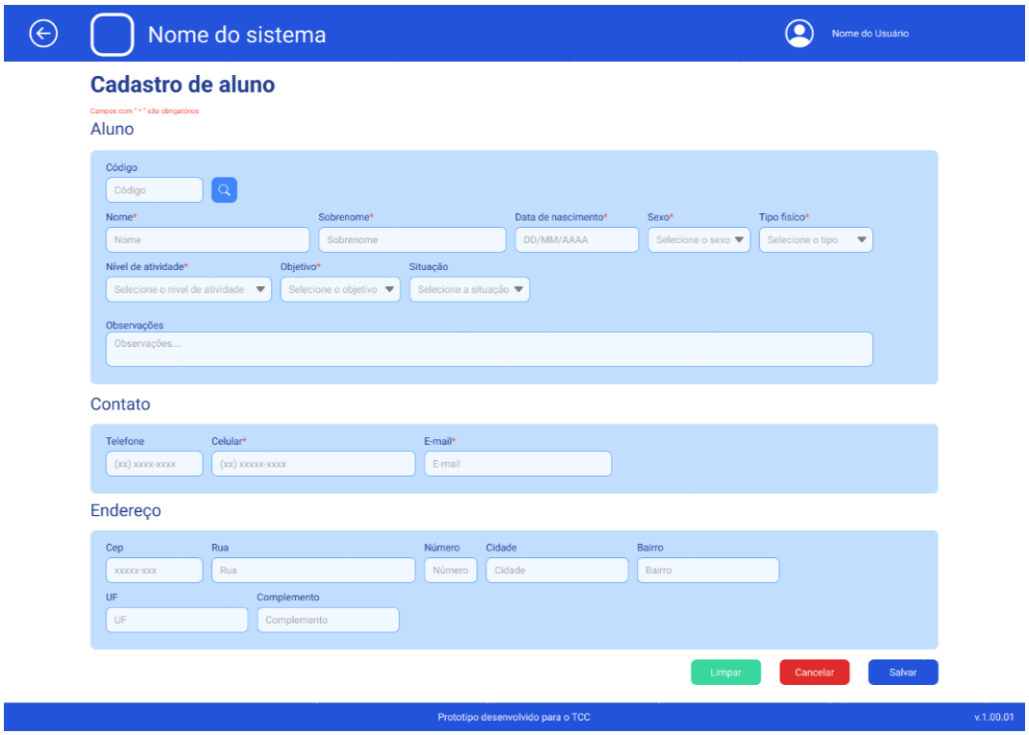
Número	**Nome**	**Descrição**	**RN**
RF01	Login	Deve dispor de uma tela de login por código do aluno ou e-mail e senha.	RN02
RF03	Visualizar treinos da semana	Deve apresentar uma tela com as letras dos treinos que o aluno vai realizar alguma atividade, para que ele possa visualizar o treino daquela letra selecionada.	RN13
RF04	Visualizar treino do dia	Conter uma tela onde será possível visualizar o treino do dia, mas não podendo realizar um treino de um dia selecionado a não ser que seja o dia de realização dele.	RN12, NR14
RF05	Card do exercício	Deve mostrar em formato de card o exercício que o aluno vai executar, onde deve conter o nome do exercício, uma imagem do exercício (de preferência que seja um gif), número de repetições, número de séries, e tempo de descanso, um local para o aluno ter mais informações sobre o exercício, e poder informar o peso utilizado naquele exercício.	RN15, RN16
RF06	Informações sobre o exercício	Deve mostrar mais informações sobre o exercício, como a carga já utilizada anteriormente, permitir alterar o peso utilizado, descrição de como executar o exercício, informações de qual grupo muscular o exercício vai afetar.	RN16
RF07	Conclusão de exercício	Deve permitir ao usuário que marque o exercício como feito.	RN13
RF08	Conclusão de treino	Deve permitir ao usuário que ao final do treino ele marque o treino do dia como concluído, mesmo se ele não terminar todos os exercícios, ao finalizar o treino não será possível marcar nenhum exercício do dia mais como concluído, com a conclusão do treino concluída o aluno recebe automaticamente a presença do dia.	RN13
RF10	Localização do aluno	Deve permitir concluir e marcar como realizado algum exercício se o aluno estiver dentro da instituição, não sendo permitido realizar conclusão de exercício nem de treino fora da instituição. Deverá ser possível localizar o dispositivo através do nome do wifi ou através da localização via GPS, sendo considerado um raio de no máximo 50m fora da academia.	RN09, RN17

Fonte: Acervo do autor.

4.4 PROTOTIPAÇÃO

Para facilitar o desenvolvimento das interfaces do sistema foi realizada a prototipação da tela principal utilizando a ferramenta Figma. No protótipo visual foi aplicada a paleta de cores do design previsto nos requisitos não funcionais. As Figura 6, Figura 7 e Figura 8 representam os protótipos das telas de Cadastro de aluno, Home e Listagem de exercícios do sistema WEB.

