VIVAFIT SENIOR – Calistenia para terceira idade[1]

Guilherme Antony Oliveira Santos[2]
Rafael Marinho e Silva [3]

RESUMO: Este Artigo teve como objetivo o desenvolvimento de um aplicativo de calistenia voltado para iniciantes da terceira idade, com foco na promoção da saúde física e mental, no incentivo à prática regular de exercícios e na melhoria da qualidade de vida. A proposta surgiu diante do aumento da longevidade e da necessidade de soluções acessíveis que atendam às limitações fisiológicas e motoras típicas desse público. O aplicativo integrou funcionalidades essenciais, como: interface amigável e intuitiva, progressão gradual de exercícios adaptados, monitoramento do progresso individual, personalização de treinos de acordo com idade e condição física, além de recursos motivacionais, como feedbacks positivos e recompensas. O desenvolvimento adotou metodologias ágeis, utilizoundo tecnologias modernas como *React Native* e *Supabase*, além de análise comparativa com outros aplicativos do mercado para garantir acessibilidade, segurança e eficácia. A pesquisa utilizou abordagem mista: qualitativa, pela análise de mercado e as funcionalidades oferecidas; e quantitativa, por meio de indicadores de adesão e desempenho físico. A solução buscou favorecer a autonomia, reduzir riscos de lesões, estimular hábitos saudáveis e proporcionar uma ferramenta tecnológica inclusiva para a prática de exercícios na terceira idade.

PALAVRAS-CHAVE: Calistenia; Terceira Idade; Aplicativo; Saúde; Acessibilidade; Tecnologia.

1 INTRODUÇÃO

O aumento da longevidade e a busca por qualidade de vida têm levado a um crescente interesse por atividades físicas que atendam às necessidades da população idosa. Entre as opções de exercício, a calistenia tem se destacado devido à sua abordagem acessível e eficaz, utilizando o peso corporal como resistência. Um aplicativo de calistenia para iniciantes da terceira idade representa uma solução inovadora e personalizada para promover a saúde e o bem-estar desta faixa etária. A proposta de um aplicativo voltado para esse público deveria considerar as especificidades fisiológicas e motoras dos idosos, oferecendo uma experiência segura, gradual e motivadora (Lucas et al., 202; MP, 2025).

---

[1] Artigo apresentado como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação, pelo Centro Universitário de Patos de Minas - UNIPAM.

[2] Graduando do Curso de Sistemas de Informação, do Centro Universitário de Patos de Minas - UNIPAM. guilhermeantony@unipam.edu.br

[3] Professor Orientador Mestre em Ciência da Computação. rafaelmarinho@unipam.edu.br

Para os idosos, a prática de calistenia pode contribuir significativamente para a melhoria da força muscular, equilíbrio, flexibilidade e mobilidade, fatores essenciais para a prevenção de quedas e a manutenção da autonomia, foi fundamental que o aplicativo desenvolvido apresentasse uma interface amigável, de fácil navegação e compreensão, adequando os exercícios às limitações físicas comuns nesta fase da vida, como a osteoporose, a artrite e a perda de massa muscular, a progressão dos exercícios deveria ser gradual, respeitando o ritmo individual de cada usuário, com instruções claras e demonstrações visuais (Lucas et al. MP, 2025).

Foi importante a inclusão de funcionalidades que promovam o monitoramento do progresso do usuário, permitindo que o idoso acompanhe seus avanços e se sinta motivado a continuar (Natarajan, 2024). A personalização do treino, com base em dados como idade, nível de atividade física e condições de saúde, pode otimizar os resultados e garantir que os exercícios sejam adequados ao perfil do usuário. A presença de feedback positivo e recompensas dentro do aplicativo também pode desempenhar um papel importante na adesão e na continuidade do programa de exercícios (MDN, 2024).

Um aplicativo de calistenia direcionado para iniciantes da terceira idade deve ser uma ferramenta acessível, segura e eficiente, capaz de proporcionar uma experiência de exercício prazerosa e transformadora. Ao integrar tecnologia, saúde e personalização, tais aplicativos podem ser uma chave para a promoção de uma vida ativa e saudável na terceira idade (Bernson, 2024), devido a tal, o aplicativo de calistenia para a terceira idade pode ser complementado com orientações sobre hábitos saudáveis, como nutrição e descanso, que foram essenciais para a recuperação muscular e o bem-estar geral. A combinação de exercícios físicos e cuidados com o estilo de vida pode oferecer uma abordagem holística para a saúde do idoso, visando não apenas a aptidão física, mas também a melhoria da saúde mental e emocional (TMD, 2023; Shukla, 2023).

