UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES
CAMPUS ERECHIM
DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

IGOR DALLAZEN FOLLADOR

DESENVOLVIMENTO DE UM SISTEMA INTEGRADO A SMARTWATCHES PARA ACOMPANHAMENTO DE USUÁRIOS COM PRÁTICAS ESPORTIVAS

ERECHIM - RS
2023

IGOR DALLAZEN FOLLADOR

DESENVOLVIMENTO DE UM SISTEMA INTEGRADO A SMARTWATCHES PARA ACOMPANHAMENTO DE USUÁRIOS COM PRÁTICAS ESPORTIVAS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel, Departamento de Engenharias e Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e das Missões Campus Erechim.

Orientador: Malomar Alex Seminotti

ERECHIM - RS
2023

AGRADECIMENTOS

Gostaria de expressar minha mais profunda gratidão e apreço a todos aqueles que me apoiaram e contribuíram para a realização deste trabalho.

Em primeiro lugar, um agradecimento especial aos meus queridos pais, cujo amor, apoio e encorajamento foram incondicionais em todas as etapas da minha vida. Sempre estiveram ao meu lado, e esta fase não foi exceção. Sua presença e suporte contínuo foram fundamentais para a minha jornada e sucesso.

Um agradecimento sincero ao meu orientador, Malomar Alex Seminotti, por sua excelente orientação, paciência e conhecimento. Seu apoio foi crucial não só no desenvolvimento deste trabalho, mas também na minha formação acadêmica e profissional.

Também gostaria de estender meus agradecimentos a todo o corpo docente do curso de Ciência da Computação. A dedicação, conhecimento e paixão de cada um de vocês pelo ensino foram essenciais para tornar possível o desenvolvimento deste projeto. Vocês foram uma fonte de inspiração e um modelo a seguir.

Por fim, agradeço a todos que, direta ou indiretamente, fizeram parte desta jornada. Cada contribuição foi valiosa para a concretização deste trabalho.

"Se o conhecimento pode criar problemas, não é através da ignorância que podemos solucioná-los."

(Isaac Asimov)

RESUMO

Este trabalho trata-se do desenvolvimento de um sistema integrado a *smartwatches* para o acompanhamento e avaliação de pessoas que possuem práticas esportivas. O objetivo é possibilitar o compartilhamento de dados de saúde entre usuários e profissionais. Os *smartwatches*, equipados com sensores, coletam informações constantemente, permitindo um acompanhamento mais preciso e individualizado. A aplicação fornecerá relatórios aos profissionais de saúde com informações relevantes para a prescrição de treinos e planos alimentares mais eficazes. A utilização de tecnologias *wearable* em conjunto com a aplicações web tem o potencial de melhorar o cuidado com pacientes, especialmente aqueles com histórico familiar de doenças cardíacas, fumantes e obesos.

Palavras-chave: Aplicativo. Saúde. Wearable. Smartwatches. Esportes.

ABSTRACT

This work involves the development of a system integrated with smartwatches to monitor and evaluate people who practice sports. The objective is to enable the sharing of health data between users and professionals. Smartwatches, equipped with sensors, constantly collect information, allowing for more precise and individualized monitoring. The application will provide reports to healthcare professionals with relevant information for prescribing more effective training and eating plans. The use of wearable technologies in conjunction with web applications has the potential to improve care for patients, especially those with a family history of heart disease, smokers and obese patients.

Key words: App. Health. Wearable. Smartwatches. Sports.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

HTML	*HyperText Mackup Language*
CSS	*Cascade Style Sheets*
API	*Application Progress Interface*
DNS	*Domain Name System*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
BD	Banco de dados
JSON	*JavaScript Object Notation*
SQL	*Structured Query Language*
APP	Aplicativo
REST	*Representational State Transfer*
UML	*Unified Modeling Language*
NPM	*Node Package Manager*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*

SUMÁRIO

1 INTRODUÇÃO

O acompanhamento e avaliação de pessoas com práticas esportivas ou pacientes são essenciais para o sucesso do tratamento e obtenção de resultados desejados. No entanto, estimar o gasto calórico diário dos pacientes é uma tarefa complexa, pois as informações são voláteis e podem variar de acordo com diversos fatores, como tipo de treino, cansaço físico acumulado, entre outros.

Para mitigar esse problema, este trabalho desenvolve uma aplicação integrada a *smartwatches*. Os *smartwatches* possuem sensores que coletam informações constantemente, o que permite um acompanhamento mais preciso e individualizado do paciente.

Com a aplicação desenvolvida, profissionais de saúde, como personal trainers, nutricionistas e médicos, terão acesso a relatórios que fornecem informações relevantes para a prescrição de treinos e planos alimentares mais eficazes, além da possibilidade de criação de notificações personalizadas para os mesmos.

A utilização de tecnologias *wearable* em conjunto com a aplicação móvel pode auxiliar na forma como os pacientes são acompanhados e avaliados, trazendo benefícios significativos para a saúde e desempenho esportivo, além de mudar a forma como profissionais da saúde realizam acompanhamentos de pessoas ditas saudáveis ou de pacientes com doenças relacionadas ao sedentarismo. A ferramenta surgiu para melhorar o cuidado com pacientes que possuam histórico familiar de doenças cardíacas/cardiovasculares, fumantes e obesos.

O Capítulo 2 trata da conceituação básica sobre tipos de esportes e formas de monitoramento dos mesmos. No Capítulo 3 são apresentados os conceitos teóricos e as tecnologias empregadas no desenvolvimento da aplicação. No Capítulo 4 é detalhado o processo de criação da aplicação, suas funcionalidades e a modelagem UML. A descrição operacional do sistema e as interfaces do Wearable FIT são apresentadas no Capítulo 5. O Capítulo 6 discorre sobre as conclusões finais e os resultados alcançados com o desenvolvimento da aplicação, além de perspectivas para futuros aprimoramentos do sistema.

2 PRÁTICAS ESPORTIVAS

Ao longo das últimas décadas, pode-se testemunhar um notável aumento na expectativa de vida humana (ARAUJO, 2010), transformando a longevidade de uma conquista extraordinária em uma realidade cada vez mais comum.

No cenário atual, onde a média de expectativa de vida em países desenvolvidos se situa em torno de oitenta anos (ARAUJO, 2010), e as pesquisas biomédicas indicam um potencial genético que sugere a possibilidade de ultrapassar a marca dos cem anos, a busca por uma vida prolongada tornou-se uma aspiração tangível.

O acesso aos avanços científicos e tecnológicos relacionados à saúde é fundamental, abrangendo inovações em técnicas, procedimentos, medicamentos, vacinas e conhecimentos emergentes sobre alimentação, bem como os efeitos agudos e crônicos do exercício físico.

Este cenário de progresso científico e tecnológico cria um terreno propício para explorar e integrar práticas esportivas inovadoras, especialmente quando combinadas com o potencial oferecido por tecnologias *wearables*.

2.1 Tipos de esportes

Os tipos de esportes podem ser amplamente categorizados em várias modalidades, incluindo esportes de equipe, individuais, de resistência, de força, entre outros. Cada categoria tem suas características únicas e contribui de maneira diferente para o desenvolvimento físico e mental dos indivíduos (SIEDENTOP, 2019).

- **Esportes de Equipe:** futebol, basquete, voleibol e hóquei, enfatizam a cooperação e a estratégia de equipe. Eles não apenas promovem a aptidão física, mas também habilidades sociais como trabalho em equipe, liderança e comunicação.
- **Esportes Individuais:** os esportes individuais, como atletismo, natação, ginástica e tênis, focam na habilidade, resistência e determinação do indivíduo. Esses esportes são frequentemente reconhecidos por sua capacidade de desenvolver a autoconfiança e a autodisciplina.
- **Esportes de Resistência:** maratonas, triatlos e ciclismo de longa distância, testam a resistência e a resistência dos atletas. Eles são excelentes para melhorar a saúde cardiovascular e a resistência física.
- **Esportes de Força:** levantamento de peso e *bodybuilding* são exemplos de esportes de força. Eles focam no desenvolvimento da força muscular e são benéficos para aumentar a massa muscular e a densidade óssea.
- **Esportes de combate:** boxe, judô e taekwondo, envolvem competições diretas entre dois adversários, visando desenvolver habilidades de autodefesa, além de promover resistência, agilidade e força.

- **Esportes Adaptativos:** são projetados para atletas com deficiências. Eles incluem basquete em cadeira de rodas, vôlei sentado e atletismo paralímpico, proporcionando oportunidades para atividades físicas inclusivas e competitivas.

2.1.1 O papel dos *smartwatches* na saúde

Os *smartwatches* se tornaram aliados valiosos para aqueles que praticam esportes, independentemente do nível de experiência. Eles oferecem uma série de recursos que auxiliam na monitorização do desempenho físico, acompanhamento da saúde e motivação para a prática regular de exercícios. Como citado no suporte do GOOGLE (2023), alguns dos recursos incluem:

Monitoramento de Atividades: os *smartwatches* registram automaticamente informações sobre passos, distâncias percorridas, calorias queimadas e frequência cardíaca durante o exercício, fornecendo um panorama detalhado do desempenho físico.

Acompanhamento de Progresso: os dispositivos mantêm um histórico das atividades realizadas, permitindo que os praticantes de esportes acompanhem o progresso ao longo do tempo e estabeleçam metas alcançáveis.

Motivação: aplicativos e notificações integrados incentivam os usuários a se manterem ativos e a atingirem seus objetivos de *fitness*, promovendo um estilo de vida mais saudável. Funcionalidades Específicas para Esportes: alguns *smartwatches* vêm com recursos específicos para diferentes modalidades esportivas, como rastreamento de GPS para corrida e métricas de natação para natação.

Monitoramento da Saúde: além das atividades esportivas, os *smartwatches* podem rastrear indicadores de saúde, como qualidade do sono, níveis de estresse e até mesmo oferecer lembretes para a hidratação.

2.2 Análise de sistemas de monitoramento de *smartwatches* existentes

Com a crescente popularidade dos *smartwatches*, os aplicativos de monitoramento de saúde e *fitness* tornaram-se ferramentas essenciais para acompanhar e analisar uma ampla gama de dados relacionados ao bem-estar e à atividade física. Esses aplicativos coletam dados em tempo real, oferecendo *insights* valiosos sobre a saúde do usuário. Nesta seção, serão explorados dois dos aplicativos de monitoramento mais populares integrados a *smartwatches*: o *Google Fit* e o *Samsung Health*.

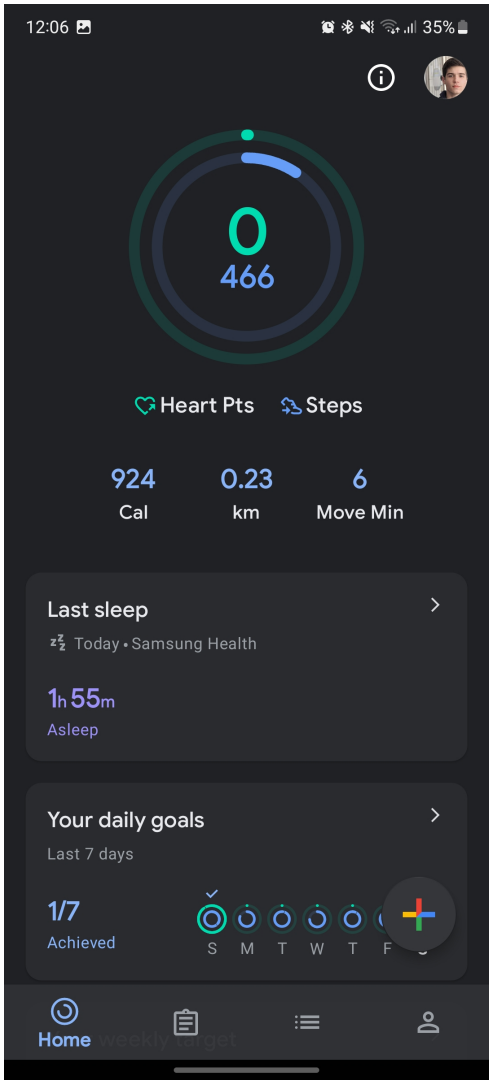
2.2.1 Aplicativo *Google FIT*

O *Google Fit*, desenvolvido pelo Google, é uma plataforma abrangente que rastreia atividades físicas, como caminhada, corrida e ciclismo, utilizando dados coletados pelo dispositivo *wearable* ou *smartphone* do usuário. Este aplicativo se destaca pela sua interface intuitiva e pela

capacidade de integrar informações de uma variedade de fontes e dispositivos, proporcionando uma visão holística da atividade física do usuário.

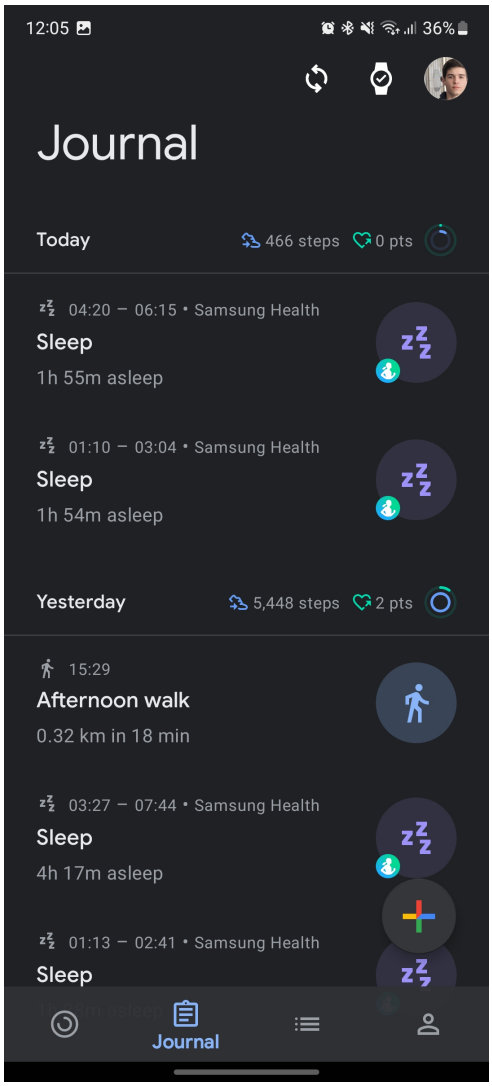
Na Figura 1 é possível visualizar as métricas gerais que são entregues pelo app do Google FIT e na Figura 2 é visualizado o diário de atividades executadas pelo usuário ao longo dos dias, assim como a origem destas informações, pois as mesmas podem ter sido integradas por meio de outro aplicativo de análise de dados de dispositivos vestíveis, como é o caso apresentado na mesma figura.

Figura 1 – Tela inicial do *Google FIT*



Fonte: Autor

Figura 2 – Diário de atividades do *Google FIT*



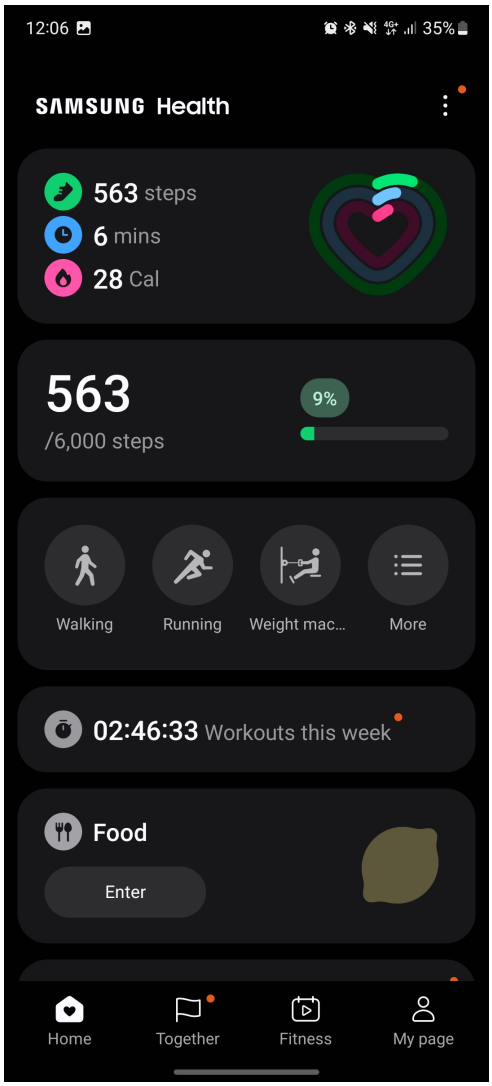
Fonte: Autor

2.2.2 Aplicativo *Samsung Health*

O *Samsung Health*, desenvolvido pela *Samsung Electronics*, oferece funcionalidades semelhantes, mas com alguns recursos adicionais específicos para dispositivos Samsung. Ele não apenas rastreia atividades físicas, mas também monitora aspectos como nutrição, qualidade do sono e saúde mental, tornando-se uma solução completa para o monitoramento do estilo de vida. Contudo, seu funcionamento é voltado para o uso em produtos da própria marca, estando disponível apenas na loja de aplicativos da própria Samsung.