Figura 6 - Protótipo interface WEB – Cadastro de aluno



Fonte: Acervo do autor.

Na Figura 07 temo o exemplo da interface web da tela de home, aonde o usuário terá acesso após realizar o login no sistema.

Figura 7 - Protótipo interface WEB - Home

Fonte: Acervo do autor.

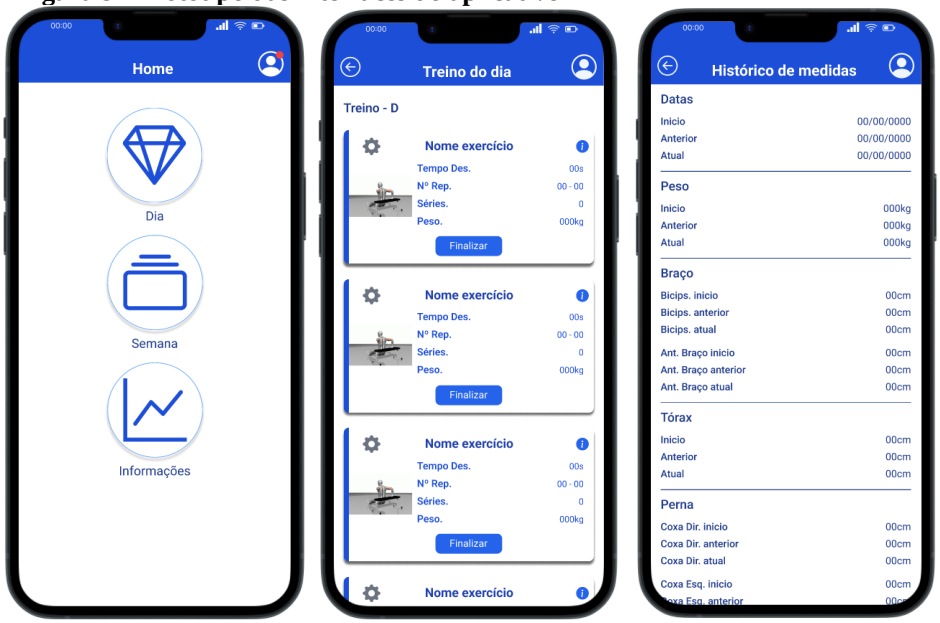
As funcionalidades Cadastro de exercício, Listagem de Equipamentos, Cadastro e listagem de medidas e Desempenho não estão contempladas na entrega do presente trabalho de conclusão de curso.

Figura 8 - Protótipo interface WEB – Cadastro de medida

Fonte: Acervo do autor.

Na Figura 9 são demonstrados alguns protótipos das telas desenvolvidos exclusivamente para o aplicativo mobile.

Figura 9 - Protótipo das interfaces do aplicativo

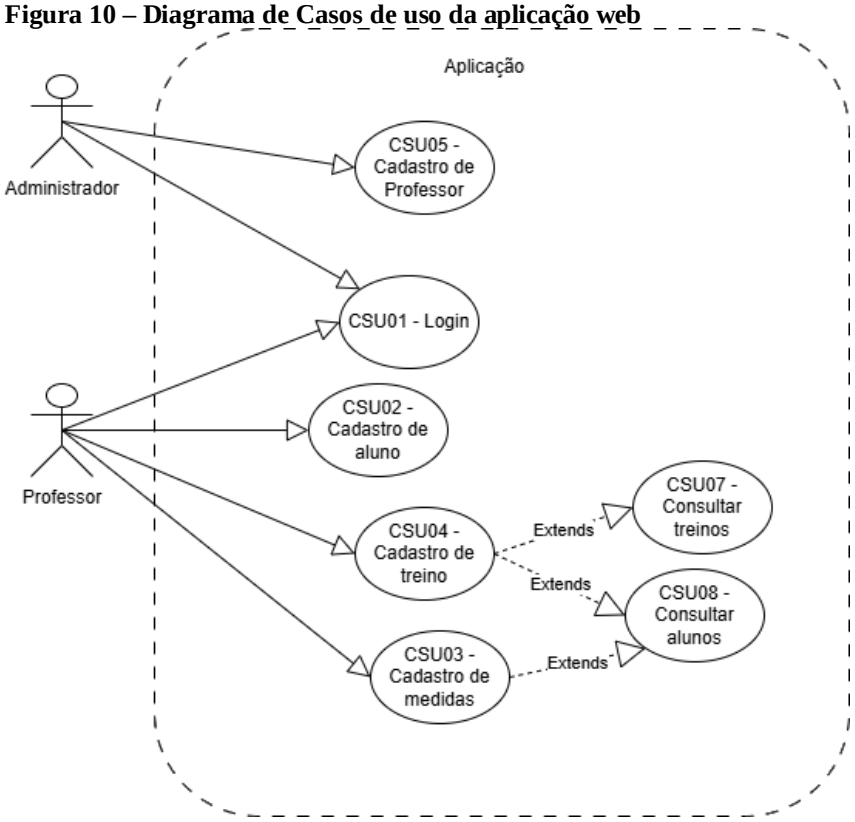


Fonte: Acervo do autor.