Nesse sentido, a questão central foi como um aplicativo de calistenia pode ser desenvolvido de maneira acessível e eficiente para este público, proporcionando uma introdução segura e gradual ao exercício físico?

Dessa forma, o objetivo geral desta proposta reside em: desenvolver um aplicativo de calistenia adaptado para pessoas da terceira idade iniciantes, com foco na promoção da saúde física e mental, segurança nos exercícios e incentivo à prática regular. Para se atingir o objetivo geral, elencam-se os seguintes objetivos específicos:

I. Analisar as necessidades e limitações físicas da terceira idade para desenvolver exercícios de calistenia adequados.

II. Criar uma interface amigável e acessível, levando em consideroução a facilidade de uso por pessoas idosas.
III. Desenvolver um sistema de acompanhamento de progresso para motivar e orientar os usuários durante sua jornada.
IV. Implementar funcionalidades de personalização de treinos, considerando a individualidade e os objetivos de cada usuário.

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentados os principais autores e temáticas do desenvolvimento do software, trazendo uma breve explicação.

2.1 Linguagem de programação *JavaScript*

A linguagem de programação *JavaScript* se consolidou como uma das mais importantes no desenvolvimento de aplicações web e móveis, principalmente devido à sua versatilidade e à constante evolução das ferramentas associadas a ela. Inicialmente criada para adicionar interatividade nas páginas web, *JavaScript* se expandiu para diversas áreas, como o desenvolvimento de aplicativos móveis e sistemas de servidor, além de integrar-se a soluções de dispositivos conectados à Internet das Coisas (IoT) (Lucas et al., 2025).

O crescimento de *frameworks* modernos como *React, Angular e Vue.js* facilitaram o desenvolvimento *front-end*, proporcionando arquiteturas baseadas em componentes e interfaces de usuário mais dinâmicas e eficientes. Esses *frameworks*, juntamente com a popularização do *Node.js*, que permitiu usar *JavaScript* no servidor, contribuíram para a crioução de aplicações *full-stack*. Essa abordagem simplificou a transição entre *front-end* e *back-end*, reduzindo a complexidade do desenvolvimento e permitindo uma maior uniformidade no uso da linguagem (Shukla, 2023).

Contudo, um dos desafios associados à adoção do *JavaScript* moderno foi a presença de padrões de código complexos, que podem tornar o código confuso e propenso a erros. Um estudo identificou padrões de código, chamados de "*confusing code patterns*", que aumentam a taxa de falhas no *software*, sendo muitas vezes corrigidos durante as atualizações. Esses padrões foram considerados problemáticos porque dificultam a leitura e a manutenção do código (Feijó et al., 2025).

Para o desenvolvimento de um aplicativo de calistenia para iniciantes da terceira idade, foi fundamental utilizar as versões mais recentes de *JavaScript*, com o suporte a *frameworks* modernos e boas práticas de codificação. A adoção dessas práticas contribui para

um desenvolvimento mais ágil e sustentável, além de melhorar a experiência do usuário, um fator importante quando se trata de aplicativos destinados ao público idoso (Lucas et al., 2025).

2.2 Framework *React Native* para desenvolvimento *Mobile*

O *React Native* foi um *framework* de código aberto desenvolvido pela Meta Platforms, Inc., que possibilita o desenvolvimento de aplicativos móveis nativos para várias plataformas, incluindo *Android, iOS, macOS, Windows e Web*. Lançado em 2015, o *React Native* permitiu que os desenvolvedores utilizem JavaScript em conjunto com a biblioteca *React*, criando interfaces de usuário nativas que proporcionam a experiência de um aplicativo nativo, mas com a eficiência do desenvolvimento multiplataforma (MP, 2025).

Uma das principais vantagens do *React Native* foi a reutilização de código, permitindo o desenvolvimento de aplicativos para diferentes plataformas com uma única base de código. Isso resulta em economia de tempo e recursos. O *framework* oferece uma rica biblioteca de componentes nativos, que facilita a criação de interfaces de usuário consistentes e de alto desempenho (MD, 2023).