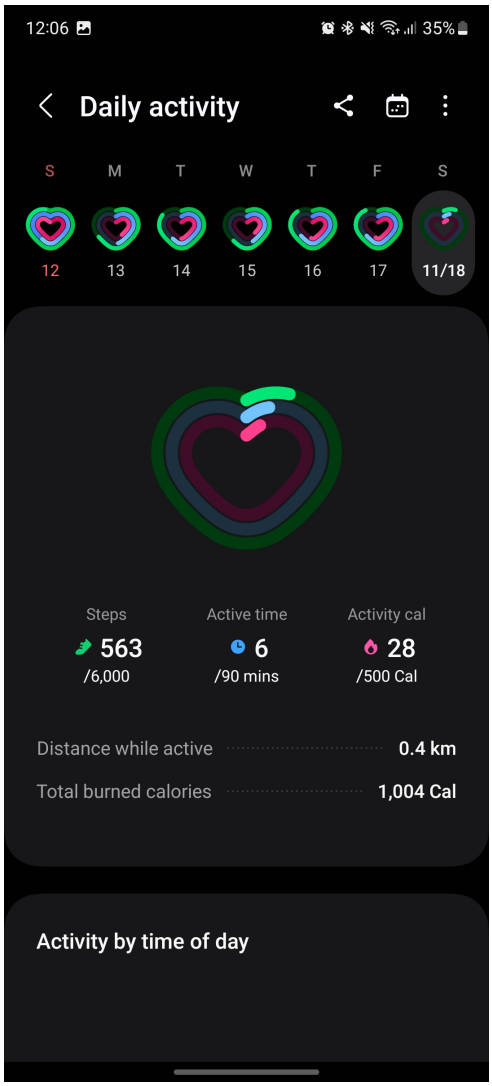
A Figura 3 e a Figura 4 demonstram os dados monitorados pelo *Samsung Health*, assim como sua funcionalidade de acompanhamento de atividades com objetivos diários.

Figura 3 – Tela inicial do *Samsung Health*



Fonte: Autor

Figura 4 – Diário de atividades do Samsung Health



Fonte: Autor

3 TECNOLOGIAS

Nesta seção são abordados os fundamentos e conceitos relacionados às tecnologias vestíveis, bem como os princípios do desenvolvimento *web* e as tecnologias utilizadas na implementação dessas soluções, proporcionando uma base teórica sólida para a compreensão e desenvolvimento do projeto.

3.1 Tecnologias vestíveis

As tecnologias vestíveis ou *wearable technology* (WT) são todos os dispositivos tecnológicos que podem ser usados como acessórios ou que podem ser vestidos (BOCARD, 2022). Dentre os dispositivos mais conhecidos, podem ser citados os *smartwatches* e *smartbands*, os quais serão contemplados neste projeto.

Dentro da Medicina, estes dispositivos já são amplamente conhecidos e utilizados, seja no acompanhamento de pacientes ou no auxílio dos profissionais como, por exemplo, o uso dos dispositivos usados na cabeça durante cirurgias, os quais possibilitam que, através de gestos, evite a contaminação das mãos do cirurgião ao realizar alguns tipos de procedimentos ou consultas.

3.1.1 *Smartwatches*

Smartwatch é um modelo de dispositivo vestível que se assemelha a relógios de pulso. Contudo, estes possuem muita tecnologia embarcada. Seu maior diferencial é combinar as funcionalidades de um relógio convencional com capacidades de *smartphones*, permitindo que os usuários tenham acesso a uma variedade de recursos, informações e aplicações diretamente no pulso (MULTISOM, 2021).

3.1.2 Tecnologia *Bluetooth*

A maior parte dos dispositivos *wearable* utilizam o protocolo *Bluetooth* para comunicação com smartphones para transferência de dados coletados pelos dispositivos. O *Bluetooth* é um protocolo de comunicação sem fio que permite que essa comunicação ocorra de forma eficiente entre dois ou mais dispositivos próximos.

Ele usa ondas de rádio de curto alcance para estabelecer conexões ponto a ponto ou em rede entre dispositivos. O *Bluetooth* opera na faixa de frequência de 2,4 GHz e usa o espectro de frequência ISM (*Industrial, Scientific and Medical*). Ele usa um sistema de comunicação em que os dispositivos *Bluetooth* formam redes pessoais sem fio chamadas de *piconets*. Em uma *piconet*, um dispositivo assume o papel de mestre (*master*) e os outros dispositivos atuam

como escravos (*slaves*). O mestre controla a comunicação na piconet (BLUETOOTH SIG, INC., 2023).

O *Bluetooth* oferece diferentes perfis, que são conjuntos de funcionalidades específicas definidas para diferentes tipos de dispositivos. Por exemplo, o perfil *Hands-Free* (HFP) é usado para fones de ouvido e sistemas de viva-voz em carros, enquanto o perfil A2DP (*Advanced Audio Distribution Profile*) é usado para transmissão de áudio estéreo de alta qualidade.

Além disso, o *Bluetooth* também suporta o conceito de BLE (*Bluetooth Low Energy*), também conhecido como *Bluetooth Smart*, que permite que dispositivos de baixo consumo de energia se comuniquem de forma eficiente em intervalos de tempo mais longos, o que é adequado para aplicações como dispositivos vestíveis (*wearables*) e sensores (BLUETOOTH SIG, INC., 2023).

3.1.3 *Google FIT*

Junto com o desenvolvimento e avanço dos dispositivos *wearable* as empresas fabricantes necessitaram de aplicações que centralizassem os dados coletados pelos aparelhos, dentre os aplicativos desenvolvidos a Google lançou uma solução que vem sendo embarcada na maior parte dos *wearables* que utilizam Android, o *Google Fit*.

Em suma, o *Google Fit* foi criado para atender à demanda crescente por uma plataforma centralizada que agrega informações de saúde e atividades físicas de diferentes dispositivos e aplicativos. Ele visa facilitar o acompanhamento, a análise e a gestão desses dados, proporcionando aos usuários uma visão holística de sua saúde e bem-estar (GOOGLE, INC., 2023).

A *Google Fit API* é uma API fornecida pelo Google que permite que os desenvolvedores acessem dados relacionados à saúde e atividades físicas dos usuários. Através da *Google Fit API*, os desenvolvedores podem obter acesso a dados como passos, distância percorrida, calorias queimadas, batimentos cardíacos, dados de atividade e muito mais (GOOGLE, INC., 2023). Esses dados são fornecidos por dispositivos vestíveis (*wearables*), como *smartwatches* e pulseiras de atividade física, além de aplicativos de saúde e fitness que utilizam o *Google Fit* como plataforma para armazenar e sincronizar os dados.

Ao utilizar a *Google Fit API*, os desenvolvedores podem criar aplicativos que se integram ao ecossistema do *Google Fit*, permitindo que os usuários visualizem, acompanhem e gerenciem suas informações de saúde e condicionamento físico em um só lugar.

A *Google Fit API* oferece uma documentação completa, com guias, exemplos e referências de programação para auxiliar os desenvolvedores na integração de seus aplicativos com o *Google Fit*. Além disso, o Google fornece suporte técnico e recursos adicionais para facilitar o desenvolvimento e a implementação bem-sucedida de aplicativos baseados na *Google Fit API*.

3.2 Desenvolvimento *web*

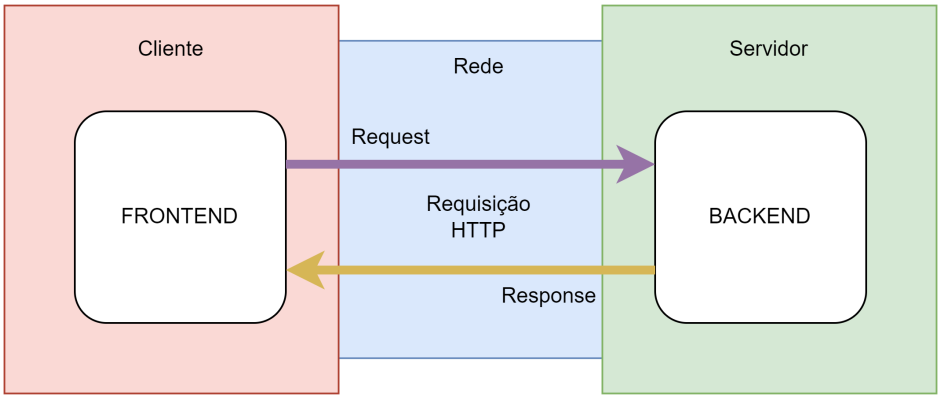
Nesta seção de desenvolvimento *web*, será explorado em detalhes o protocolo HTTP e o que é uma API que utiliza a arquitetura REST.

3.2.1 *Hypertext Transfer Protocol*

O HTTP, segundo a RFC 2616, é um protocolo de comunicação que funciona a nível de aplicação. Ele um dos protocolos de comunicação utilizados para tranferir informações na World Wide Web (GASPAR, 2021). Seu funcionamento é baseado no modelo cliente-servidor (Figura 5), onde o cliente submete pedidos (requisições) ao servidor para obter recursos como, páginas *web* (texto) e arquivos binários (imagens e videos). O servidor, por sua vez, responde a estas solicitações com os conteúdos correspondentes (SANTOS, 2023).

O protocolo estabelece as regras de como as mensagens entre cliente-servidor devem ser formatas e transmitidas. Como dito anteriormente, ele funciona na camada da aplicação, operando sob a camada de transporte que normalmente utiliza o protocolo TCP/IP como protocolo para transmitir as informações do protocolo sobrejacente (GASPAR, 2021).

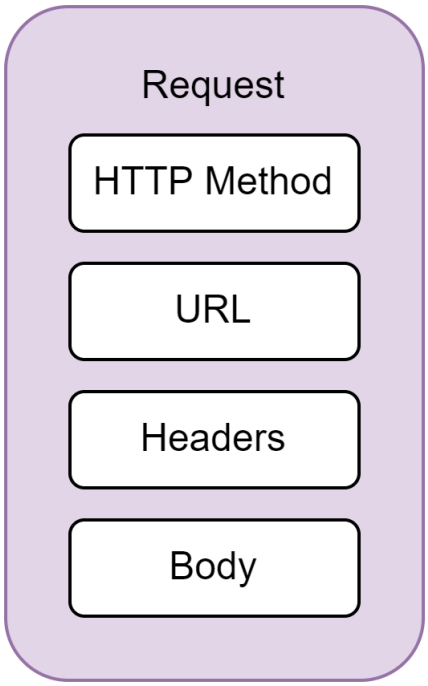
Figura 5 – Comunicação cliente-servidor



Fonte: (SANTOS, 2023)

Para que o cliente realize uma solicitação, o protocolo estabelece oito diferentes métodos (Figura 6) que possibilitam realizá-las, sendo os seguintes: *GET, POST, PUT, DELETE, HEAD, TRACE, OPTIONS e CONNECT*. Cada método possui um conjunto de cenários específicos para uso, sendo os quatro primeiros citados anteriormente os mais comuns de serem utilizados (GASPAR, 2021)

Figura 6 – Estrutura de uma requisição



Fonte: (SANTOS, 2023)

O método *GET* é utilizado para solicitar um recurso específico de um servidor (FIELDING et al., 1999). É amplamente utilizado para obter informações, como solicitar páginas da web, imagens ou documentos de um servidor. Uma característica importante do método *GET* é que ele é seguro e idempotente, o que significa que não deve ter efeitos colaterais e pode ser repetido sem causar alterações.

Por outro lado, o método *POST* é usado para enviar dados ao servidor para processamento. Geralmente, é utilizado em formulários HTML, como envio de informações de registro, mensagens ou criação de postagens em fóruns. O método *POST* não é idempotente, o que significa que enviar a mesma requisição várias vezes pode resultar em ações repetidas (FIELDING et al., 1999).

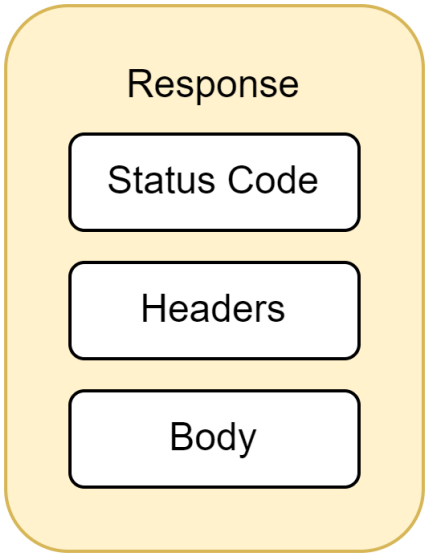
O método *PUT* é utilizado para enviar dados para um servidor, com a intenção de substituir ou atualizar um recurso existente no servidor. É idempotente, ou seja, fazer a mesma requisição várias vezes não deve ter efeitos diferentes além da atualização do recurso (FIELDING et al., 1999). O método *PUT* é comumente usado em *APIs RESTful* para atualizar informações de um recurso existente, como atualizar os detalhes de um usuário em um sistema.

Já o método *DELETE* é utilizado para solicitar a remoção de um recurso específico do servidor. Assim como o método *PUT*, o *DELETE* também é idempotente, o que significa que repetir a mesma requisição não deve causar efeitos adicionais além da remoção do recurso. É comumente usado para remover recursos, como excluir um *post* de um blog, remover um arquivo ou deletar uma entrada em um banco de dados (FIELDING et al., 1999).

Assim como métodos de solicitação, o protocolo HTTP fornece uma série códigos de status para elas. Estes códigos são utilizados principalmente pelo servidor para indicar em que estado se encontra determinada solicitação realizada pelo cliente (Figura 7). Existem diferentes contextos para a utilização destes códigos, eles possuem a sua formação por um número inteiro de três dígitos, sendo o primeiro dígito o que representa o contexto em que ele ocorre (FIELDING et al., 1999).

Respostas de informações geralmente são retornadas por códigos HTTP da faixa de 100 a 199. Respostas de sucesso da faixa 200 a 299. Já redirecionamento de requisições são dados na faixa de 300 a 399. Erros relacionados ao contexto do cliente são retornados com códigos da faixa de 400 a 499, enquanto erros relacionados no contexto do servidor ocorrem na faixa de 500 a 599 (FIELDING et al., 1999).

Figura 7 – Estrutura de uma resposta



Fonte: (SANTOS, 2023)

É importante citar que o protocolo HTTP possui uma variação chamada *Hypertext Transfer Protocol Secure* ou HTTPS que difere unicamente em uma camada de segurança adicional na qual adiciona criptografia na comunicação, permitindo assim que os dados trafegados sejam criptografados no remetente e sejam lidos unicamente pelo destinatário.

O HTTPS é geralmente baseado no protocolo SSL/TLS, protocolo de comunicação subjacente ao HTTPS. Este protocolo permite que a comunicação entre cliente-servidor ocorra de forma muito mais segura impedindo que intermediários desta comunicação tenham acesso a dados sensíveis existentes no corpo da mensagem (FIELDING et al., 1999).

3.2.2 API RESTful

Uma *Application Programming Interface* (API) é um conjunto de definições e protocolos estabelecidos para que diferentes aplicações possam integrar e interagir entre si. Ela permite que diferentes sistemas, aplicativos ou serviços se comuniquem e compartilhem informações de forma padronizada (RED HAT, INC., 2022).

Uma API RESTful, nada mais é que uma API construída seguindo a arquitetura *Representational State Transfer* (REST) que define um conjunto de regras para desenvolvimento de serviços web.

Segundo o site da Amazon Web Services (AMAZON WEB SERVICES, INC., 2023a), uma API construída dentro desta arquitetura deve seguir alguns pontos para ser considerada RESTful:

- **Arquitetura Cliente-Servidor:** a API RESTful segue o modelo cliente-servidor onde o cliente realiza solicitações de ações servidas pela máquina servidora.
- **Recursos identificáveis:** os recursos são identificados por meio de URIs (*Uniform Resource Identifiers*), como URLs, e são manipulados por meio de métodos HTTP. Cada recurso/ação do servidor tem sua própria URI única.
- **Operações baseadas em métodos HTTP:** a API deve efetuar sua comunicação através do protocolo HTTP e deve seguir o uso correto dos métodos do protocolo (*GET, POST, PUT e DELETE*) para que o cliente tenha acesso a ação desejada no servidor.
- **Ausência de estado (*stateless*):** cada requisição do cliente deve possuir todas as informações necessárias para processá-la, assim o servidor não necessita manter informações do cliente entre as solicitações.
- **Sistema em camadas:** o cliente pode se conectar a outros intermediários entre o cliente e o servidor e ainda receber respostas do servidor. Por mais que a requisição seja destinada a um servidor, este pode utilizar esta comunicação para acesso em outros, sem que isso seja visível para o cliente.
- **Capacidade de armazenamento:** os serviços da Web RESTful permitem o armazenamento em *cache* como parte de sua arquitetura. O armazenamento em *cache* é um processo pelo qual as respostas de uma API são armazenadas temporariamente em um local próximo ao cliente ou em intermediários, como servidores *proxy*, para reduzir a necessidade de buscar a mesma informação repetidamente do servidor.
- **Código sob demanda:** o servidor pode enviar trechos de código executável para o cliente, que o interpreta e executa localmente. Isso permite que o cliente adquira novas funcionalidades ou comportamentos específicos definidos pelo servidor.