4.5 DIAGRAMAS

Na Figura 10 temos o caso de uso da aplicação web para os professores.

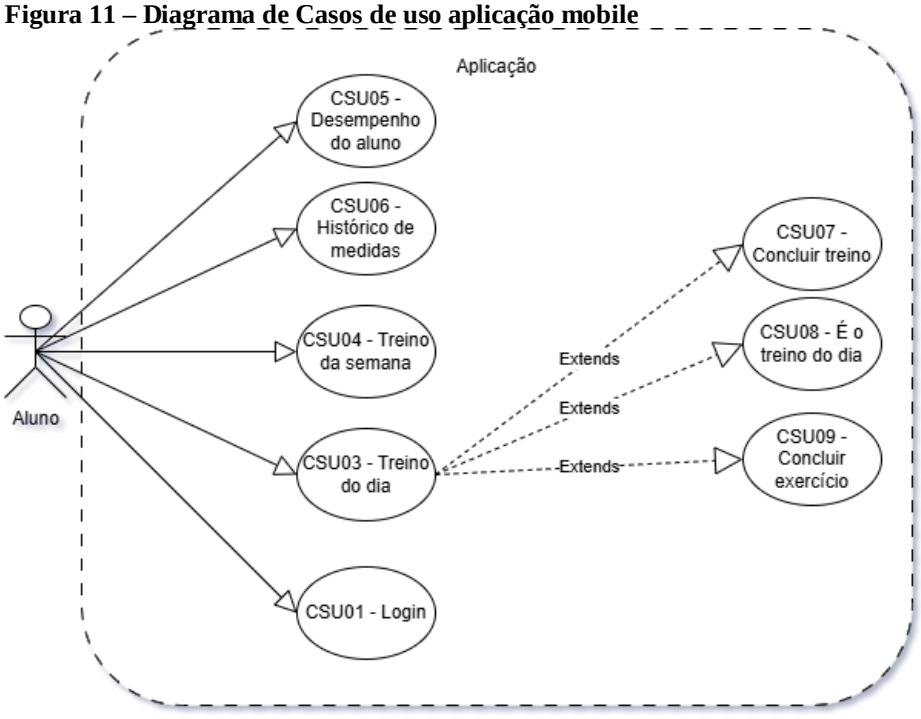
Figura 10 – Diagrama de Casos de uso da aplicação web



Fonte: Acervo do autor.

Na Figura 11, demonstra-se o diagrama de caso de uso da aplicação mobile para o aluno.

Figura 11 – Diagrama de Casos de uso aplicação mobile



Fonte: Acervo do autor.

4.6 IMPLEMENTAÇÃO

Para implementação foram separados em três partes, sendo (1) a implementação do sistema WEB, (2) Aplicativo mobile e (3) conexão com a API, conforme descritos nos tópicos seguintes

4.6.1 **Implementação Sistema Web**

Para o desenvolvimento da aplicação web foi utilizado HTML, Tailwind CSS, React Hook Form, Typescript, React, NextJS e para o desenvolvimento dos protótipos foi utilizado o Figma.

O desenvolvimento foi realizado usando o Next.js, um framework baseado em React, em conjunto com o TypeScript, que é um *superset* de JavaScript. O TypeScript melhora a legibilidade do código e permite validações de tipos durante o desenvolvimento. O Tailwind CSS foi escolhido para acelerar o processo de estilização, permitindo que o CSS seja escrito diretamente no HTML de forma mais concisa que o CSS tradicional. Além disso, o Tailwind carrega apenas os estilos necessários para cada página, otimizando o desempenho.

Para um desenvolvimento ainda mais ágil, fora utilizada a biblioteca *shadcn*, que oferece componentes prontos e permite controle total sobre a estilização. O React Hook Form foi utilizado para a validação dos formulários, eliminando a necessidade de arquivos complexos para validação de tipos ou dados.

Na Figura 12, é apresentado a título de exemplo uma parte do código da tela de login da aplicação, onde os usuários realizam o acesso. Nessa implementação, utilizamos o Tailwind CSS para a estilização, componentes do shadcn para os inputs, e o React Hook Form para a validação dos campos.