Em 2023, o *React Native* passou por atualizações significativas com a introdução do novo sistema de renderização "*Fabric*" e a implementação dos "*Turbo Modules*". Essas melhorias aumentaram ainda mais o desempenho e a escalabilidade do *framework*, tornando-o ainda mais adequado para o desenvolvimento de aplicativos que exigem alto desempenho (TMD, 2023). Essas atualizações foram particularmente benéficas para o desenvolvimento de aplicativos focados na terceira idade, que frequentemente adotam funcionalidades como integração com dispositivos de monitoramento de saúde e suporte a exercícios físicos.

O *React Native* também conta com uma comunidade ativa de desenvolvedores e uma vasta gama de bibliotecas e *plugins*, o que facilita a integração de funcionalidades adicionais e a personalização dos aplicativos. A constante evolução do *framework*, impulsionada pela colaboração dessa comunidade, garante que ele se mantenha alinhado com as últimas inovações tecnológicas (USTD, 2023), com isso, o mesmo se torna uma ferramenta poderosa para o desenvolvimento de soluções móveis acessíveis e eficientes.

O *React Native* se apresenta como uma escolha robusta e eficiente para o desenvolvimento de aplicativos móveis, especialmente para iniciativas voltadas para o público da terceira idade. Sua capacidade de combinar desempenho nativo com o desenvolvimento multiplataforma, aliada a uma comunidade ativa, faz dele uma ferramenta valiosa para a crioução de soluções tecnológicas inclusivas e de fácil acesso (MD, 2023).

2.3 *Supabase*: *Rest API*, Autenticação e Banco de Dados como Serviço

O *Supabase* é uma plataforma *Backend-as-a-Service* (BaaS) que oferece um ecossistema integrado baseado no PostgreSQL, agregando camadas de autenticação, controle de acesso, APIs geradas automaticamente, eventos em tempo real e ferramentas de administração (SUPABASE, 2025). Diferentemente de soluções proprietárias, apoia-se em tecnologias consolidadas e padrões abertos, reduzindo acoplamento tecnológico e facilitando portabilidade.

No núcleo da plataforma está o *PostgreSQL*, SGBD relacional maduro que provê transações ACID, extensibilidade e mecanismos avançados de segurança como *Row Level Security* (RLS) (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2025; STONEBRAKER; ROWE, 1986). O *Supabase* expõe automaticamente uma API REST e *endpoints* GraphQL (quando habilitados) a partir do esquema do banco, eliminando a necessidade de *boilerplate* no *backend* e acelerando o ciclo de desenvolvimento (SUPABASE, 2025).

O módulo de autenticação abstrai fluxos de registro, *login* e gerenciamento de sessão, suportando email/senha, provedores OAuth e OTP (SUPABASE, 2025). *Tokens* de acesso seguem o padrão *JSON Web Token* (JWT), permitindo validação *stateless* no cliente e integração transparente com políticas de acesso (JONES; BRADLEY; SAKIMURA, 2015). A integração entre Auth e RLS é um diferencial: cada requisição autenticada carrega no JWT as *claims* (por exemplo, sub ou role) usadas em políticas declarativas no banco para restringir leitura e escrita linha a linha, alinhando segurança à camada de dados e minimizando lógica imperativa dispersa (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2025).

As políticas de RLS permitem modelar cenários como: usuário acessa apenas seus próprios registros; perfis administrativos obtêm visão agregada; filtros contextuais (ex.: data, estado ou nível de progressão). Essa abordagem reduz riscos de falhas de controle de acesso típicas de verificações manuais redundantes (OWASP FOUNDATION, 2024). A utilização de chaves distintas (*anon*/*public* e *service role*) reforça o princípio do menor privilégio: no aplicativo móvel emprega-se a chave pública (*anon*), limitada às políticas vigentes; operações privilegiadas (mutações administrativas, tarefas agendadas) deveriam residir em ambiente seguro utilizando a chave de serviço ou funções de *edge* (SUPABASE, 2025).

2.4 Geração de instalável com Expo e EAS-CLI

O uso do Expo como plataforma de desenvolvimento React Native permite um fluxo de trabalho gerenciado que abstrai configurações nativas e acelera a prototipagem e manutenção do aplicativo, concentrando o desenvolvimento em JavaScript/TypeScript e nas APIs fornecidas pelo SDK (Expo, 2025).