As APIs RESTful possuem diversas vantagens, como simplicidade e facilidade de uso, por utilizar o protocolo HTTP a forma de comunicação já é conhecida e suportada pela maior parte de aplicações que funcionam no contexto do cliente.

Por possuir uma arquitetura que exige *Stateless* ela possui facilidade para escalabili-

dade por não necessitar de armazenar dados entre solicitações. Uma API RESTful permite com que diversas aplicações se integrem a ela, independente da tecnologia, apenas necessitando que a aplicação siga o padrão imposto pela arquitetura.

3.3 Tecnologias *web*

No mundo do desenvolvimento *web*, a escolha das tecnologias corretas desempenha um papel crucial na criação de aplicativos modernos, eficientes e escaláveis. Neste tópico, são exploradas algumas das tecnologias mais populares e poderosas que impulsionam o desenvolvimento de aplicações web.

3.3.1 HTML e CSS

O HTML (*Hyper Text Markup Language*) é uma linguagem de marcação responsável por definir, estruturar e formatar documentos ou páginas da *web* (DUCKETT, 2011). É o componente fundamental utilizado na construção de projetos *web* e é composto por elementos conhecidos como tags. As tags são palavras específicas definidas em HTML, envolvidas por sinais de menor que (<) e maior que (>), geralmente aparecendo em pares para indicar o início e o fim da marcação.

A interpretação desse hipertexto é realizada por um navegador da *web*, que é um *software* responsável por interpretar essas marcações estruturais e construir uma página da *web* com recursos de hipermídia com os quais o usuário pode interagir.

No livro *Desenvolvimento de Software II*, MILETTO e BERTAGNOLLI (2014) destaca o CSS (*Cascading Style Sheets*) como uma linguagem de estilo desenvolvida pelo W3C (*World Wide Web Consortium*) em 1996 e tem como principal objetivo estilizar a aparência (*layout*, cor e fonte) dos vários elementos de um documento que são definidos por uma linguagem de marcação como o HTML. Ela foi criada para que se tenha a separação da estrutura do documento de sua aparência.

3.3.2 Javascript

No livro *An Introduction to HTML and Javascript*, BROOKS (2007) evidencia o Javascript como uma linguagem de programação de alto nível criada no ano de 1995 por Brendan Eich quando trabalhava pela empresa Netscape Communications Corporation. Criado com o objetivo principal de validar formulários HTML, mas com o passar do tempo foi modificado se tornando uma linguagem que disponibiliza uma série de recursos de interface gráfica (tais como botões, campos de entrada e seletores) viabilizando assim a construção de páginas *web* mais interativas e dinâmicas.

O Javascript é *client-side* (processo executado localmente na máquina do cliente), desta maneira fornece às páginas *web* a possibilidade de programação, transformação e processa-

mento de dados enviados e recebidos, interagindo com a marcação e exibição de conteúdo da linguagem HTML e com a estilização desse conteúdo proporcionada pelo CSS nessas páginas.

No entanto, vale ressaltar que o Javascript não se limita apenas a interações de *client-side*. Com o desenvolvimento do Node.js (Subseção 3.3.3), uma plataforma de execução de código Javascript no servidor, o Javascript também pode ser utilizado no lado do servidor, permitindo a construção de aplicações *web* completas. Isso significa que o Javascript é uma linguagem versátil, capaz de lidar tanto com a lógica do lado do cliente quanto do lado do servidor.

3.3.3 Node.js

O Node.js é uma plataforma de tempo de execução de código aberto, construída sobre o motor V8 do Google Chrome, que permite a execução de JavaScript no servidor. Ele oferece um ambiente altamente escalável e eficiente para desenvolver aplicações *web* e serviços de rede (MORAES, 2021).

O "coração"do Node.js é o *loop* de eventos (*event loop*), que é responsável por lidar com eventos e chamadas de retorno (*callbacks*). O *loop* de eventos permite que o Node.js execute tarefas de forma assíncrona, liberando o *thread* principal para lidar com outras tarefas enquanto aguarda a conclusão de operações de I/O. Isso resulta em uma maior capacidade de processamento e escalabilidade do Node.js (NODE.JS, 2023).

O Node.js possui uma arquitetura de *thread* único, o que significa que utiliza apenas um *thread* principal para executar todo o código JavaScript. No entanto, o Node.js usa uma biblioteca chamada libuv, que implementa uma camada de abstração sobre operações de I/O assíncronas, permitindo que o Node.js aproveite recursos de múltiplos *threads* subjacentes ao sistema operacional (MORAES, 2021).

Outro componente importante do Node.js são os módulos, que são blocos de código reutilizáveis que encapsulam funcionalidades específicas. Os módulos no Node.js seguem o padrão CommonJS, que permite a modularização e organização eficiente do código em diferentes arquivos. Além disso, o Node.js possui um sistema de gerenciamento de pacotes chamado npm (*Node Package Manager*), que permite a instalação e o compartilhamento de bibliotecas e dependências de forma fácil e conveniente (MORAES, 2021).

3.3.4 React.js

O React é uma biblioteca JavaScript de código aberto que foi desenvolvida pelo Facebook, atualmente Meta, lançado em 2013. Ela é amplamente utilizada para a construção de interfaces de usuário interativas e eficientes (BANKS; PROCELLO, 2021).

O principal objetivo do React é permitir a criação de componentes reutilizáveis e declarativos, que facilitam a construção de interfaces complexas. Ele utiliza uma abordagem baseada em componentes, onde a interface do usuário é dividida em pequenos elementos independentes chamados de componentes. Cada componente possui seu próprio estado interno e pode ser

renderizado e atualizado de forma independente (BANKS; PROCELLO, 2021).

Uma das principais vantagens do React é sua eficiência no processo de renderização. Ele utiliza um algoritmo chamado "Virtual DOM"(DOM Virtual) para realizar atualizações otimizadas na interface do usuário. O Virtual DOM é uma representação em memória da estrutura da interface do usuário e permite que o React calcule as alterações mínimas necessárias para atualizar a tela de forma eficiente, sem a necessidade de re-renderizar todo o conteúdo.

Outra vantagem do React é sua grande comunidade de desenvolvedores e o ecossistema de bibliotecas e ferramentas que o acompanham. Existem diversas extensões e pacotes adicionais disponíveis que facilitam o desenvolvimento com o React, como o *React Router* para controle de rotas, o Redux para gerenciamento de estado global e o *Styled Components* para estilização de componentes.

O React também é altamente flexível e pode ser utilizado tanto para a criação de aplicações *web* (React.js) quanto para o desenvolvimento de aplicativos móveis (*React Native* - Subseção 3.3.5). Essa capacidade de compartilhar a lógica de negócio e os componentes entre diferentes plataformas é uma grande vantagem para os desenvolvedores, pois reduz o esforço e o tempo necessário para criar e manter aplicativos em múltiplas plataformas (BANKS; PROCELLO, 2021).

3.3.5 React Native

O React Native é um *framework* de desenvolvimento de aplicativos móveis que foi criado pelo Facebook, atualmente Meta, em 2015. Ele permite que os desenvolvedores utilizem a linguagem JavaScript e a biblioteca React para construir aplicativos nativos para iOS e Android.

A principal ideia por trás do React Native é permitir que os desenvolvedores compartilhem uma grande parte do código entre as plataformas, mantendo uma aparência e desempenho nativos. Ao contrário das abordagens tradicionais de desenvolvimento móvel, que exigem o uso de diferentes linguagens e ambientes de desenvolvimento para cada plataforma, o React Native oferece uma solução única e unificada (EISENMAN, 2016). O React Native ainda possui outros *frameworks* de desenvolvimento que visam facilitar mais ainda a portabilidade das aplicações construídas, um exemplo é o Expo - Subsubseção 3.3.6.3.

Uma das maiores vantagens do React Native é a reutilização de código. Grande parte da lógica de negócio e dos componentes de interface do usuário pode ser compartilhada entre as versões iOS e Android de um aplicativo. Isso resulta em um desenvolvimento mais rápido, pois as atualizações e correções de *bugs* podem ser aplicadas de forma consistente em ambas as plataformas.

Além disso, o React Native oferece acesso a componentes nativos por meio de uma camada de abstração. Isso significa que os desenvolvedores podem utilizar os recursos nativos do dispositivo, como câmera, GPS, sensores e notificações, sem sair do ambiente JavaScript. Essa integração nativa permite que os aplicativos desenvolvidos com React Native tenham um desempenho semelhante ao de aplicativos nativos tradicionais (EISENMAN, 2016).

3.3.6 *Frameworks* de desenvolvimento

Um *framework* é um pacote de códigos prontos que podem ser utilizados no desenvolvimento de sites. A proposta de uso dessa ferramenta é aplicar funcionalidades, comandos e estruturas já prontas para garantir qualidade no projeto e produtividade (GAMMA et al., 1994). Partindo deste princípio, abaixo são definidos os *frameworks* que foram utilizados para a implementação da aplicação proposta.

3.3.6.1 Express.js

Conforme adquirido de fontes oficiais (OPENJS FOUNDATION, 2017), o Express.js é um *framework* para a plataforma do Node.js (Subseção 3.3.3), rápido e que tem como objetivo, facilitar o desenvolvimento de aplicações *back-end* ou em conjunto com sistemas de templates, aplicações *full-stack*.

3.3.6.2 Next.js

Next.js é um *framework* React para produção. Oferece funcionalidades como renderização do lado do servidor e geração de sites estáticos para React. É uma ferramenta poderosa para construir interfaces de usuário com melhor desempenho, otimização para SEO e escalabilidade (VERCEL, INC., 2023).

3.3.6.3 Expo

Expo é uma plataforma de código aberto para fazer aplicativos universais para Android, iOS e *web* com JavaScript e React. Facilita o processo de desenvolvimento de um aplicativo com React Native, oferecendo um conjunto de ferramentas e serviços que aceleram o desenvolvimento e a implantação (650 INDUSTRIES, INC., 2023).

3.4 *Amazon Web Services*

Lançada em 2006, a *Amazon Web Services* (AWS) destaca-se como a plataforma de nuvem mais amplamente adotada e abrangente globalmente, disponibilizando mais de 200 serviços completos a partir de seus *datacenters* distribuídos globalmente. Clientes em todo o espectro, desde *startups* em ascensão até as maiores corporações e órgãos governamentais, escolhem a AWS para otimizar custos, alcançar maior agilidade e impulsionar inovações de forma mais ágil (AMAZON WEB SERVICES, INC., 2023b).

A AWS se destaca por sua flexibilidade, permitindo que organizações adaptem seus recursos de computação de acordo com as demandas do negócio, pagando apenas pelos serviços utilizados. A elasticidade da AWS permite que empresas dimensionem seus recursos de maneira dinâmica, garantindo eficiência operacional e redução de custos (AMAZON WEB SERVICES, INC., 2023b).

3.4.1 *Amazon Route 53*

O *Amazon Route 53* é um serviço de Sistema de Nomes de Domínio (DNS) e Registro de Domínio oferecido pela AWS. Seu principal objetivo é simplificar o registro e a gestão de nomes de domínio, além de facilitar o roteamento eficiente do tráfego na internet (AMAZON WEB SERVICES, INC., 2023d).

Os usuários podem registrar novos nomes de domínio diretamente no *Route 53*, tornando o processo de aquisição e administração de domínios mais acessível. Além disso, o serviço fornece funcionalidades robustas de DNS gerenciado, permitindo a criação e administração de zonas hospedadas, registros DNS e resolução de nomes (AMAZON WEB SERVICES, INC., 2023d).

Uma característica fundamental do *Route 53* é a capacidade de roteamento de tráfego, onde os usuários podem configurar regras com base em políticas como ponderação de registros, *failover* e geolocalização. Isso possibilita uma distribuição eficiente do tráfego, melhorando a disponibilidade e o desempenho de aplicativos.

O serviço também monitora a saúde de recursos AWS, como instâncias do EC2, redirecionando automaticamente o tráfego para recursos saudáveis em caso de falha. Sua integração fluida com outros serviços da AWS, como Amazon S3, *Elastic Load Balancing* e *AWS Certificate Manager*, torna-o uma escolha conveniente para construir e gerenciar infraestruturas escaláveis na nuvem.

3.4.2 *Amazon Elastic Compute Cloud* (EC2)

O *Amazon Elastic Compute Cloud* (Amazon EC2) representa um serviço fundamental na oferta da AWS, proporcionando uma solução robusta para a computação em nuvem (AMAZON WEB SERVICES, INC., 2023c). Este serviço permite aos usuários executarem máquinas virtuais, conhecidas como instâncias, de maneira escalável e conforme a demanda. Essa flexibilidade é crucial para atender às variadas necessidades de aplicativos e cargas de trabalho, permitindo que organizações dimensionem seus recursos computacionais de maneira eficiente.

Com o Amazon EC2, os usuários têm à disposição uma ampla gama de opções, podendo escolher entre diferentes tipos de instâncias, sistemas operacionais e configurações. Sendo assim, o serviço pode se adaptar precisamente aos requisitos específicos de seus projetos. A grande vantagem desse serviço reside na sua natureza sob demanda, eliminando a necessidade de investimentos significativos em infraestrutura física. Os usuários pagam apenas pelos recursos que efetivamente utilizam, tornando o EC2 uma escolha econômica para organizações de todos os tamanhos (AMAZON WEB SERVICES, INC., 2023c).

A implementação do Amazon EC2 oferece não apenas escalabilidade, mas também agilidade no provisionamento de recursos. Isso significa que os desenvolvedores podem rapidamente criar, configurar e implantar instâncias para atender às exigências dinâmicas de seus aplicativos. Além disso, a AWS assegura uma infraestrutura altamente confiável e segura para

as instâncias do EC2, respaldando a integridade e a disponibilidade dos dados dos usuários. Em resumo, o Amazon EC2 é uma ferramenta essencial para quem busca uma solução flexível, eficiente e econômica para suas necessidades computacionais na nuvem.

3.5 Banco de dados

Um banco de dados é uma coleção organizada de dados, geralmente armazenados e acessados eletronicamente a partir de um sistema de computador (SILBERSCHATZ; KORTH; SUDARSHAN, 2010). Onde os sistemas de arquivos simples não são suficientes para atender às demandas de armazenamento de dados em grande escala e rápido acesso, os bancos de dados se tornam essenciais. Eles oferecem uma maneira eficiente de armazenar, gerenciar e recuperar informações, garantindo integridade, segurança e consistência dos dados.

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs) são *softwares* que interagem com o usuário, aplicativos e o banco de dados em si para capturar e analisar os dados. Um SGBD faz mais do que apenas armazenar os dados, ele gerencia aspectos como recuperação de dados, segurança, consistência e integridade dos dados (SILBERSCHATZ; KORTH; SUDARSHAN, 2010). Alguns SGBDs populares incluem MySQL (Subseção 3.5.1), PostgreSQL, Oracle Database e Microsoft SQL Server.

3.5.1 MySQL: Gerenciamento de Banco de Dados Relacional

O MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS - *Relational Database Management System*) amplamente utilizado. Ele foi desenvolvido para armazenar, organizar e recuperar grandes volumes de dados de forma eficiente e confiável (ORACLE, INC., 2023).

O MySQL utiliza a linguagem SQL (*Structured Query Language*) como sua principal linguagem de consulta. A SQL é uma linguagem padronizada que permite aos usuários realizarem diversas operações, como inserir, atualizar, consultar e excluir dados em um banco de dados (ORACLE, INC., 2023).

O funcionamento do MySQL baseia-se no modelo relacional, onde os dados são organizados em tabelas com linhas e colunas. Cada tabela representa uma entidade específica, e as colunas definem os diferentes atributos ou características dessa entidade. Os relacionamentos ocorrem entre as tabelas e são estabelecidos por meio de chaves primárias e chaves estrangeiras.

Para utilizar o MySQL, é necessário estabelecer uma conexão com o servidor MySQL, que hospeda o banco de dados. Através dessa conexão, os usuários podem executar comandos SQL para realizar operações no banco de dados, como criar tabelas, inserir dados, consultar informações e atualizar registros.

O MySQL oferece recursos avançados, como suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), controle de acesso e segurança, índices para

otimização de consultas e replicação para alta disponibilidade e tolerância a falhas (ORACLE, INC., 2023).

3.6 *Unified Modeling Language*

A UML (*Unified Modeling Language*) é uma linguagem de modelagem visual amplamente utilizada na engenharia de *software*. Ela oferece uma notação gráfica padronizada que permite representar diferentes aspectos de um sistema de *software*, como sua estrutura, comportamento e interações entre os componentes (BOOCH, 2010).