Figura 12 - Código form login

```
<form onSubmit={handleSubmit(handleSubmitLogin)} method="post" className="flex flex-col items-center gap-1">

              {/* codigo ou e-amil */}
              <div className="w-full mx-10">

                <label htmlFor="user" className="text-sm text-blue-900">Usuário</label>

                <Input {...register("user")}
                  placeholder="código ou e-mail"
                  variant={errors?.user && "error"}
                  autoComplete="username"/>

                {errors?.user ? <span className="text-xs text-red-500 ">{errors.user.message}</span> : <span> </span>}

              </div>

               {/* senha */}
              <div className="w-full">
                <label htmlFor="password" className="text-sm text-blue-900">Senha</label>
                <Input {...register("password")}
                  type={viewPassword ? "text" : "password"}
                  variant={errors?.password && "error"}
                  placeholder="senha"
                  autoComplete="current-password"
                />
                {errors?.password ? <span className="text-xs text-red-500">{errors.password.message}</span> : <span> </span>}
              </div>

              {/* checkbox */}
              <div className="flex mb-4 gap-2 w-full items-end">
                <Checkbox id="terms"  onClick={() => setViewPassword(value => !value)}/>
                <label
                  htmlFor="terms"
                  className="text-sm text-blue-900 leading-none cursor-pointer align-bottom"
                >
                  Exibir senha
                </label>
              </div>

              <Button type="submit" className="min-w-28" disabled={isSubmitting}>{isSubmitting ? <Spinner /> : "LOGIN"}</Button>

            </form>
```

Fonte: Acervo do autor.

Com a validação dos campos, usando o React Hook Form em conjunto com o Zod, onde definimos as regras de validação e os tipos de cada campo. Na Figura 13, mostramos o exemplo da validação dos inputs de login. É necessário utilizar a diretiva ‘use client’ neste arquivo, pois as validações são realizadas no lado do cliente.

Figura 13 - Criação de uma validação de um input

```
'use client';
import { z } from "zod";

const emailSchema = z.string()
    .email({message: "E-mail inválido"})
    .endsWith("unidavi.edu.br", {message: "O e-mail não é o da instituição!"});

const codeSchema = z.coerce.number()
    .min(1,{message: "Digite um código ou e-mail válido."})

export const LoginFromSchema = z.object({
    user: z.union([codeSchema, emailSchema]),
    password: z.string().min(1,{message: "Digite sua senha."})
})
```

Fonte: Acervo do autor.

Na Figura 14, apresentamos como as validações são integradas ao React Hook Form, e passado para o *resolver* LoginFromSchema, e no *mode* onde informamos “onBlur” que significa que será verificado sempre que o usuário sai de um campo.

Figura 14 - Integração React-Hook-Forms com Zod

```
const {
  handleSubmit,
  register,
  formState : {errors , isSubmitting}
} = useForm<TypeLogin>({
  mode: 'onBlur',
  resolver: zodResolver(LoginFromSchema)
})
```

Fonte: Acervo do autor.

4.6.2 **Implementação Aplicativo Mobile**

Para o desenvolvimento do aplicativo mobile utilizamos React-Native com a biblioteca EXPO, Typescript, para os protótipos também foi utilizado o Figma.

O React-Native tem algumas diferenças em relação ao React apesar de ser a mesma forma de codificação não usa HTML para desenvolver os componentes e telas, são utilizados componentes que vão dar lugar a componentes nativos após a compilação para código nativo, um exemplo é a Tag HTML *div*, que no React-Native utilizamos View aonde englobamos nosso

conteúdo, um texto não pode ser passado diretamente para uma View como no HTML é possível no React-Native precisamos envolver o texto com uma Tag Text.

Na Figura 15 onde tem-se um componente de Modal que tem a implementação de uma Tag View que está agrupando todos os demais componentes da tela e adicionando um style, temos a Tag de Text como citado anteriormente não podemos colocar apenas texto em nossa View. Um Componente de Input aonde vamos capturar o peso que o aluno digitar, e um componente de Button aonde temos o onPress que irá fazer a chamada API que irá persistir os dados, par consumo futuro.

Figura 15 - Implementação Modal

```
<Modal
                title="Anotações"
                animationType="fade"
                onClose={handleCloseAnotacao}
                visible={showModalAnotacoes}
            >
                <View style={{marginVertical: 10, rowGap: 10}}>
                    <Text>Anote aqui o peso que você ultilizou no treino.</Text>
                    <Text>ID EXERCICIOS: {idExercicio}</Text>

                    <Input title="peso(kg)">
                        <Input.Field
                            keyboardType="numeric"
                            maxLength={peso.includes(".") ? 5 : 4}
                            onChangeText={text => setPeso(text.replace(',', '.'))}
                            value={peso}
                        />
                    </Input>

                    <View style={{flexDirection: "row", justifyContent: "space-around"}}>

                        <Button style={{flex:1}} size="lg" onPress={handleSalvarAnotacao} isLoading={isLoading}>
                            <Button.Title>Salvar</Button.Title>
                        </Button>

                    </View>

                    <View style={{height: 40}}/>

                </View>
            </Modal>
```

Fonte: Acervo do autor.