Complementado pelo EAS (Expo Application Services), o processo de build passa a ser executado na nuvem com profiles configuráveis, gerenciamento de credenciais e suporte a segredos, viabilizando a geração reprodutível de artefatos Android (APK ou AAB) sem a necessidade de infraestrutura local de compilação (EAS, 2025).

Para distribuição direta e testes em dispositivos, o APK é o formato mais imediato, enquanto o AAB é recomendado para publicação na Play Store conforme as diretrizes do Android (Android Developers, 2024).

As principais vantagens observadas são: reprodutibilidade de builds entre desenvolvedores e CI, redução do custo e complexidade de setup local, e gerenciamento centralizado de credenciais; limitações incluem dependência de serviço remoto e possíveis ajustes nativos para módulos que exigem configuração manual (EAS, 2025).

3 METODOLOGIA

A metodologia deste estudo foi estruturada para proporcionar um entendimento claro e preciso do desenvolvimento de um aplicativo de calistenia voltado para iniciantes da terceira idade. A seguir, foram descritos os materiais, métodos e as abordagens que foram utilizadas para alcançar os objetivos propostos, considerando as necessidades específicas do público-alvo e o ambiente tecnológico envolvido (Lakatos; Gil, 2022).

3.1 Material e métodos

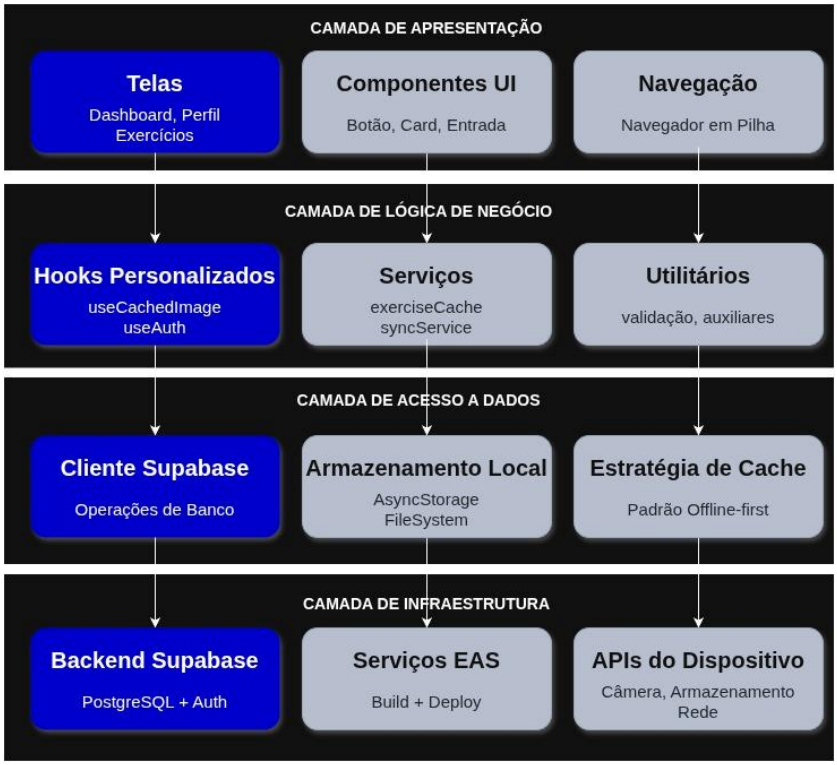
Para o desenvolvimento do sistema, foram adotoudas metodologias ágeis, como Scrum, e utilizoudas ferramentas como *React Native* para a interface móvel e *Supabase* para o *backend*. Além disso, foi empregado o banco de dados *PostgreSQL* dentro do *Supabase* para a gestão das informações dos usuários, incluindo dados de progresso físico e histórico de atividades. No que se refere aos métodos, a pesquisa seguiu um modelo de desenvolvimento comparativo, no qual as funcionalidades foram desenvolvidas verificando os padrões de mercado e funcionalidades populares.

3.2 Arquitetura do *software*

A pesquisa adotou uma arquitetura multicamadas baseada em *React native*, priorizando simplicidade, acessibilidade e experiencia offline para usuários idosos.