A UML desempenha um papel fundamental no processo de desenvolvimento de software, proporcionando uma linguagem comum que permite que os membros da equipe se comuniquem de forma eficiente e compartilhem uma compreensão clara do sistema em desenvolvimento. Ela possibilita a criação de diagramas visuais, como diagramas de classe, diagramas de sequência, diagramas de atividade, entre outros, que ajudam a visualizar e entender melhor a arquitetura, o design e a funcionalidade do sistema.

A importância da UML no desenvolvimento de software está relacionada a várias vantagens que ela oferece. Primeiro, a UML promove a comunicação efetiva entre as partes interessadas, permitindo que todos tenham uma compreensão comum do sistema. Isso evita mal-entendidos e ajuda a alinhar as expectativas de todos os envolvidos.

Além disso, a UML auxilia no processo de análise e design do sistema, permitindo que os desenvolvedores identifiquem requisitos, estruturem a lógica do software e definam as relações entre os componentes. Com a UML, é possível modelar o sistema de forma modular e identificar possíveis problemas e melhorias antes mesmo de iniciar a implementação (BOOCH, 2010).

4 DESENVOLVIMENTO DA APLICAÇÃO

Neste capítulo são aprofundadas as técnicas e metodologias aplicadas durante o desenvolvimento da aplicação, sendo dividido entre o módulo da API e a aplicação web voltada para ser a *dashboard* do profissional.

4.1 Solução do *Wearable FIT*

A evolução das aplicações de monitoramento de saúde em *smartwatches* representa um campo significativo de estudo na interseção entre tecnologia *wearable* e cuidados com a saúde. Como abordado na Seção 2.2, enquanto aplicações como *Google Fit* e *Samsung Health* estabeleceram um padrão para o auto-monitoramento de atividades e saúde, o desenvolvimento da aplicação deste trabalho, o *Wearable FIT*, introduz uma nova dimensão na interação digital entre pacientes e profissionais de saúde.

A capacidade do *Wearable FIT* de compartilhar dados com profissionais de saúde permite uma avaliação mais precisa do estado de saúde do paciente e pode contribuir para intervenções mais oportunas e personalizadas. Ao contrário das metas autodirigidas em aplicações padrão, o *Wearable FIT* oferece a possibilidade de metas serem estabelecidas ou ajustadas por profissionais de saúde. Este aspecto pode ser particularmente benéfico para pacientes com condições específicas que requerem um acompanhamento mais cuidadoso.

4.2 Recursos utilizados para o desenvolvimento

Tendo em vista o desenvolvimento do aplicativo, foram utilizadas ferramentas e tecnologias que suprissem as necessidades da aplicação, descritas a seguir.

- **NPM (Node Package Manager)**: Este é o sistema de gerenciamento de pacotes padrão do NodeJS. Foi empregado para instalar todas as dependências requeridas no projeto.
- **PhpMyAdmin**: Utilizado para gerenciamento do banco de dados na nuvem, oferecendo facilidade de manutenção através de sua interface gráfica amigável.
- **OpenSSH**: Ferramenta empregada para estabelecer conexão segura e gerenciar o servidor utilizado no deploy da aplicação.
- **Astah UML**: Utilizado para a criação de diagramas UML, com a ajuda de uma licença institucional, para planejar e estruturar o sistema.
- **Visual Studio Code**: Escolhido por ser um ambiente de desenvolvimento integrado (IDE) gratuito e customizável, foi utilizado para a edição do código fonte do projeto.
- **Overleaf**: Editor LaTeX baseado na nuvem, escolhido para a elaboração de documentos teóricos do projeto.

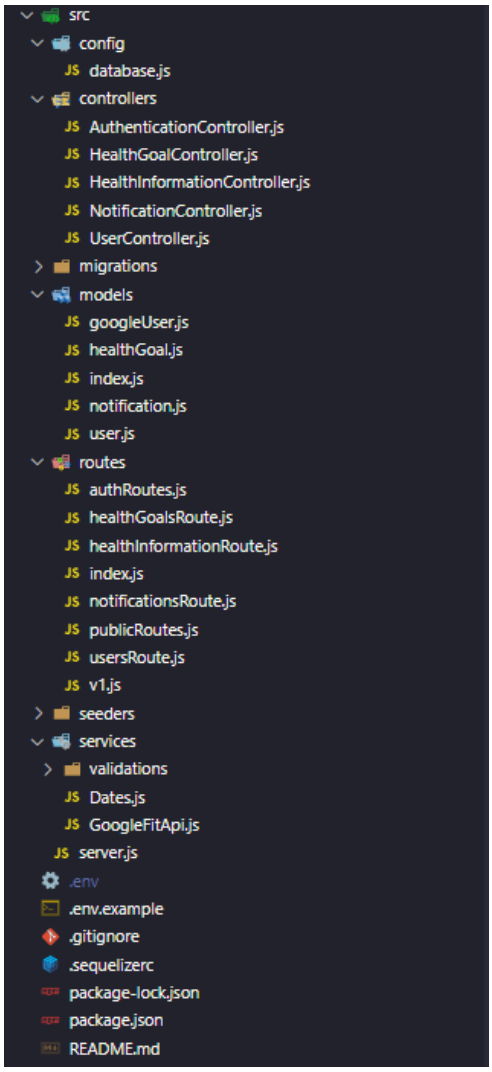
- **Postman**: Interface simples e eficiente para a realização de testes e desenvolvimento de APIs.

4.3 Desenvolvimento da API

A aplicações web necessitam de um centralizador das regras de negócio, este responsável pela manutenção dos dados, acesso ao banco e disponibilização das funcionalidades do sistema. Para isso, foi criada uma API responsável pela lógica e fluxo de dados. O código fonte pode ser encontrado no seguinte repositório remoto: github.com/IgorFollador/wearable-fit-api.

A API foi desenvolvida em NodeJS utilizando o Express.js, que é um *framewrok* minimalista para criação de servidores HTTP. A estrutura proposta para organização do código fonte foi através da divisão de funções como pode ser visualizado na Figura 8.

Figura 8 – Estrutura de arquivos da API

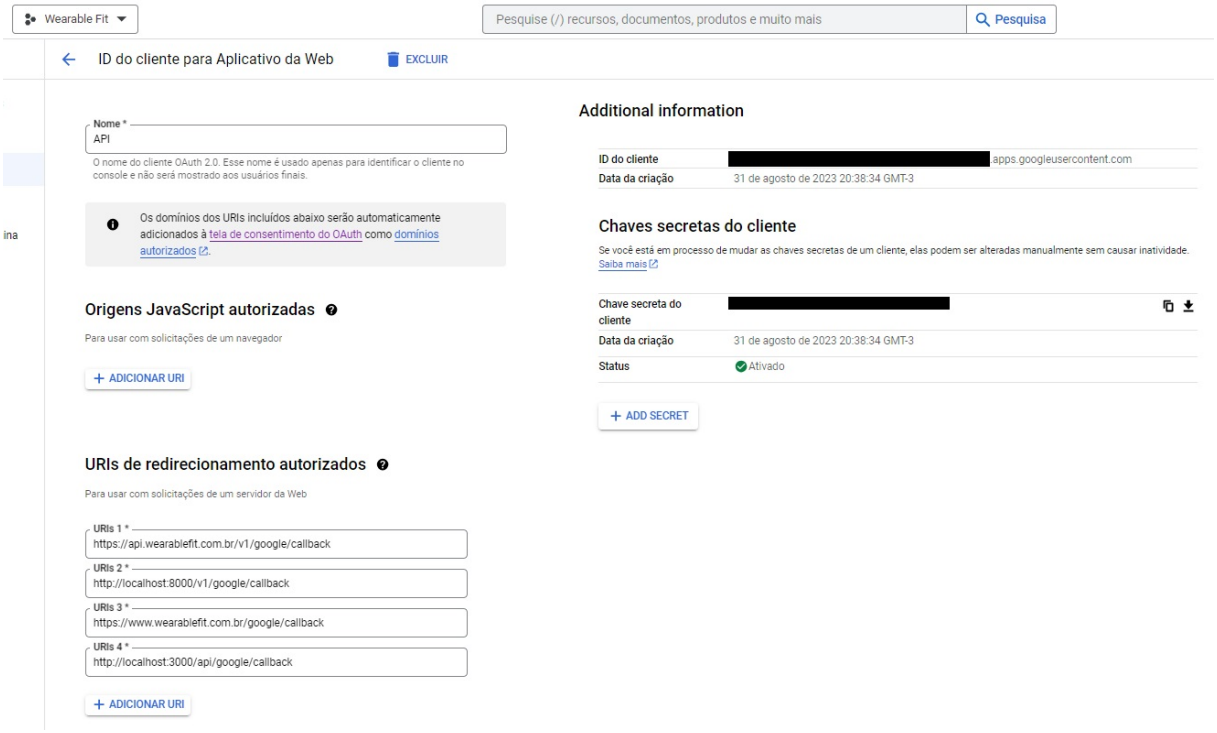


Fonte: Autor

4.3.1 Comunicação com o Google API

Para que o usuário possa compartilhar suas informações com o profissional é necessário que ele autorize a aplicação *Wearable FIT* para ter acesso aos seus dados. Este fluxo é realizado através de uma autorização utilizando *oAuth* do Google. O primeiro passo foi a configuração do projeto dentro da plataforma do Google API Console para ter acesso a um ID de cliente e um token que será utilizado para gerar a URL de autorização da aplicação aos dados do usuário (Figura 9).

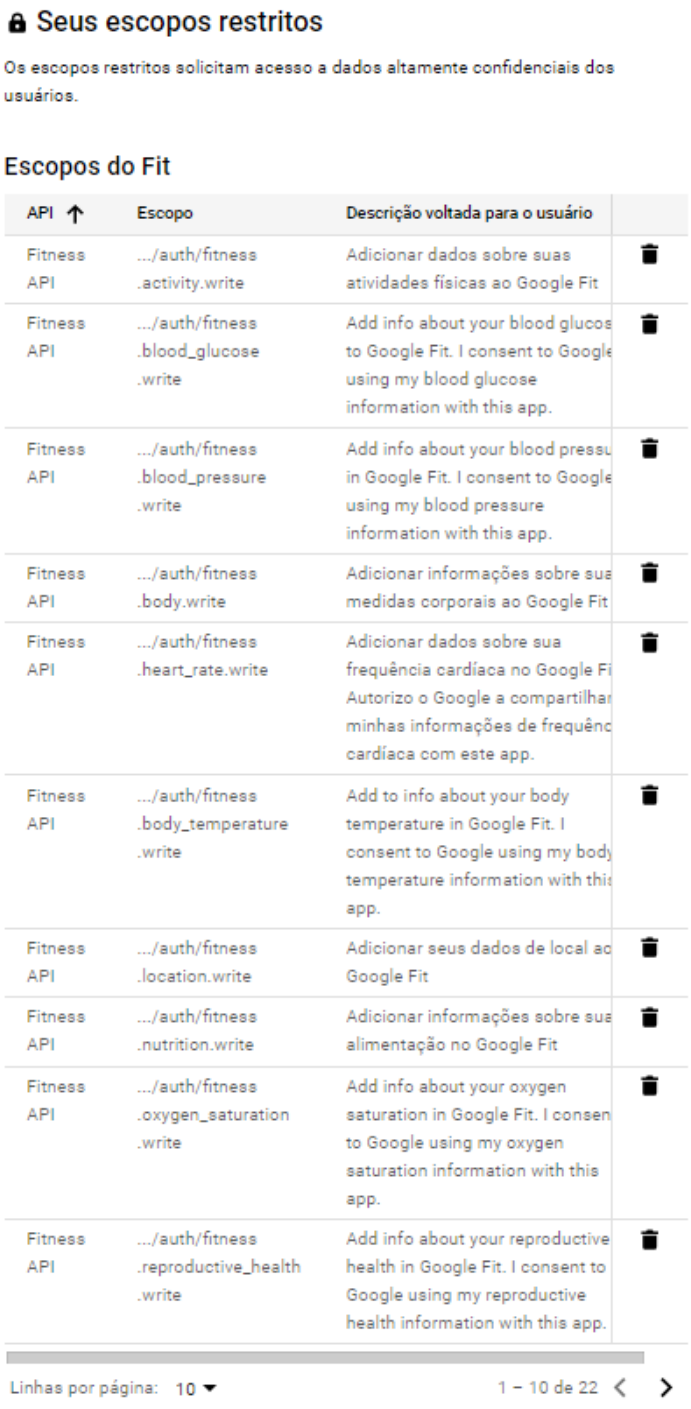
Figura 9 – Criação do projeto no Google API Console



Fonte: Autor

Durante a criação do projeto no Google API Console, é necessário definir os escopos de acesso que o projeto deseja acessar na conta do usuário que permitirá o acesso. Os escopos escolhidos foram unicamente os do Google FIT API, como pode ser visualizado na Figura 10.

Figura 10 – Escopos do projeto no Google API Console



Seus escopos restritos

Os escopos restritos solicitam acesso a dados altamente confidenciais dos usuários.

Escopos do Fit

API	Escopo	Descrição voltada para o usuário	
Fitness API	.../auth/fitness .activity.write	Adicionar dados sobre suas atividades físicas ao Google Fit	
Fitness API	.../auth/fitness .blood_glucose .write	Add info about your blood glucos to Google Fit. I consent to Google using my blood glucose information with this app.	
Fitness API	.../auth/fitness .blood_pressure .write	Add info about your blood pressu in Google Fit. I consent to Google using my blood pressure information with this app.	
Fitness API	.../auth/fitness .body.write	Adicionar informações sobre sua medidas corporais ao Google Fit	
Fitness API	.../auth/fitness .heart_rate.write	Adicionar dados sobre sua frequência cardíaca no Google Fi Autorizo o Google a compartilhar minhas informações de frequênc cardíaca com este app.	
Fitness API	.../auth/fitness .body_temperature .write	Add to info about your body temperature in Google Fit. I consent to Google using my body temperature information with this app.	
Fitness API	.../auth/fitness .location.write	Adicionar seus dados de local ao Google Fit	
Fitness API	.../auth/fitness .nutrition.write	Adicionar informações sobre sua alimentação no Google Fit	
Fitness API	.../auth/fitness .oxygen_saturation .write	Add info about your oxygen saturation in Google Fit. I consen to Google using my oxygen saturation information with this app.	
Fitness API	.../auth/fitness .reproductive_health .write	Add info about your reproductive health in Google Fit. I consent to Google using my reproductive health information with this app.	

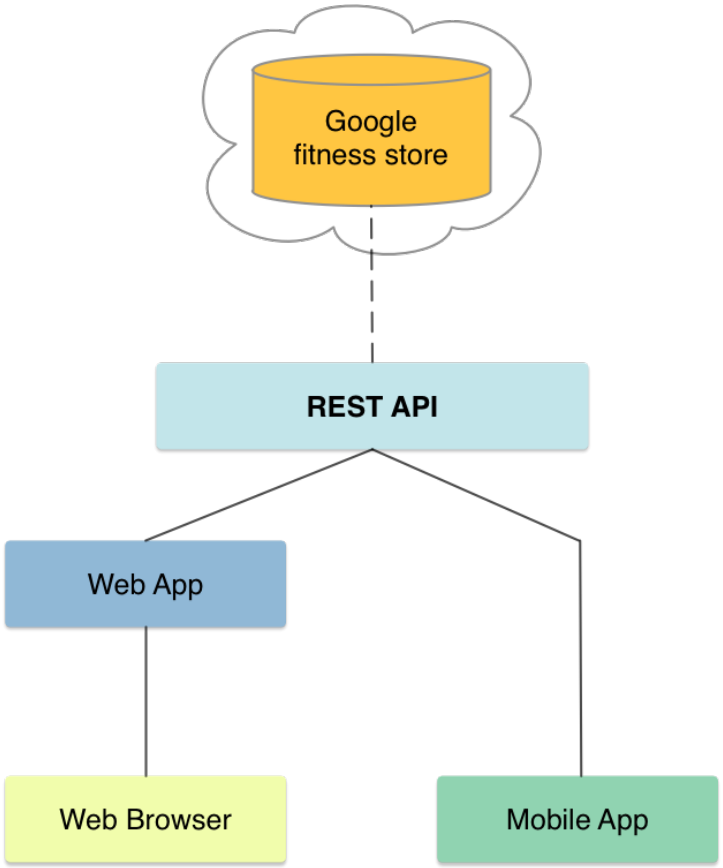
Linhas por página: 10 1 – 10 de 22

Fonte: Autor

4.3.1.1 Integração com o serviço do *Google FIT API*

O processo detalhado de autorização, para o OAuth 2.0, depende do tipo de aplicativo que está buscando o acesso (Figura 11).