4.6.3 Conexão com API

Em ambos os sistemas Mobile e WEB vemos utilizar o Axios, uma biblioteca que nos auxilia a fazermos conexões com APIs de forma simples e eficiente.

Para utilização do Axios precisamos primeiramente instanciá-lo, aonde vamos definir sua baseURL, assim não precisaremos digitar a rota completa quando quisermos acessar alguma rota na API, vamos utilizar o exemplo do nosso sistema Web mas a mesma configuração é feita em ambos os sistemas, a Figura 16 apresenta como está sendo instanciado o Axios e estamos utilizando algumas de suas funcionalidades que é não precisamos passar um header com Authorization em cada requisição, definimos isso na quando instanciamos ele e desta forma

todas nossas requisições automaticamente terão a Authorization e por fim exportamos nossa apiUniFit.

Figura 16 - Conexão Axios

```
import { enum_NameCookies } from "@/lib/enuns";
import axios from "axios";
import Cookies from "js-cookie";

const token = Cookies.get(enum_NameCookies.TOKEN)

const apiUniFit = axios.create({
    baseURL: "http://localhost:3000",
})

apiUniFit.interceptors.request.use(function (config) {
    return config;
  }, function (error) {
    return Promise.reject(error);
});

apiUniFit.defaults.headers.common['Authorization'] = `Bearer ${token}`;

export default apiUniFit;
```

Fonte: Acervo do autor.

Finalizada a implementação da instância do Axios já podemos importar esse método apiUniFit e utilizar os verbos para realizar as requisições. Como na Figura 17 temos a utilização do verbo Post para realizar a chamada da rota (/api/login), temos algumas observações que a serem notadas, ao utilizar o método post, precisamos pássaro um body para a nossa requisição contendo usuário e senha, com isso teremos o nosso retorno que já estamos ele tipado como o tipo TypeUser que é passado logo após o método post assim quando tivermos o terno dos dados já teremos os dados tipado e prontos para uso.

Figura 17 - Implementação de uma requisição

```
import apiUniFit from "@/services/api";
    async function login({user, password}:TypeLogin){
        try {
            const dados = await apiUniFit.post<TypeUser>('/api/login', {user,password})

            if (dados.status === 200){
                const agora = new Date();
                const dozeHorasEmMilissegundos = 12 * 60 * 60 * 1000; // 12 horas em milissegundos
                const dataExpiracao = new Date(agora.getTime() + dozeHorasEmMilissegundos);

                Cookies.set(enum_NameCookies.USER, JSON.stringify(dados.data), {secure: true})
                Cookies.set(enum_NameCookies.TOKEN, JSON.stringify(dados.data.token), {secure: true, expires: dataExpiracao})
                Cookies.set(enum_NameCookies.TYPE_USER, JSON.stringify(dados.data.userType), {secure: true})

                return ["Login realizado com sucesso" , true]
            }

        } catch (error: any) {
            return ["erro ao realizar login" , false]
        }
    }
```

Fonte: Acervo do autor.

4.7 UTILIZAÇÃO E FUNCIONAMENTO SISTEMA WEB

A aplicação web deste projeto foi desenvolvida para atender ao conjunto de requisitos elicitados e descritos anteriormente, foram desenvolvidas telas que são de caráter obrigatório para o funcionamento do sistema, como tela de Login, Tela Inicial, Cadastro de Professor e Aluno, Prescrição de treino e apara o sistema mobile Login, Tela Inicial e Listagem de Exercícios.

A tela inicial do sistema web tem apenas a navegação do usuário para as demais telas , sendo necessário fazer login antes para ter acesso a esta tela e as demais.

A tela de cadastro de professor ou aluno ambos contêm campos para cadastrar um novo cadastro, sendo necessário preencher alguns campos obrigatórios, aqueles que conterem um asterisco no título do campo, os demais campos podem ser deixados em branco e preenchido posteriormente.

A tela de prescrição de treino contém as informações necessárias para se cadastra um novo treino para um aluno, aonde será preciso selecionar um aluno para o treino, uma data de início e fim que será a data de duração deste treino, bem como a seleção dos exercícios que o aluno executara, tendo a possibilidade de alteração de repetições, tempo de descaço e series individualmente por exercício selecionado.