Na Figura 1 é possível visualizar a arquitetura completa do *software*, onde cada serviço está presente e o fluxo seguido.

Figura 1 – Arquitetura do App



Fonte: Dados da pesquisa (2025)

A *Prsentation Layer* (Camada de Apresentação) é a camada mais externa, responsável pela interface do usuário e interação direta com o usuário final. Contém toda a lógica visual e de navegação do aplicativo.

A *Business Logic Layer* (Camada de Lógica de Negócio) é a intermediária que contém toda a lógica de negócio, regras da aplicação e orquestração de dados. Atua como ponte entre a apresentação e o acesso a dados.

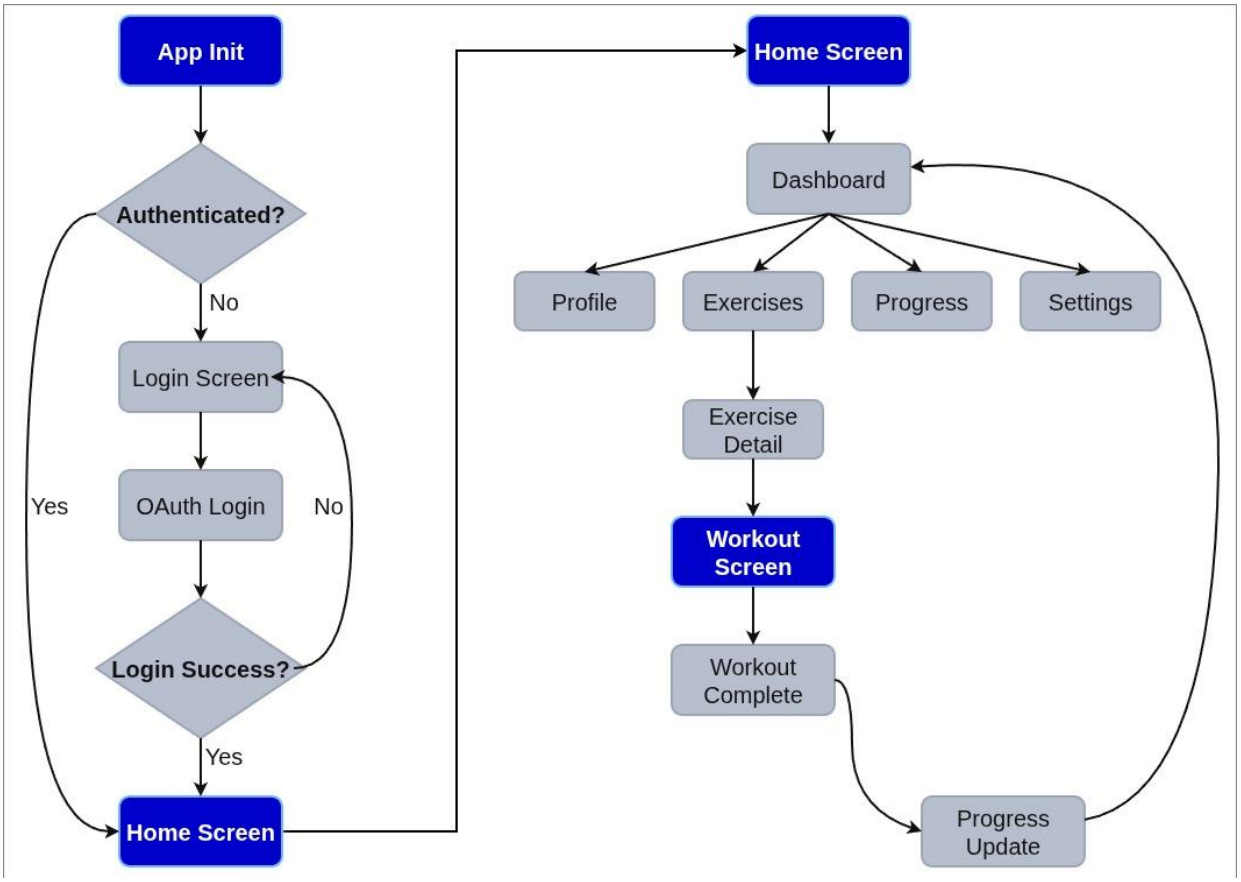
A *Data Access Layer* (Camada de Acesso a Dados) é a camada responsável por toda comunicação com fontes de dados externas e internas. Abstrai os detalhes de persistência e recuperação de dados.