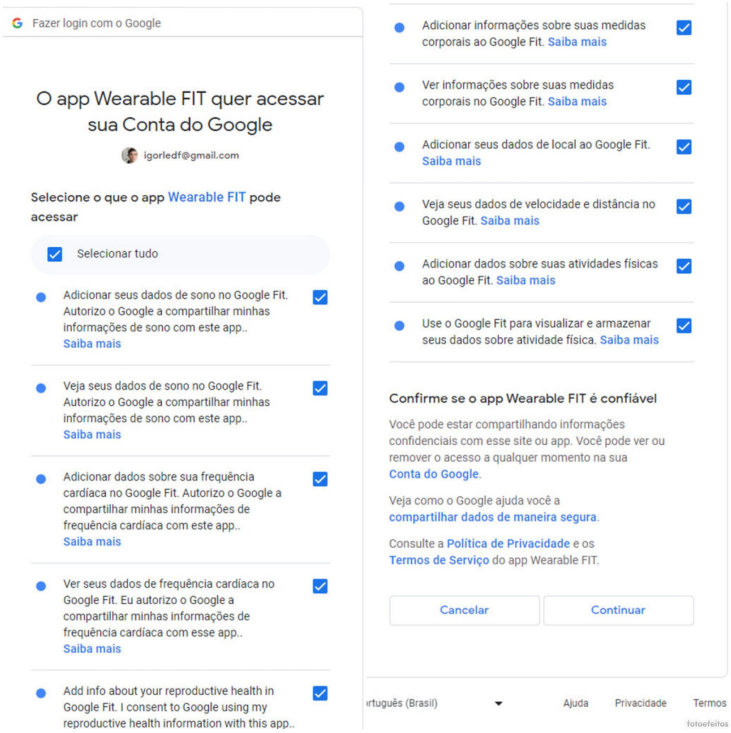
Figura 11 – *Google FIT Rest API*



Fonte: (GOOGLE, INC., 2023)

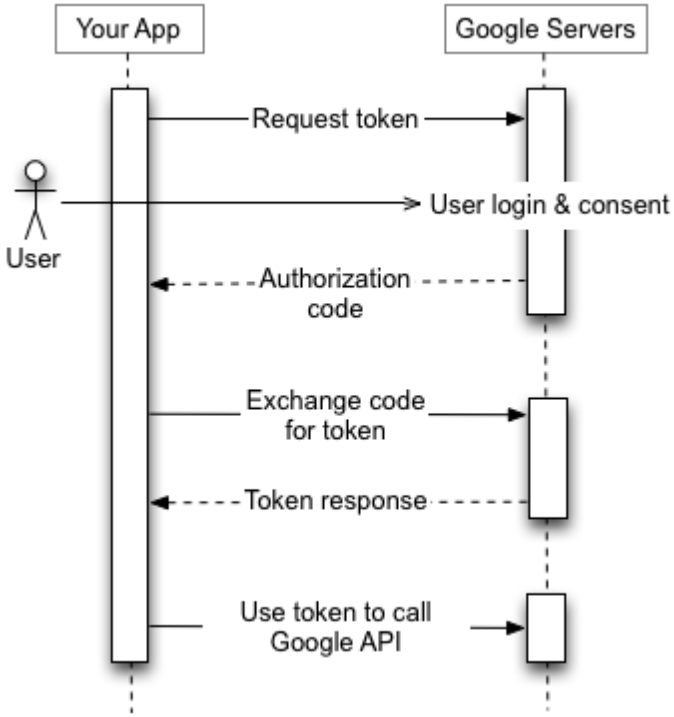
Através do *Client ID* e *Secret Key* fornecidos na criação da aplicação no Google API Console, é possivel gerar uma URL de autorização para qual o usuário da aplicação *Wearable FIT* é redirecionado. Na página mostrada na Figura 12 o usuário libera o consentimento na aplicação e com isso é disparado um *callback* para a URL configurada onde a aplicação recebe credenciais para acessar futuramente os dados do usuário (Figura 13).

Figura 12 – Consentimento Google API



Fonte: Autor

Figura 13 – Autenticação Google API



Fonte: (GOOGLE, INC., 2023)

Com os acessos, é possível então requisitar dados de *data sources*, *data sets* e *sessions*, respeitando os escopos liberados. Na Figura 14 é possível evidenciar a criação da classe responsável por servir as funções de comunicação com o Google API.

Foram criadas duas funções principais para consulta de *data sources* e *data sets*, respectivamente (Figura 15). A função assíncrona, *fetchDataByDataType*, se destina a recuperar dados baseados em um tipo específico (*dataTypeName*) e data. Primeiro, a data é ajustada para o início do dia. A função então constrói um objeto de solicitação que inclui o ID do usuário, as credenciais de autenticação, e um objeto *resource* especificando como os dados devem ser agregados.

O *resource* define o tipo de dado a ser agregado, um intervalo de tempo para agrupamento (neste caso, 30 minutos) e os limites de tempo (início e fim do dia baseados na data fornecida). Após enviar a solicitação para a API do *Google FIT*, a função processa a resposta para extrair e retornar os pontos de dados relevantes.

Figura 14 – Classe GoogleFitApi.js

```
const { google } = require('googleapis');

You, 1 second ago | 2 authors (You and others)
class GoogleFitApi {
    constructor(redirectUrl = `${process.env.HOST}/google/authCallback`) {
        this.fitness = google.fitness('v1');

        this.oAuthClient = new google.auth.OAuth2(
            process.env.GOOGLE_CLIENT_ID,
            process.env.GOOGLE_CLIENT_SECRET,
            redirectUrl
        );

        this.scopes = [
            'https://www.googleapis.com/auth/fitness.activity.read',
            'https://www.googleapis.com/auth/fitness.activity.write',
            'https://www.googleapis.com/auth/fitness.blood_glucose.read',
            'https://www.googleapis.com/auth/fitness.blood_glucose.write',
            'https://www.googleapis.com/auth/fitness.blood_pressure.read',
            'https://www.googleapis.com/auth/fitness.blood_pressure.write',
            'https://www.googleapis.com/auth/fitness.body.read',
            'https://www.googleapis.com/auth/fitness.body.write',
            'https://www.googleapis.com/auth/fitness.body_temperature.read',
            'https://www.googleapis.com/auth/fitness.body_temperature.write',
            'https://www.googleapis.com/auth/fitness.heart_rate.read',
            'https://www.googleapis.com/auth/fitness.heart_rate.write',
            'https://www.googleapis.com/auth/fitness.location.read',
            'https://www.googleapis.com/auth/fitness.location.write',
            'https://www.googleapis.com/auth/fitness.nutrition.read',
            'https://www.googleapis.com/auth/fitness.nutrition.write',
            'https://www.googleapis.com/auth/fitness.oxygen_saturation.read',
            'https://www.googleapis.com/auth/fitness.oxygen_saturation.write',
            'https://www.googleapis.com/auth/fitness.reproductive_health.read',
            'https://www.googleapis.com/auth/fitness.reproductive_health.write',
            'https://www.googleapis.com/auth/fitness.sleep.read',
            'https://www.googleapis.com/auth/fitness.sleep.write'
        ];
    }

    setTokens(tokens) {
        this.oAuthClient.setCredentials({
            access_token: tokens.accessToken,
            refresh_token: tokens.refreshToken
        });
    }
```

Fonte: Autor

Figura 15 – Funções auxiliares de requisição

```
async getFitnessDataByDataSourceId(dataSourceId) {
    const response = await this.fitness.users.dataSources.get({
        userId: 'me',
        dataSourceId: dataSourceId,
        auth: this.oAuthClient
    });

    return response;
}

async fetchDataByDataType(dataTypeName, date) {
    // Ajuste a data para o início do dia.
    date.setHours(0, 0, 0, 0);

    const request = {
        userId: 'me',
        auth: this.oAuthClient,
        resource: {
            aggregateBy: [{ dataTypeName }],
            bucketByTime: { durationMillis: 1800000 }, // Agrupe por 30 minutos
            startTimeMillis: date.getTime(),
            endTimeMillis: date.getTime() + 86400000 // Fim do mesmo dia
        }
    };

    try {
        const response = await this.fitness.users.dataset.aggregate(request);
        const dataPoints = [];

        if (response.data && response.data.bucket) {
            response.data.bucket.forEach(bucket => {
                if (bucket.dataset[0].point) {
                    bucket.dataset[0].point.forEach(point => {
                        dataPoints.push(point);
                    });
                }
            });
        }

        return dataPoints;
    } catch (error) {
        console.error(`Erro ao recuperar dados de ${dataTypeName}:`, error);
        throw error;
    }
}
```

Fonte: Autor

Através das funções definidas anteriormente foi possível criar funções auxiliares para retornar diretamente os dados de retorno de atividades realizadas, calorias ativas gastas e número de passos, conforme Figura 16.

Figura 16 – Funções de contagem de passos, calorias queimadas e tempo de atividade física

```
async getDailyStepCount(date) {
    const dataTypeName = 'com.google.step_count.delta';
    const dataPoints = await this.fetchDataByDataType(dataTypeName, date);
    return  dataPoints.reduce((total, point) => total + point.value[0].intVal, 0);
}

async getDailyCaloriesBurned(date) {
    const dataTypeName = 'com.google.calories.expended';
    const dataPoints = await this.fetchDataByDataType(dataTypeName, date);
    return dataPoints.reduce((total, point) => total + point.value[0].fpVal, 0);
}

async getDailyPhysicalActivityDuration(date) {
    const dataTypeName = 'com.google.activity.segment';
    const dataPoints = await this.fetchDataByDataType(dataTypeName, date);
    let activities = {};

    for (const point of dataPoints) {
        const activityType = point.value[0].intVal;
        const duration = (point.endTimeNanos - point.startTimeNanos)  / 1000000000 / 60; // Convertendo para minutos

        if (activities[activityType]) {
            activities[activityType].duration += duration;
        } else {
            activities[activityType] = { duration, name: this.getActivityName(activityType) };
        }
    }

    return activities;
}
```

Fonte: Autor

Algumas informações necessitavam de maiores cuidados como foi o caso da função de retorno de tempo de sono (Figura 17) onde foi necessário utilizar o recurso de *sessions* do *Google FIT* e delimitar um tempo possível para estas sessões.

Para garantir abranger a maior parte de sessões de sono foi escolhido filtrar utilizando como horário inicial as 12 (doze) horas da manhã e abranger um turno de 24 horas, pois nos testes realizados as sessões de sono eram interrompidas caso iniciassem na hora 0 (zero) do mesmo dia, já que na maioria das vezes as pessoas iniciam seu turno de sono horas antes do dia terminar.

Figura 17 – Função de retorno do tempo de sono

```
async getDailySleepDuration(date) {
    // Filtro da data as 12pm até as 12pm do outro dia
    date.setHours(12, 0, 0, 0);
    const startTime = date.getTime();
    const endTime = startTime + 86400000 ; // Fim do mesmo dia + 24h
    let totalSleepDuration = 0;

    try {
        const sessionsResponse = await this.fitness.users.sessions.list({
            userId: 'me',
            auth: this.oAuthClient,
            startTime: new Date(startTime).toISOString(),
            endTime: new Date(endTime).toISOString(),
            activityType: 72 // Tipo de atividade para sono
        });

        if (!sessionsResponse.data.session || sessionsResponse.data.session.length === 0) {
            console.log("Nenhuma sessão de sono encontrada para o dia.");
            return 0;
        }

        const aggregateResponse = await this.fitness.users.dataset.aggregate({
            userId: 'me',
            auth: this.oAuthClient,
            resource: {
                aggregateBy: [
                    {
                        dataTypeName: 'com.google.sleep.segment'
                    }
                ],
                endTimeMillis: sessionsResponse.data.session[0].endTimeMillis,
                startTimeMillis: sessionsResponse.data.session[0].startTimeMillis
            }
        });

        if (!aggregateResponse.data.bucket || aggregateResponse.data.bucket.length === 0) {
            console.log("Nenhum bucket de sono encontrada para o dia.");
            return 0;
        }

        totalSleepDuration = (aggregateResponse.data.bucket[0].endTimeMillis - aggregateResponse.data.bucket[0].startTimeMillis) / 3600000; // converte em horas

        return totalSleepDuration > 0 ? totalSleepDuration : 0;
    } catch (error) {
        console.error('Erro ao recuperar dados de sono:', error);
        throw error;
    }
}
```

Fonte: Autor

Na Figura 18 é possível identificar que foi utilizada uma lógica diferente, onde não é listada uma média dos batimentos cardíacos do usuário e sim seus batimentos coletados de 30 em 30 minutos. Caso não sejam encontrados dados de monitoramento em um determinado horário, o mesmo é omitido.

Figura 18 – Função de retorno de batimentos cardíacos

```
async getDailyHeartRate(date) {
    const dataTypeName = 'com.google.heart_rate.bpm';
    const dataPoints = await this.fetchDataByDataType(dataTypeName, date);

    const heartRates = {};
    for (let hour = 0; hour < 24; hour++) {
        for (let minute = 0; minute < 60; minute += 30) {
            const timeKey = `${hour.toString().padStart(2, '0')}:${minute.toString().padStart(2, '0')}`;
            heartRates[timeKey] = [];
        }
    }

    dataPoints.forEach(point => {
        const timestamp = new Date(point.startTimeNanos / 1000000);
        const hour = timestamp.getHours();
        const minute = timestamp.getMinutes() - (timestamp.getMinutes() % 30);
        const timeKey = `${hour.toString().padStart(2, '0')}:${minute.toString().padStart(2, '0')}`;
        heartRates[timeKey].push(point.value[0].fpVal);
    });

    // Calcular a média de frequência cardíaca para cada intervalo
    Object.keys(heartRates).forEach(timeKey => {
        const rates = heartRates[timeKey];
        if (rates.length > 0) {
            const averageRate = rates.reduce((sum, rate) => sum + rate, 0) / rates.length;
            heartRates[timeKey] = Math.round(averageRate); // Arredondando o valor
        } else {
            delete heartRates[timeKey]; // Removendo intervalos sem dados
        }
    });

    return heartRates;
}
```

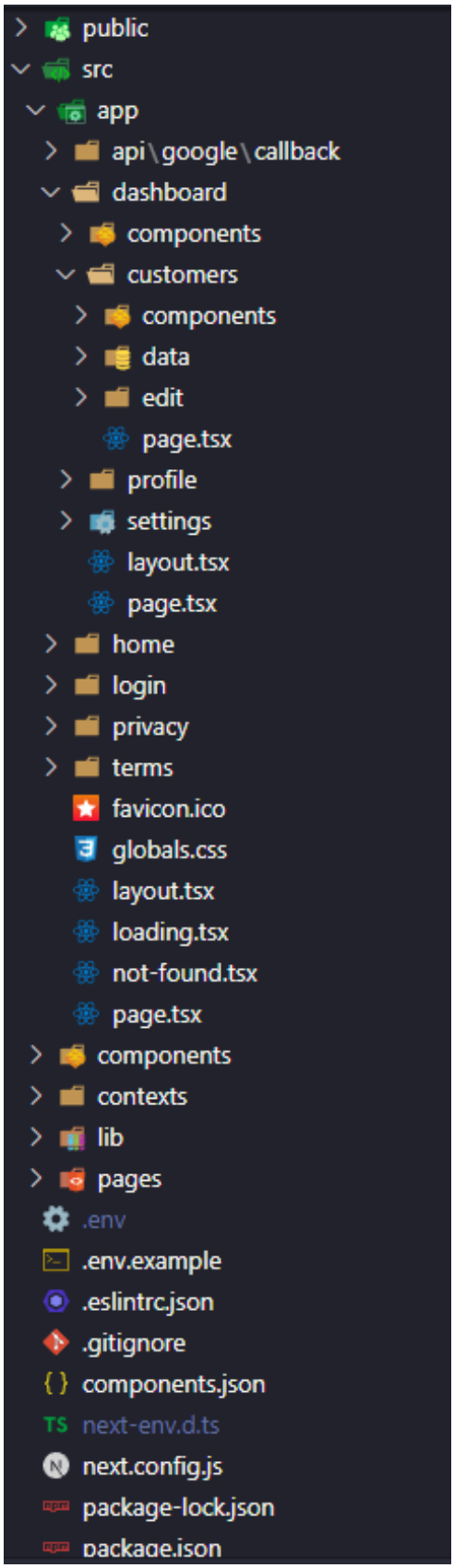
Fonte: Autor

4.4 Desenvolvimento da *dashboard*

No desenvolvimento da *dashboard web* foi utilizado o *framework Next.js* (Figura 19). Este possibilitou um desenvolvimento mais rápido das telas através da montagem de rotas por pastas (disponível na versão 13) do *framework*.

Com o auxilio de bibliotecas de componentes como *shadcn/ui* e *radix/ui* o desenvolvimento se tornou mais ágil e eficiente. O código fonte pode ser encontrado no seguinte repositório remoto: github.com/IgorFollador/wearable-fit-dashboard.

Figura 19 – Estrutura de arquivos da *dashboard*



Fonte: Autor

4.4.1 Requisitos do sistema

A partir das especificações definidas para o sistema, foram levantados os principais requisitos que constituem o desenvolvimento do aplicativo. Nesta seção são abordados os requisitos como forma de visualização geral do funcionamento do APP.