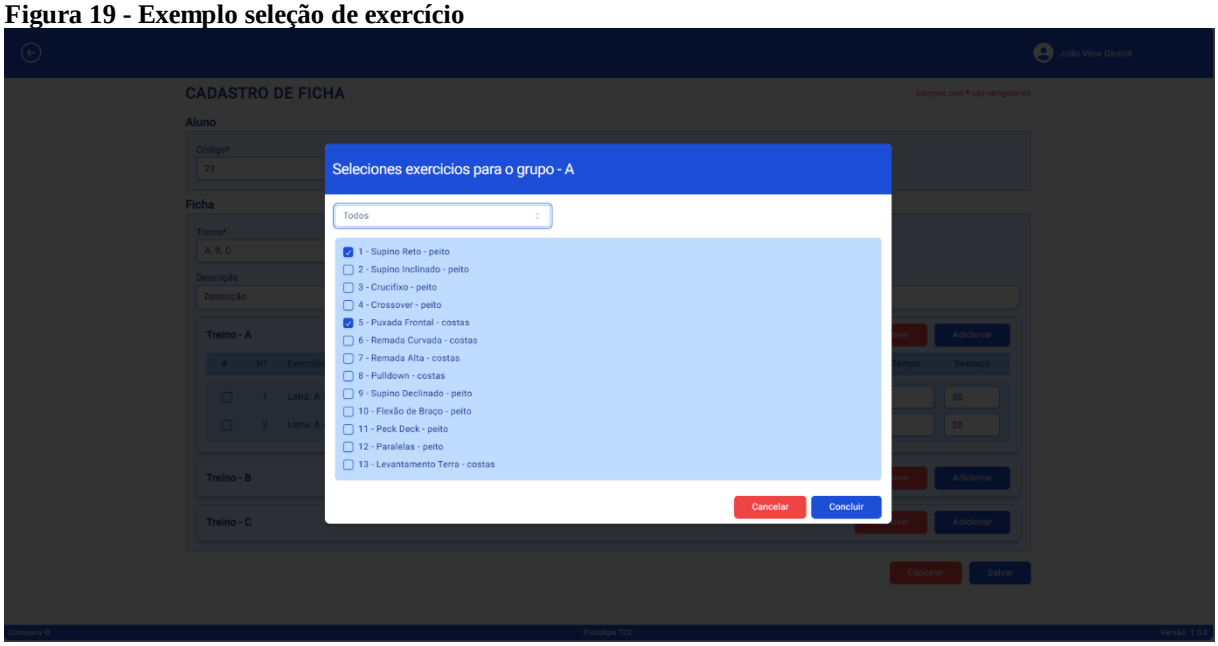
Na Figura 18 temos um exemplo de um cadastro de ficha para um novo aluno fictício, com os dados do aluno preenchido o instrutor seleciona os exercícios para caga treino, e após selecionar os exercícios como exemplificado na Figura 19, ele preenche os campos referente a séries, repetições e descanso e assim finaliza o cadastro de uma nova ficha de treino.

Figura 18 - Exemplo cadastro de ficha

Fonte: Acervo do autor.

Figura 19 seleção de exercícios a serem realizados em um determinado grupo.

Figura 19 - Exemplo seleção de exercício



Fonte: Acervo do autor.

4.7 UTILIZAÇÃO E FUNCIONAMENTO APLICATIVO MOBILE

A aplicação mobile foi desenvolvida com foco em atender ao aluno, implementando funcionalidades essenciais para o uso dos alunos que frequentarão a academia, também descritas nos requisitos funcionais. O aplicativo contempla telas fundamentais como a de Login, Tela Inicial e Listagem de Exercícios.

A Tela Inicial permite ao aluno visualização inicial para poder navegar entre as demais telas.

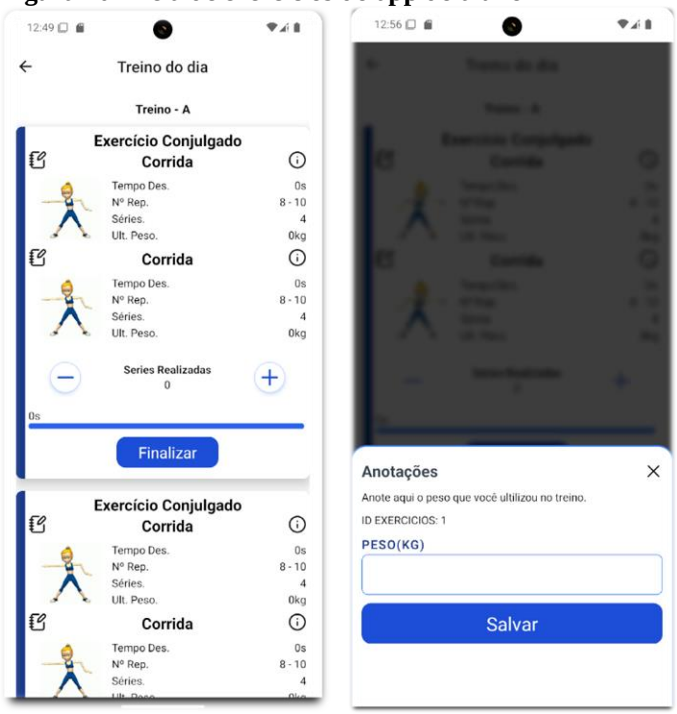
A tela de Listagem de Exercícios cada card de exercício exibe informações relevantes, como o nome do exercício, número de repetições, séries e o peso utilizado na última sessão. Um dos diferenciais dessa tela é a possibilidade de visualizar os exercícios conjugados, onde dois exercícios são combinados em uma única sessão de treino como pode ser visto na Figura 20.