A *Infrastructure Layer* (Camada de Infraestrutura) é a camada base que fornece serviços de infraestrutura, APIs externas e configurações do ambiente.

3.2.1 Arquitetura de Navegação

A Figura 2 mostra que a navegação foi projetada seguindo os princípios de usabilidade para idosos, priorizando fluxos intuitivos e redução da carga cognitiva. O sistema utiliza o *React Navigation* v6 com *Native Stack Navigator*, proporcionando transições nativas e performance otimizada para dispositivos com menor capacidade de processamento.

Figura 2 – Arquitetura da Navegação no momento de Login



Fonte: Dados da pesquisa, 2025.

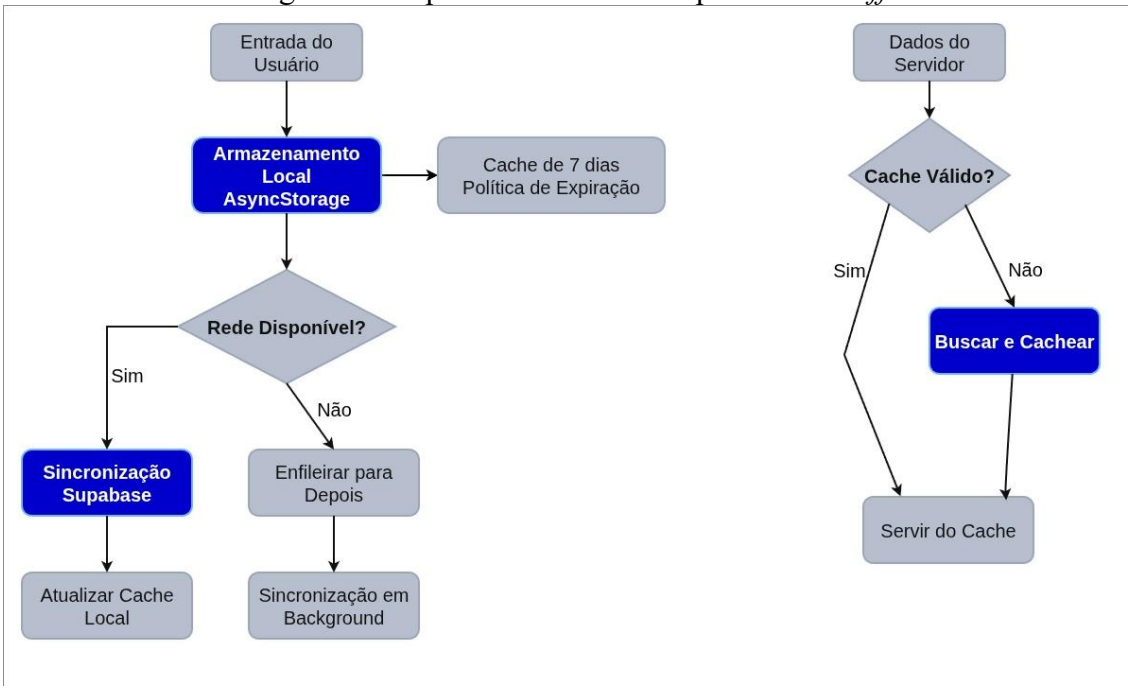
O fluxo inicia-se abrindo o app, que verifica automaticamente a existência de token JWT válido no *AsyncStorage*, que realiza três verificações: existência, validade temporal e integridade do token. Se positivo, redireciona para a *Home* caso contrário, para tela de *Login*. A tela de *Login* oferece dois métodos de autenticação: visitante e *OAuth* 2.0 com Google. A autenticação por visitante valida credenciais no *backend* Supabase e armazena o token recebido. O OAuth Login usa método hash para segurança em aplicativos móveis, gerando códigos, redirecionando para autorização do Google e trocando código por tokens.

Após autenticação, o usuário acessa a tela *Home*, o *dashboard* central que apresenta saudação personalizada, dica do dia e acesso rápido às telas disponíveis.