- **Vincular pacientes:** é responsável pelo vinculo de outros usuários ao profissional;
- **Manter metas:** é o principal módulo da aplicação, neste o profissional pode realizar a alteração das metas de cada paciente;
- **Visualizar métricas:** possibilita que tanto o paciente quanto o profissional possa visualizar as informações de saúde monitoradas pelo *smartwatch*;
- **Manter noticiações:** módulos no qual o profissional pode visualizar notificações populadas ou criar notificações para o paciente;

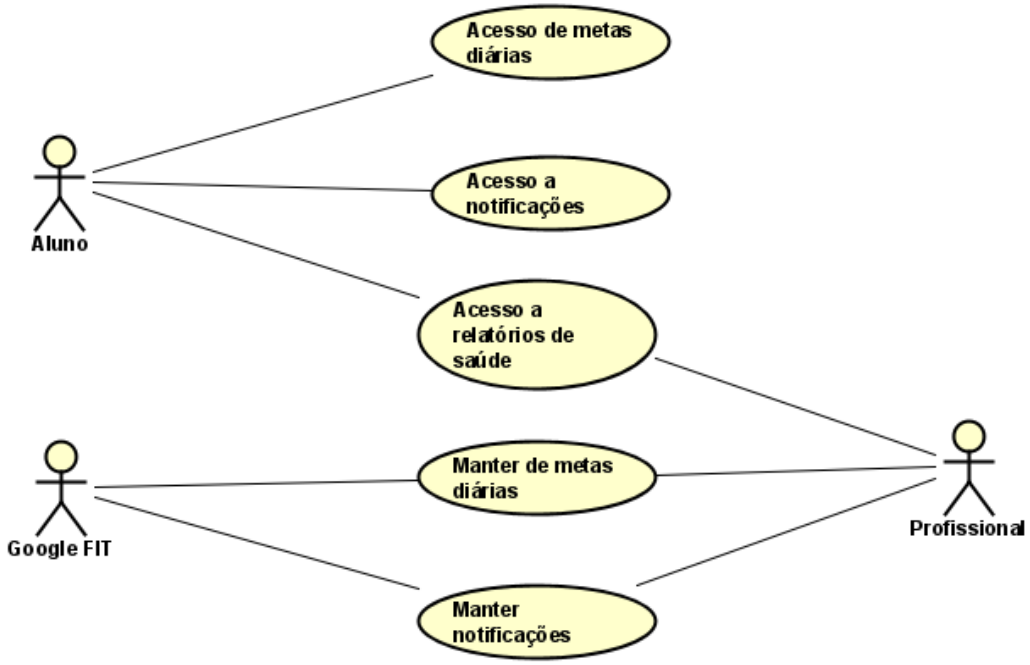
4.5 Diagramas UML

Nesta seção são abordados alguns dos diagramas UML mais importantes para o desenvolvimento da aplicação. Todos foram gerados utilizando a ferramenta Astah UML.

4.5.1 Diagrama de Caso de Uso

O diagrama de caso de uso, representado nesta seção, identifica as diferentes interações existentes entre utilizadores e o sistema. O objetivo principal do diagrama é demonstrar os diferentes serviços presentes na aplicação. Na aplicação desenvolvida é possível identificar três atores, como pode ser evidenciado na Figura 20, onde é vemos os atores Profissional e *Google FIT* como os únicos capazes de realizar ações de escrita e manipulação do sistema. Já o ator Aluno apenas com funções de leitura.

Figura 20 – Diagrama de Caso de Uso

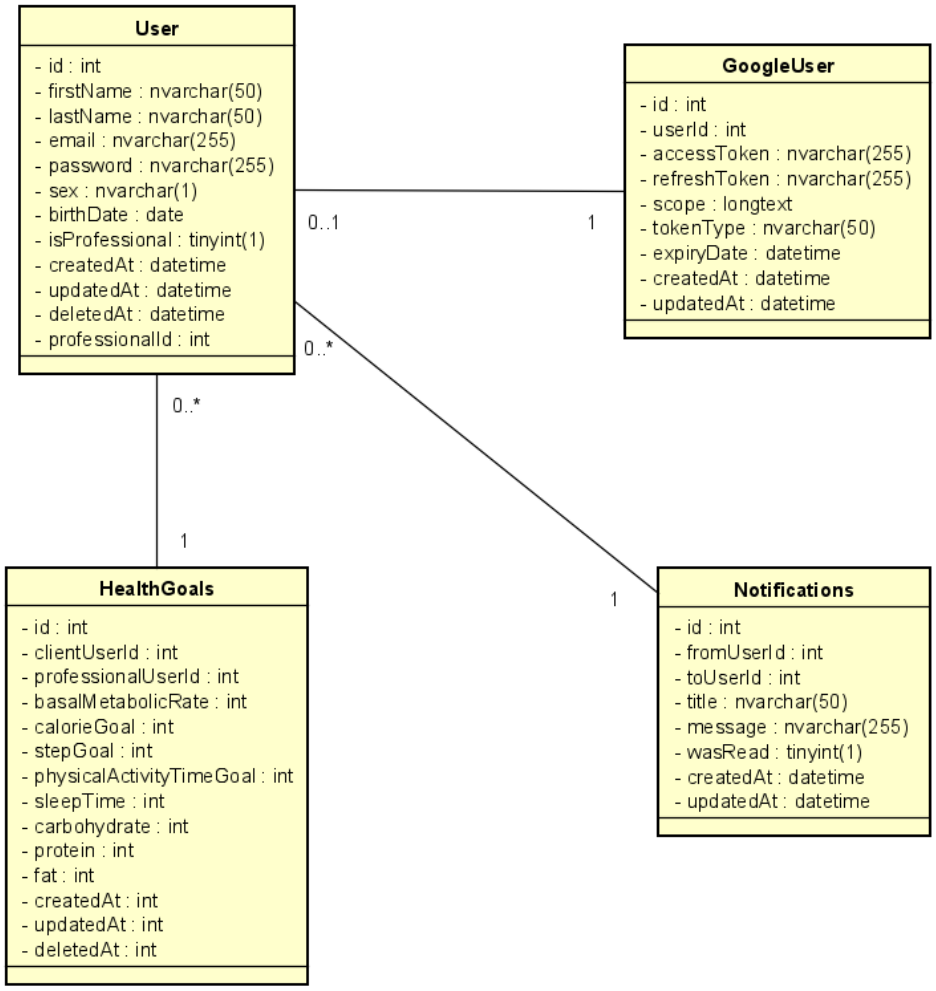


Fonte: Autor

4.5.2 Diagrama de Classes

Considerado o mais importante e frequentemente usado na UML, o diagrama de classes é fundamental para a maioria dos outros diagramas UML. Ele estabelece a estrutura das classes empregadas no sistema, detalhando os atributos e métodos específicos de cada classe. Além disso, ele delineia as relações entre as classes e como elas interagem e compartilham informações umas com as outras (BOOCH, 2010).

Figura 21 – Diagrama de Sequência - Relatório de saúde

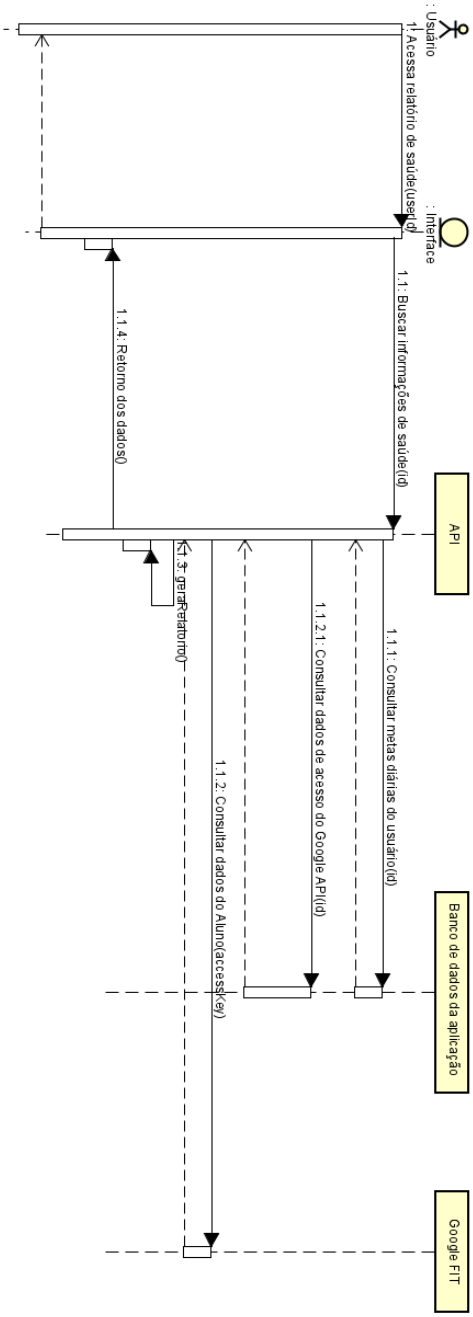


Fonte: Autor

4.5.3 Diagrama de Sequência

O diagrama de sequência da Figura 22 mostra a sequência de mensagens transmitidas entre objetos para obtenção das métricas de saúde obtidas do *Google FIT API*.

Figura 22 – Diagrama de Sequência - Relatório de saúde

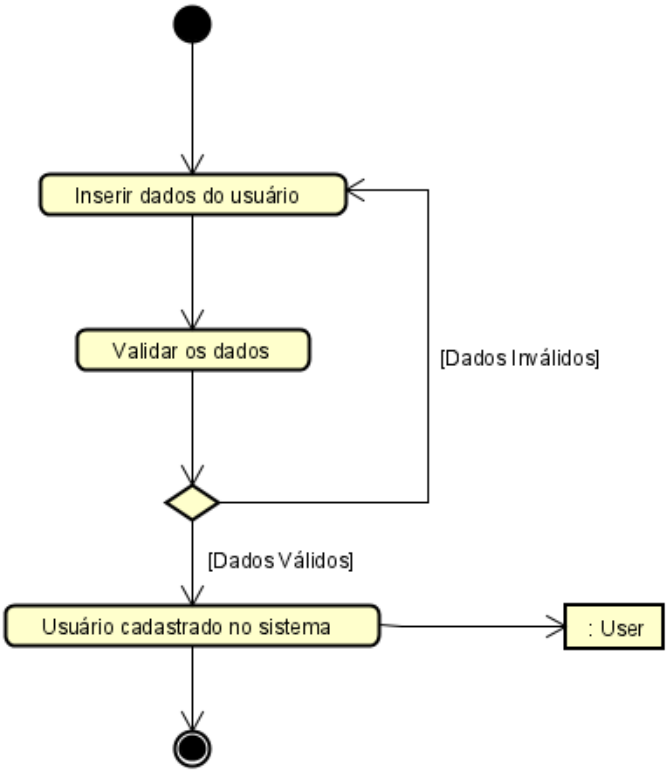


Fonte: Autor

4.5.4 Diagrama de Atividades

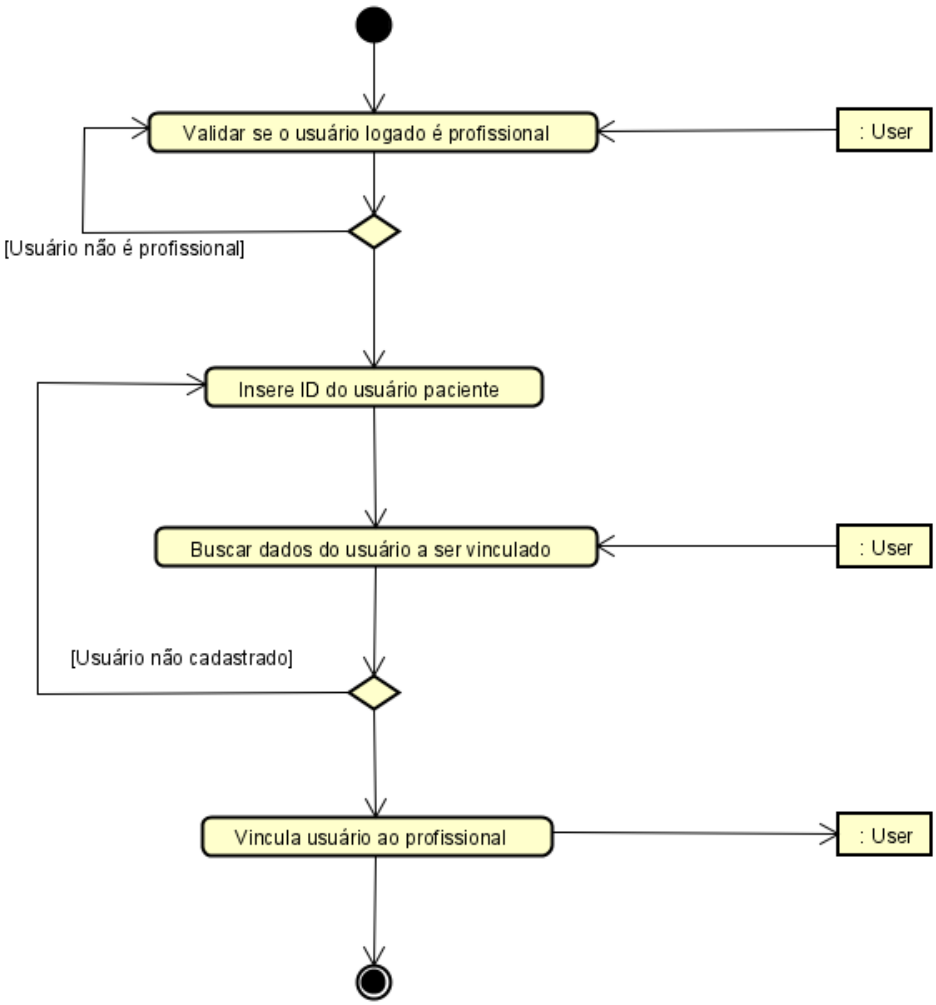
Os diagramas de atividade mostrados na Figura 23 e na Figura 24 representam respectivamente as ações do fluxo de criação de usuários e vínculo de profissionais e alunos.

Figura 23 – Diagrama de Atividades - Criação de usuário



Fonte: Autor

Figura 24 – Diagrama de Atividades - Vínculo aluno profissional



Fonte: Autor

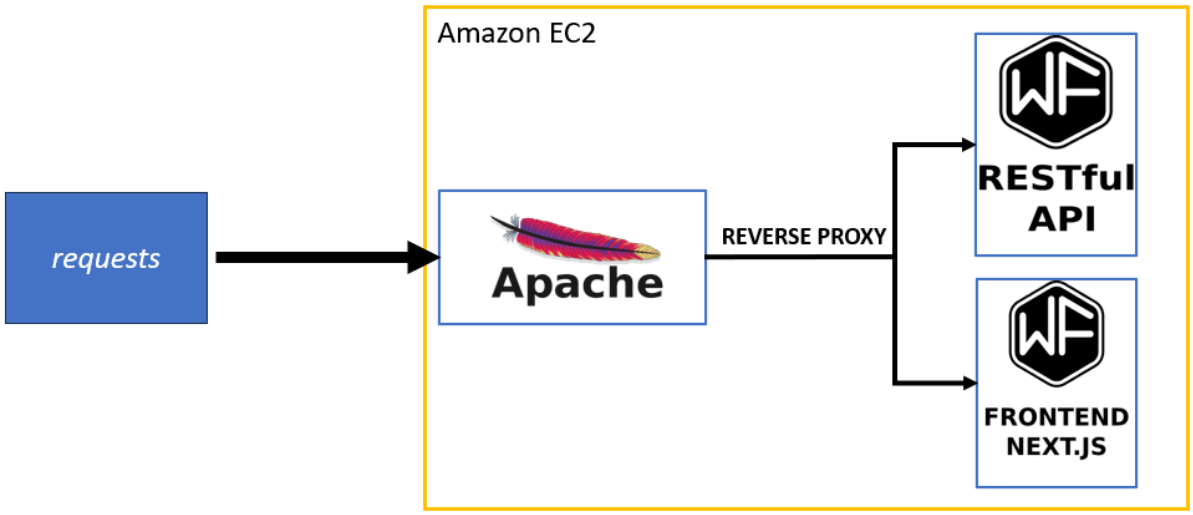
4.6 Ambiente de *deployment*

Utilizando os serviços citados no Seção 3.4, foi utilizada uma estrutura simples (Figura 25) para realizar o *deploy* de todo o sistema.

Endereço de hospedagem da API: api.wearablefit.com.br/v1

Endereço de hospedagem da *dashboard*: www.wearablefit.com.br

Figura 25 – Diagrama do fluxo de requisições no servidor



Fonte: Autor

Para reduzir custos é possível observar que ambos serviços do sistema *Wearable FIT* estão sendo servidos no mesmo servidor. Para isto ser possívell, foi utilizado como servidor HTTP central, o *Apache Server* onde este redireciona as requisições utilizando *virtual hosts* e um *proxy* reverso (Figura 26) para destiná-las para o servidor da API ou para o servidor do *front-end*.