A interação principal nesta tela se dá pela opção de o usuário registrar o peso utilizado durante o exercício. Ao clicar no ícone de edição localizado no card do exercício, abre o modal inferior. Nessa tela de Anotações, é possível registrar o peso utilizado para cada exercício, garantindo o acompanhamento e progressão do treino.

No formulário de anotações, o campo de "Peso (kg)" é de preenchimento e salvo para quando o aluno for realizar novamente o exercício ele ter essa informação a disposição.

Ainda na tela principal o aluno tem a opção de ir marcando as series que ele realizou para um melhor acompanhamento dos exercícios, e ao finalizar um exercício ele pode clicar em finalizar para finalizar aquele exercício e prosseguir para o próximo.

Figura 20 - Tela de exercícios do app do aluno



Fonte: Acervo do autor

5. CONCLUSÃO

O presente projeto foi pensado na criação de uma interface de usuário para um sistema de prescrição e avaliação de treinos em academias. O objetivo principal era otimizar o controle de treinos e o acompanhamento do desempenho dos alunos, substituindo as fichas de papel por um sistema digital integrado e moderno. Para o desenvolvimento da aplicação, foram utilizadas tecnologias modernas como React, Next.js e React Native, sendo estas as principais tecnologias para implementar as funcionalidades do sistema, tanto na versão web quanto na mobile.

Com base no estado da arte e na tabela comparativa realizada neste trabalho, observou-se que, embora existam outras aplicações de gestão de academia, mostra que o mercado ainda está bem aquecido para este ramo, sendo o projeto desenvolvido para atender a essas necessidades, oferecendo um sistema intuitivo que permite o controle detalhado de treinos e desempenho dos alunos.

Além disso, foi realizada a integração com uma API back-end para gerenciamento de dados, o que possibilita um fluxo de informação eficiente entre as interfaces web e mobile. Foram desenvolvidas funcionalidades como a prescrição de treinos, ajuste de repetições, tempo de descanso, e acompanhamento tanto pelos professores quanto pelos alunos.

Como planejado, as dificuldades do dia a dia na academia foram identificadas, os requisitos necessários para a criação do sistema foram levantados, e as tecnologias mais adequadas foram selecionadas para o desenvolvimento. Importante citar também, os protótipos de telas que foram construídos e implementados na camada de front-end.

Assim, pode-se concluir que os objetivos propostos para este trabalho foram alcançados com sucesso. Entretanto, considerando as limitações deste trabalho, no tópico 5.1 serão listadas algumas recomendações para trabalhos futuros, incluindo melhorias e novas funcionalidades para o protótipo.

5.1 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Como recomendação de continuidade desse projeto, uma delas seria a implantação de novas funcionalidades. No sistema Web, ter funcionalidade para os professores através de um dashboard, onde possa acompanhar o progresso de seus alunos. Funcionalidade de auxílio na montagem de treinos por inteligência artificial, para que o professor escolha o melhor exercício com base nos dados do aluno.

Para o aplicativo mobile do aluno, a implementação de bonificações a fim de gerar mais engajamento entre o aluno e a academia. Funcionalidade de um leitor de QR Code para que o aluno possa escanear o código presente nas máquinas e acessar informações sobre a funcionalidade de cada aparelho.

Implementar maior segurança no login utilizando de protocolos como o oAuth2, disponibilizar autenticação de dois fatores, implementação de técnicas para evitar IDOR (Insecure Direct Object Reference), como utilização de UUIDs no lugar de IDs para pesquisas por alguma informação do usuário.

Adequar o sistema ou ter um módulo para atendimento de personal e alunos. Uma parte dos alunos possui personal próprio e pode ser um outro nicho de mercado para o projeto.

Por fim, a realização de diferentes tipos de testes mais aprofundados com um grupo de usuários da academia, a fim de validar o sistema em sua totalidade.

REFERÊNCIAS

ADRIANO. **Guia prático de TypeScript**: Melhore suas aplicações JavaScript, 2022. Disponível em: <https://kupdf.net/download/guia-pratico-de-typescript-melhore-suas-aplicacoes-javascript_62e3a1a4e2b6f542324ba281_pdf> Acesso em 17 abr. 2024.

ALVES, William Pereira. **HTML & CSS:** aprenda como construir páginas web. São Paulo Expressa 2021. *E-book.*

AXIOS, **Introdução**, 2024. Disponível em < https://axios-http.com/docs/intro > Acesso em 28 ago 2024.