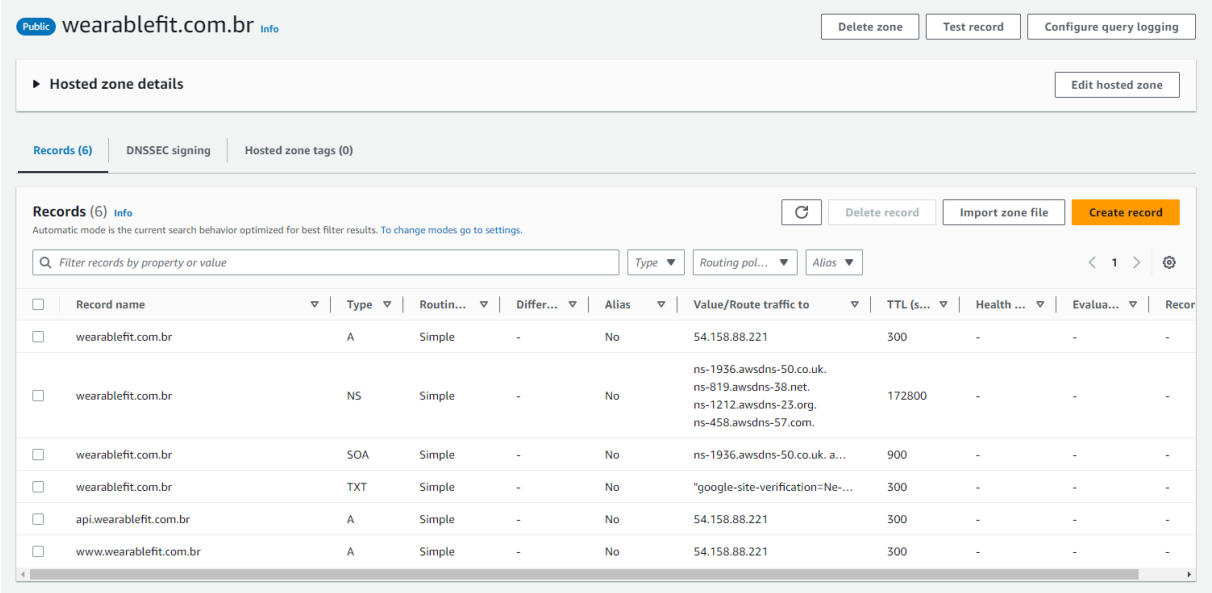
Figura 26 – Configuração do *virtual host* no *Apache Server*

```
<VirtualHost *:80>
  ServerName www.wearablefit.com.br

  ProxyPreserveHost On
  ProxyPass / http://localhost:3000/
  ProxyPassReverse / http://localhost:3000/
  RewriteEngine on
  RewriteCond %{SERVER_NAME} = www.wearablefit.com.br
  RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>

<VirtualHost *:80>
  ServerName api.wearablefit.com.br

  ProxyPreserveHost On
  ProxyPass / http://localhost:8888/
  ProxyPassReverse / http://localhost:8888/
  RewriteEngine on
  RewriteCond %{SERVER_NAME} = api.wearablefit.com.br
  RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Fonte: Autor

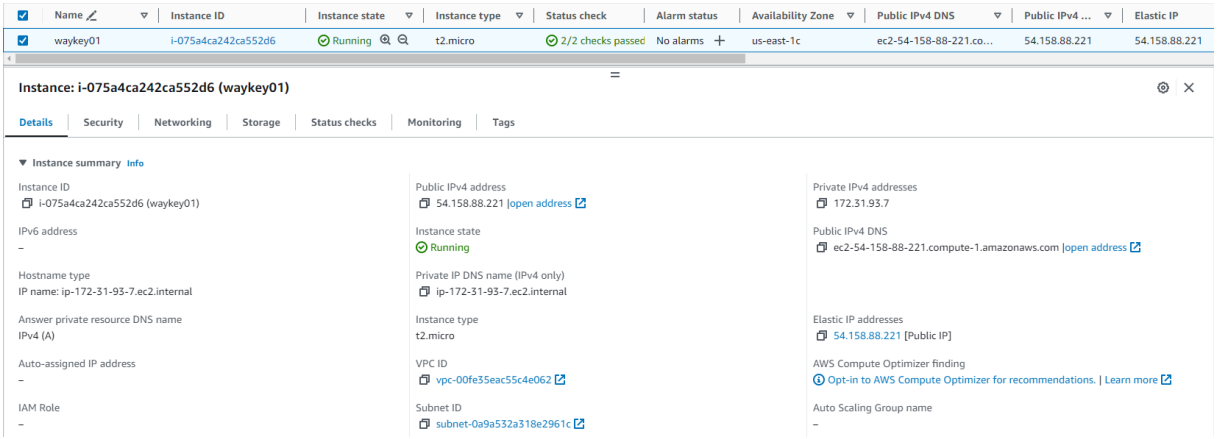
No serviço do *Route 53* foi criada uma zona de DNS do domínio *Wearable FIT* (Figura 27) onde foram criado os registros A *api.wearablefit.com.br* e *www.wearablefit.com.br*, ambos apontando para o mesmo servidor EC2 configurado na AWS, conforme Figura 28.

Figura 27 – Configuração da zona de DNS no *Route 53*



Fonte: Autor

Figura 28 – Configuração do EC2 do tipo t2.micro



Fonte: Autor

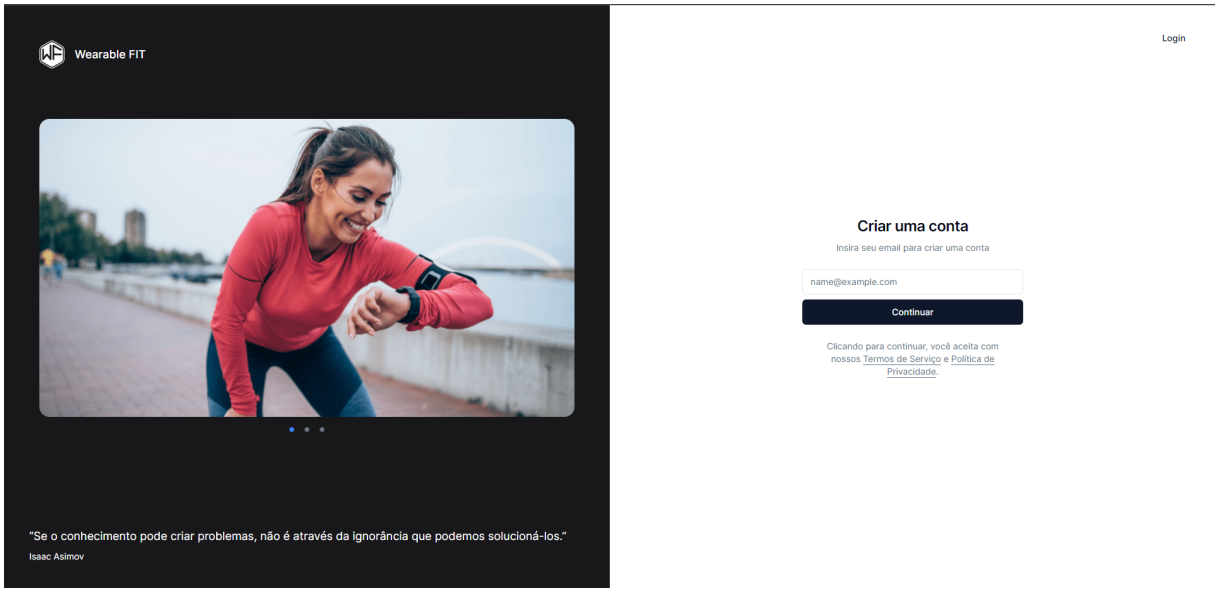
5 DESCRIÇÃO DO SISTEMA

Esta seção explora detalhadamente as interfaces do usuário do sistema, apresentando as telas que compõem a aplicação proposta. Este exame proporcionará uma visão abrangente do *design*, da funcionalidade e da experiência do usuário em cada tela, ilustrando como elas se integram para formar o resultado final da aplicação. São discutidos aspectos como *layout*, elementos de interface, fluxo de navegação e interatividade, oferecendo assim uma compreensão completa de como o aplicativo opera e é experienciado pelos usuários finais.

5.1 Tela *home*

A página inicial da aplicação, mostrada na Figura 29, é composta basicamente por um formulário para cadastro, um botão de acesso à página de autenticação e links para verificação dos termos de serviço e política de privacidade da aplicação.

Figura 29 – Tela *Home* - *Wearable FIT*

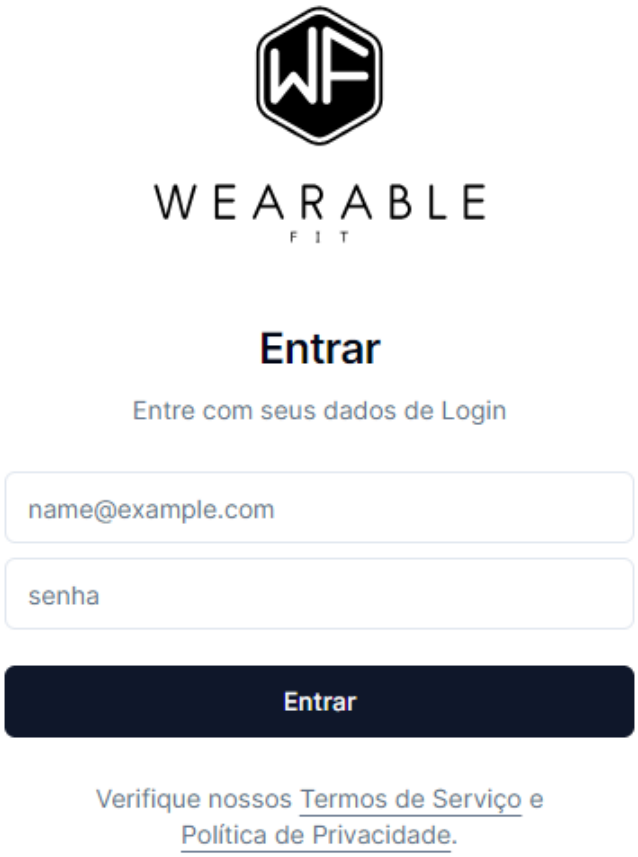


Fonte: Autor

5.2 Tela de autenticação

A Figura 30 é referente à tela de autenticação. Esta tela é obrigatória para se ter acesso ao *dashboard* (caso profissional) e acesso aos dados do aluno.

Figura 30 – Tela de autenticação - *Wearable FIT*



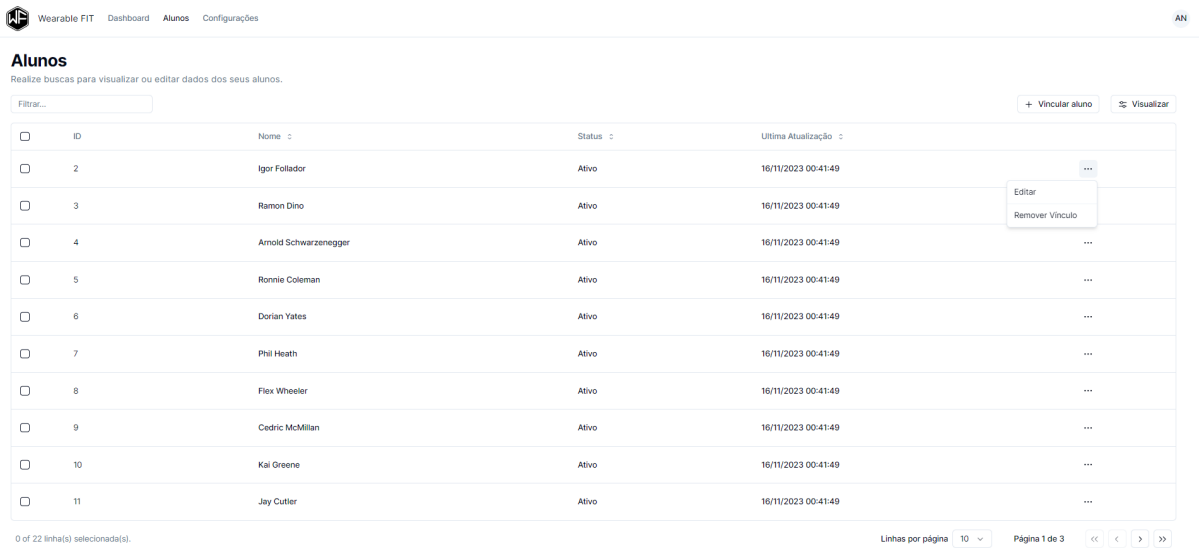
Fonte: Autor

5.3 Tela de listagem de alunos

Na Figura 31 é possível evidenciar a tela de listagem de alunos. Projetada especificamente para facilitar buscas eficientes, esta interface é otimizada para permitir que os profissionais filtrem e encontrem alunos rapidamente, possibilitando o acesso ou a edição de seus dados de forma ágil e intuitiva.

Além disso, a tela também incorpora uma funcionalidade crucial: um botão dedicado para estabelecer vínculos com novos alunos. Este recurso é essencial para expandir e gerenciar a rede de alunos dentro do sistema, garantindo que o profissional possa adicionar novos participantes com facilidade e eficiência.

Figura 31 – Tela de listagem de alunos - *Wearable FIT*



Fonte: Autor

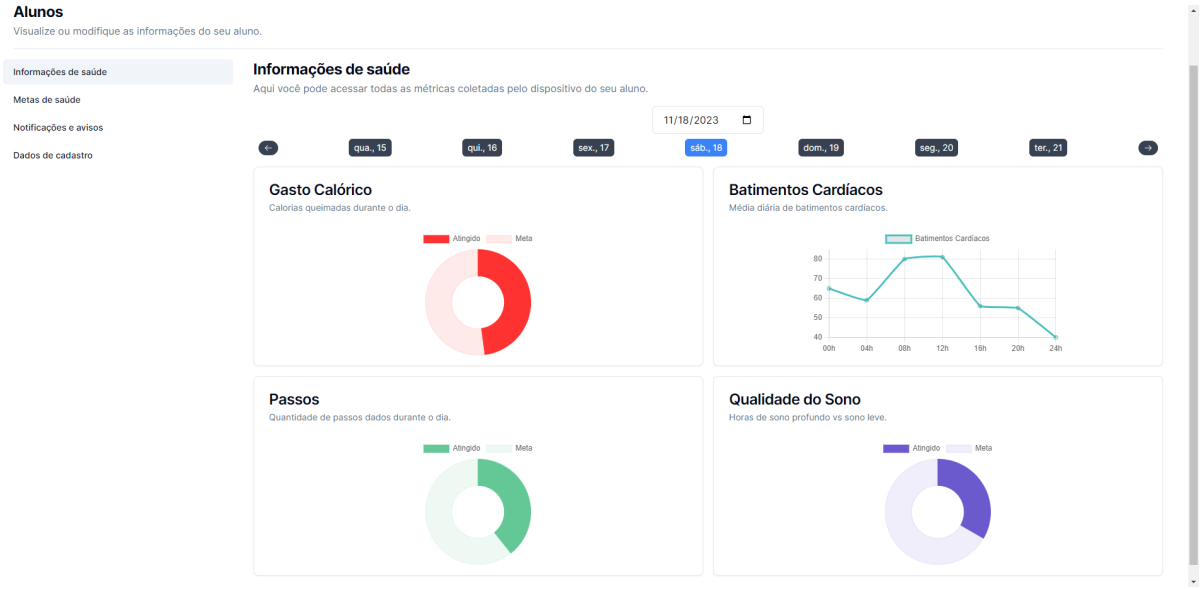
5.4 Informações de saúde do aluno

Na tela de informações de saúde, referenciada como Figura 32, é exibido o relatório detalhado resultante da integração do *Google FIT* com as metas definidas na aplicação. Esta interface é projetada para proporcionar uma análise profunda e personalizada dos dados de saúde do usuário.

Os usuários podem efetuar um filtro por data usando a barra horizontal, que exibe os sete dias mais recentes, ou através de um componente de calendário para seleção de uma data específica. Isso permite o acesso a informações detalhadas sobre o desempenho do usuário em uma data particular, incluindo o total de calorias queimadas, número de passos dados, horas de sono registradas e a frequência cardíaca monitorada.

Esses dados são cruciais para que os profissionais e os usuários possam acompanhar o progresso em relação às metas de saúde e bem-estar estabelecidas, oferecendo uma visão clara e abrangente do estado físico e da atividade do usuário.

Figura 32 – Tela de informações de saúde - *Wearable FIT*

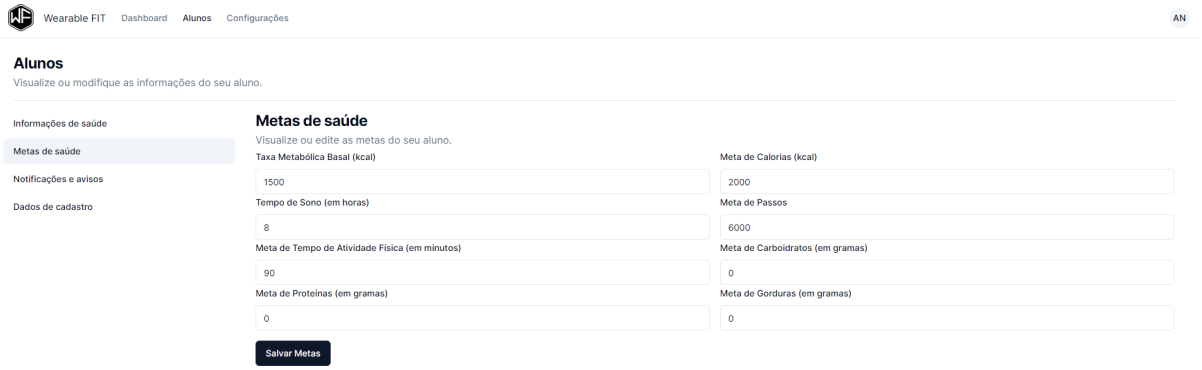


Fonte: Autor

5.5 Cadastro de metas

O recurso de cadastro de metas representa o principal diferencial da aplicação *Wearable FIT*. Na Figura 33 é exibido o formulário de edição de metas.

Figura 33 – Tela de edição de metas de saúde - *Wearable FIT*

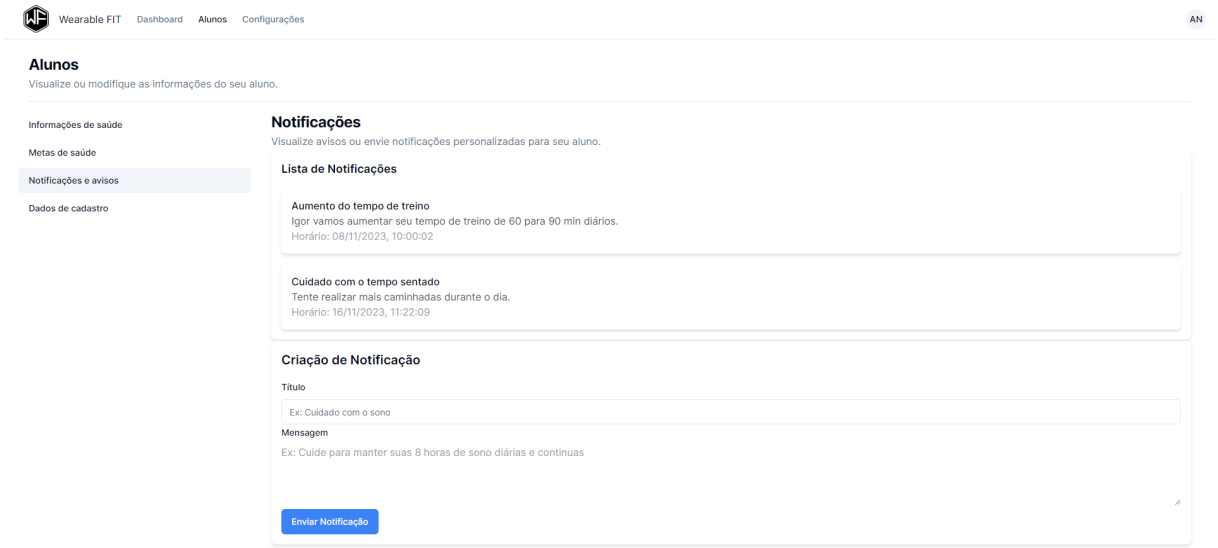


Fonte: Autor

5.6 Cadastro e visualização de notificações

Na Figura 34 é destacada a funcionalidade essencial da aplicação *Wearable FIT*, que permite aos profissionais de saúde enviar notificações diretamente para seus alunos. Esta tela é projetada para facilitar a comunicação entre profissionais e alunos, tornando-a mais direta e eficiente. Por meio desta interface, os profissionais podem rapidamente disparar lembretes, avisos, orientações ou qualquer outro tipo de mensagem importante, garantindo uma interação constante e produtiva.

Figura 34 – Tela de cadastro e visualização de notificações - *Wearable FIT*

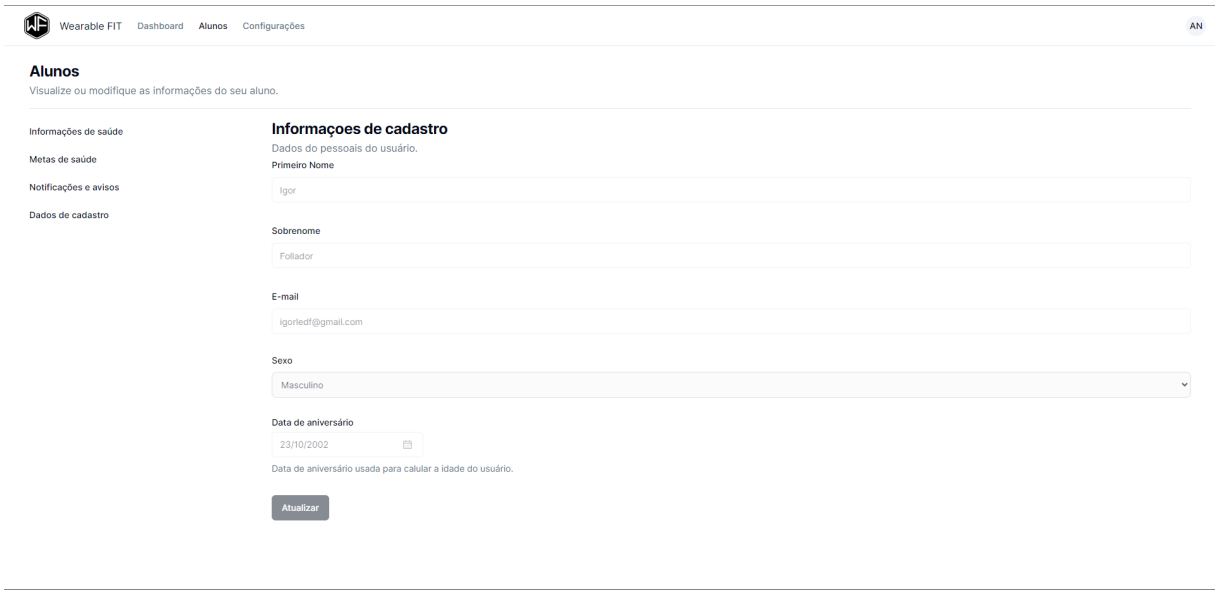


Fonte: Autor

5.7 Visualização dos dados cadastrais

A tela representada na Figura 35 possibilita que os profissionais vejam as informações de cadastro do aluno.

Figura 35 – Tela de visualização dos dados cadastrais - *Wearable FIT*



Fonte: Autor

5.8 Tela da política de privacidade e termos de serviço

Para assegurar a transparência na manipulação dos dados pela aplicação, especialmente considerando que ela acessa informações sensíveis de saúde e permite que esses dados sejam compartilhados com outros usuários, foram desenvolvidas duas telas cruciais: uma para a Política de Privacidade (Figura 36) e outra para os Termos de Serviço (Figura 37).

Estas telas são essenciais, pois os usuários devem concordar com elas ao se registrar na plataforma. Além disso, elas estão diretamente vinculadas à integração com o *Google FIT*, tendo sido registradas durante a criação do projeto no Google API Console. Essa abordagem garante que os usuários estejam plenamente cientes e de acordo com a forma como seus dados são coletados, utilizados e compartilhados, alinhando-se às melhores práticas de privacidade e segurança de dados.

Figura 36 – Tela da política de privacidade - *Wearable FIT*

Política de Privacidade

Esta Política de Privacidade descreve como a aplicação "Wearable Fit" coleta, utiliza e protege informações pessoais dos usuários. Ao utilizar nossa aplicação, você concorda com as práticas descritas nesta política.

1. Informações Coletadas

1.1. **Dados do Google Fit**: A aplicação "Wearable Fit" obtém informações da conta Google conectada, incluindo dados de saúde e fitness do Google Fit API. Esses dados podem incluir informações sobre atividade física, batimentos cardíacos, passos, entre outros.

2. Uso das Informações

2.1. **Fornecimento de Serviços**: Utilizamos os dados do Google Fit para fornecer funcionalidades de acompanhamento de saúde, como monitoramento de atividade física e gerenciamento de metas de saúde.

2.2. **Melhorias na Aplicação**: As informações coletadas nos ajudam a aprimorar nossa aplicação e oferecer uma experiência mais personalizada aos usuários.

2.2.1 **Proteção de Dados**: Nós tomamos medidas para proteger suas informações pessoais, incluindo:
Segurança: Implementamos medidas de segurança para proteger os dados contra acesso não autorizado, alteração ou divulgação.
Acesso Limitado: Apenas funcionários autorizados têm acesso às informações do usuário.

2.2.2 **Compartilhamento de Informações**: Não compartilhamos informações pessoais com terceiros, a menos que seja necessário para o funcionamento da aplicação ou exigido por lei.

2.2.3 **Menores de Idade**: Nossa aplicação não é destinada a menores de 18 anos. Não coletamos intencionalmente informações de menores de idade. Se soubermos que informações pessoais foram coletadas de um menor, as informações serão excluídas.

2.2.4 **Alterações na Política de Privacidade**: Reservamo-nos o direito de modificar esta Política de Privacidade a qualquer momento. Alterações serão comunicadas aos usuários por meio da aplicação.

Home

Fonte: Autor

Figura 37 – Tela dos termos de serviço - *Wearable FIT*

Fonte: Autor

6 CONCLUSÃO

A aplicação desenvolvida neste trabalho possibilitou o compartilhamento de dados de saúde entre profissionais e seus alunos de forma mais direta e rápida. Em combinação com estabelecimento de metas pelo lado do profissional e o sistema de notificações foram abertas as possibilidades de interação entre estes dois atores do sistema, dessa forma os profissionais puderam melhorar seu contato com seus alunos.

Este trabalho usufruiu de diversos conhecimentos da área da computação. Dentre eles, foram aplicados conhecimentos de programação *web*, principalmente na comunicação entre a aplicação utilizada pelo cliente e a API contendo a lógica de negócio usando o protocolo HTTP, em conjunto de conceitos de banco de dados para realizar o armazenamento, recuperação e gerenciamento de informações do sistema.

Os objetivos de criar uma ferramenta que melhora a interação entre dados de saúde e a sua análise prática foram plenamente alcançados, demonstrando a utilidade prática da aplicação no campo da tecnologia em saúde. Ao integrar dados do *Google Fit* e oferecer funcionalidades personalizadas para o estabelecimento de metas e comunicação direta entre profissionais e pacientes, a aplicação demonstrou como a tecnologia pode ser empregada de maneira eficiente e inovadora no setor da saúde.

Muitos obstáculos foram encontrados durante o percurso de desenvolvimento, dentre eles, a falta de transparência de como filtrar os dados proveniente do *Google FIT API*, que, mesmo com a documentação disponível, foi um desafio, e também a utilização do Next.js 13, que instituiu de forma muito mais presente, os *server components* do React, o que foi uma novidade para o autor no desenvolvimento.

Uma direção promissora para trabalhos futuros é a exploração mais detalhada dos dados fornecidos pelo *Google Fit*. Uma análise mais profunda desses dados poderia permitir que os aplicativos os utilizem com maior precisão, alinhando-os aos objetivos propostos e oferecendo *insights* valiosos para os profissionais de saúde. Adicionalmente, a integração com outras APIs, como a do *Samsung Health* ou o desenvolvimento de um aplicativo para o *Wear OS*, poderia expandir as fontes de dados e aumentar o alcance da aplicação.

Outro campo de interesse é a integração com sistemas voltados para nutricionistas e educadores físicos. Isso incluiria a criação de planos alimentares inteligentes, baseados no gasto calórico dos pacientes e ferramentas para que profissionais de educação física monitorem a progressão de treinos de maneira eficiente. Além disso, a criação de uma aplicação mobile para alunos é uma estratégia valiosa, facilitando o acesso a relatórios e dados, semelhante ao que já é oferecido por aplicativos existentes no mercado.

Em resumo, a aplicação desenvolvida demonstrou ser uma ferramenta promissora na área da saúde digital, mostrando como a inovação tecnológica pode ser aplicada para melhorar a qualidade de vida dos pacientes e auxiliar os profissionais de saúde em seu trabalho diário.

REFERÊNCIAS

650 INDUSTRIES, INC. *Expo Documentation*. 2023. <https://docs.expo.dev/>. Acessado em 22/10/2023. Citado na página 28.

AMAZON WEB SERVICES, INC. **O que é API RESTful?** 2023. <https://aws.amazon.com/pt/what-is/restful-api/>. Acessado em 27/05/2023. Citado na página 24.

AMAZON WEB SERVICES, INC. **O que é AWS**. 2023. <https://aws.amazon.com/pt/what-is-aws/>. Acessado em 15/10/2023. Citado na página 28.

AMAZON WEB SERVICES, INC. **What is Amazon EC2?** 2023. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>. Acessado em 15/10/2023. Citado na página 29.

AMAZON WEB SERVICES, INC. **What is Amazon Route 53?** 2023. <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html>. Acessado em 15/10/2023. Citado na página 29.

ARAUJO, D. S. M. S. D. **Aptidão física, saúde e qualidade de vida relacionada à saúde em adultos**. 2010. <https://doi.org/10.1590/S1517-86922000000500005/>. Acessado em 05/11/2023. Citado na página 13.

BANKS, A.; PROCELLO, E. **Learning React: Modern Patterns for Developing React Apps**. [S.l.]: O'Reilly Media, 2021. Citado 2 vezes nas páginas 26 e 27.

BLUETOOTH SIG, INC. **Bluetooth Wireless Technology**. 2023. <https://blog.multisom.com.br/o-que-e-smartwatch/>. Acessado em 21/06/2023. Citado na página 20.

BOCARD, T. **Wearables: o que são as "tecnologias vestíveis"?** 2022. <https://usemobile.com.br/wearable//>. Acessado em 07/05/2023. Citado na página 19.

BOOCH, G. **The Unified Modeling Language User Guide**. [S.l.]: Pearson Education, 2010. Citado 2 vezes nas páginas 31 e 45.

BROOKS, D. R. **An Introduction to HTML and JavaScript: for Scientists and Engineers**. [S.l.]: Springer, 2007. Citado na página 25.

DUCKETT, J. **HTML and CSS: Design and Build Websites**. [S.l.]: Wiley, 2011. Citado na página 25.

EISENMAN, B. **Learning React Native**. [S.l.]: O'Reilly Media, 2016. Citado na página 27.

FIELDING, R. et al. **Hypertext Transfer Protocol – HTTP/1.1**. 1999. <https://datatracker.ietf.org/doc/html/rfc2616>. Acessado em 27/05/2023. Citado 2 vezes nas páginas 22 e 23.

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. [S.l.]: Addison-Wesley, 1994. Citado na página 28.

GASPAR, L. **Protocolo HTTP: entenda o que é e para que serve!** 2021. <https://www.hostgator.com.br/blog/o-que-e-protocolo-http/>. Acessado em 01/07/2023. Citado na página 21.

GOOGLE. **Suporte do Google FIT**. 2023. <https://support.google.com/fit/faq/6108483?hl=pt-BR>. Acessado em 23/09/2023. Citado na página 14.

GOOGLE, INC. **Google Fit - Google for Developers**. 2023. <https://developers.google.com/fit/>. Acessado em 01/06/2023. Citado 3 vezes nas páginas 20, 36 e 37.

MILETTO, M.; BERTAGNOLLI, C. **Desenvolvimento de Software II**. [S.l.]: Bookman, 2014. Citado na página 25.

MORAES, W. B. **Construindo Aplicações com NodeJS**. [S.l.]: Novatec, 2021. Citado na página 26.

MULTISOM. **O que é Smartwatch e como ele pode ajudar na sua rotina**. 2021. <https://blog.multisom.com.br/o-que-e-smartwatch/>. Acessado em 07/06/2023. Citado na página 19.

NODE.JS. **About Node.js**. 2023. <https://nodejs.org/en/about>. Acessado em 21/06/2023. Citado na página 26.

OPENJS FOUNDATION. **Express - Node.js web application framework**. 2017. <https://expressjs.com/>. Acessado em 15/10/2023. Citado na página 28.

ORACLE, INC. **What is MySQL?** 2023. <https://www.oracle.com/mysql/what-is-mysql>. Acessado em 27/06/2023. Citado 2 vezes nas páginas 30 e 31.

RED HAT, INC. **What is an API?** 2022. <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>. Acessado em 27/05/2023. Citado na página 24.

SANTOS, L. **Desenvolvimento Web - Protocolo HTTP**. 2023. <https://www.tabnews.com.br/TraineeCodeplays/desenvolvimento-web-protocolo-http/>. Acessado em 27/05/2023. Citado 3 vezes nas páginas 21, 22 e 23.

SIEDENTOP, D. **Introduction to Physical Education, Fitness, and Sport**. 9. ed. [S.l.]: McGraw-Hill Education, 2019. Citado na página 13.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. [S.l.]: McGraw-Hill, 2010. Citado na página 30.

VERCEL, INC. **Learn Next.js**. 2023. <https://nextjs.org/learn?utm_source=next-site&utm_medium=homepage-cta&utm_campaign=home>. Acessado em 22/10/2023. Citado na página 28.