BROOKSHEAR, J. Glenn. Ciência da computação: uma visão abrangente. 11. Porto Alegre: Bookman 2013

DEITEL, Paul J. **Android 6 para programadores:** uma abordagem baseada em aplicativos. 3. São Paulo Bookman 2016. *E-book.*

DYNAMO, **Consistência Nos Exercícios**, 2024. Disponível em < https://academiadynamo.com.br/consistencia-nos-exercicios/ > Acesso em 01 Nov. 2024.

EXPO, **Desenvolvendo um aplicativo com Expo**, 2024a. Disponível em: < https://docs.expo.dev/workflow/overview/#what-is-the-difference-between-expo-and > Acesso em 25 ago. 2024.

EXPO, **Expo SecureStore**, 2024b. Disponível em: <https://docs.expo.dev/versions/latest/sdk/securestore/ > Acesso em 26 ago. 2024.

FERREIRA, Arthur Gonçalves. **Design patterns e gerência de configuração:** do projeto ao controle de versões. São Paulo Platos Soluções Educacionais 2021. *E-book.*

FIGMA, **Guia para prototipagem no Figma**, 2024. Disponível em: < Guia para prototipagem no Figma – Figma Learn - Central de Ajuda> Acesso em 25 ago. 2024.

MARCOLINO, Anderson da Silva. **Frameworks front end,** São Paulo Platos Soluções Educacionais 2021. *E-book.*

MDN WEB DOCS. **CSS básico**, 2024. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/CSS_basics > Acesso em 17 abr. 2024.

MEDICINA SA, **Levantamento mostra que Brasil tem mais de 32 mil academias**, 2024. Disponível em: < https://medicinasa.com.br/academias-brasil/#:~:text=O%20Brasil%20conta%20hoje%20com,pratica%20muscula%C3%A7%C3%A3o%20e%20outros%20esportes > Acesso em 07 abr. 2024.

MEDIUM, **Construindo formulários dinâmicos com o React Hook Form**, 2023. Disponível em: < https://medium.com/@greennolgaa/building-dynamic-forms-with-react-hook-form-2119c144d717 > Acesso em 28 nov. 2024.

MILETTO, Evandro M.; BERTAGNOLLI, Silvia C. **Desenvolvimento de software ii**: introdução ao desenvolvimento web com html, css, javascript e php. Porto Alegre Bookman 2014. *E-book.*

NEXT.JS, **Armazenamento em cache no Next.js**, 2024a. Disponível em: < https://nextjs.org/docs > Acesso em 07 abr. 2024.

NEXT.JS, **Introdução**, 2024b. Disponível em: < https://nextjs.org/docs/app/building-your-application/caching > Acesso em 07 abr. 2024.

OLIVEIRA, Cláudio Luís Vieira. **JavaScript descomplicado:** programação para a Web, IoT e dispositivos móveis. São Paulo Erica 2020. *E-book.*

PRESSMAN, Roger, S.; MAXIM, Bruce R. Engenharia de software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021. Ebook.

REACT, **Início Rapido**, 2024. Disponível em: < https://react.dev/learn > Acesso em 22 abr. 2024.

REACT-NATIVE, **React-Native**, 2024. Disponível em < https://reactnative.dev/docs/intro-react-native-components > Acesso em 25 ago. 2024.

SAUDATE, Alexandro. **REST:** construa API's inteligentes de maneira simples. 1. ed. São Paulo: 2013. Disponível em: <https://archive.org/details/livros-rest-construa-apis-inteligentes-de-maneira-simples/page/20/mode/2up> Acesso em: 17 jun. 2024.

SOMMERVILLE, Ian. Engenharia de Software. São Paulo: Pearson Prentice Hall, 2011.

SWAGGER, 2024. Disponível em < https://swagger.io/tools/swagger-ui/ > Acesso em 29 abr. 2024.

TAILWINDCSS, **Introdução ao Tailwind CSS**, 2024. Disponível em: < https://tailwindcss.com/docs/installation > Acesso em 08 abr. 2024.

TURNEL, Evandro Carlos. **HTML 5:** guia prático. 2. São Paulo Erica 2014. *E-book*

TYPESCRIPT, **TypeScript para o novo programador**, 2024. Disponível em:< https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html > Acesso em 07 abr. 2024.

VERCEL, **Introdução à Vercel**, 2024. Disponível em:< https://vercel.com/docs/getting-started-with-vercel > Acesso em 23 abr. 2024.

VISUAL STUDIO CODE, **Primeiros passos**, 2024. Disponível em < https://code.visualstudio.com/docs > Acesso em 25 abr. 2024.

ZOD, **Introdução**, 2024. Disponível em < https://zod.dev/?id=introduction > Acesso em 15 abr. 2024.