3.2.2 Arquitetura de Dados e Persistência

A Figura 3 representa como os dados são armazenados, possibilitando que o aplicativo funcione *offline*, sincronizando quando estiver *online*.

Figura 3 – arquitetura dos dados e persistência *offline*



Fonte: Dados da pesquisa, 2025.

O fluxo inicia-se com a entrada do usuário, que aciona o armazenamento local *AsyncStorage* para persistência imediata dos dados, que mantém em cache de 7 dias. Simultaneamente, o sistema verifica disponibilidade de rede.

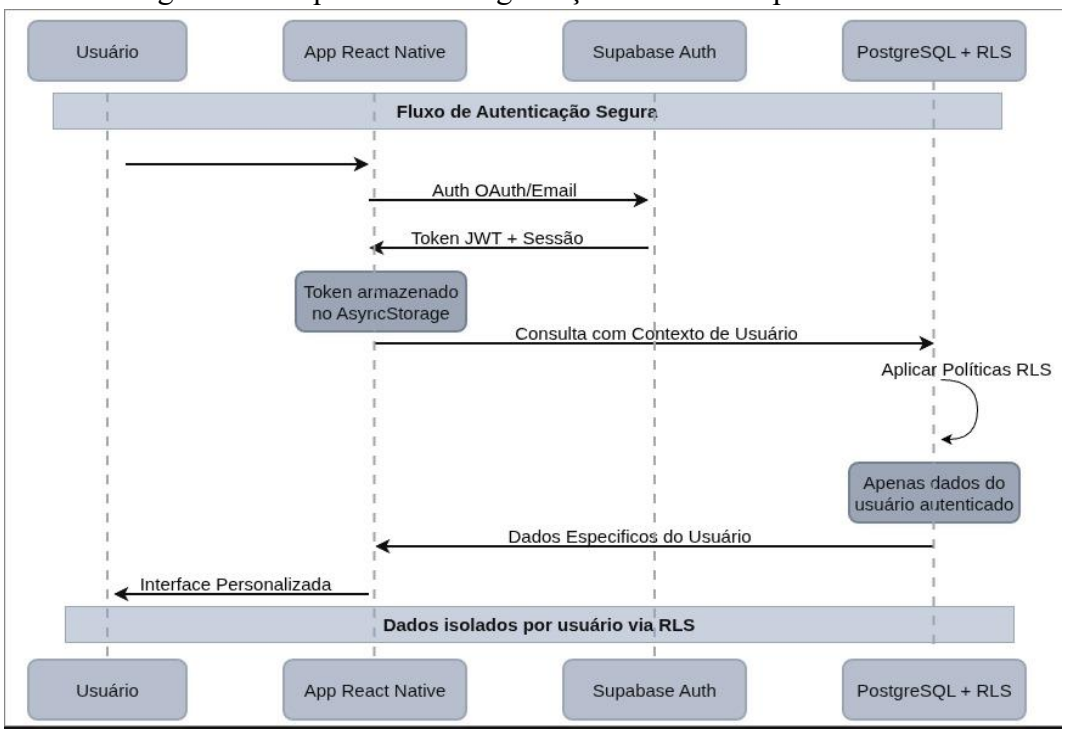
Quando há conexão, o sistema executa sincronização no Supabase em tempo real, atualizando o *backend* e acionando processos paralelos de atualizar o cache local e sincronização em b*ackground*. No cenário sem conectividade, dados são enfileirados para armazenar depois, armazenando operações pendentes localmente para processamento posterior. O lado direito do diagrama ilustra o fluxo de requisição de dados ao servidor, onde o sistema primeiro verifica se o cache é válido. Se positivo, serve dados do cache; caso contrário, executa uma busca e cacheamento, atualizando repositório local antes de disponibilizar informações ao usuário.

Esta arquitetura dual (escrita local + sincronização assíncrona) assegura que usuários idosos, frequentemente com conectividade instável, possam registrar treinos e consultar exercícios sem interrupções, com sincronização transparente ao restabelecer conexão

3.2.3 Arquitetura de Autenticação e Segurança

A Figura 4 demonstra como está implementada a arquitetura de segurança, evidenciando o uso do *supabase* para tratar de segurança a nível de linha no próprio banco através das RLS (*Row Level Security*).

Figura 4 – Arquitetura de Segurança através do supabase



Fonte: Dados da pesquisa, 2025.

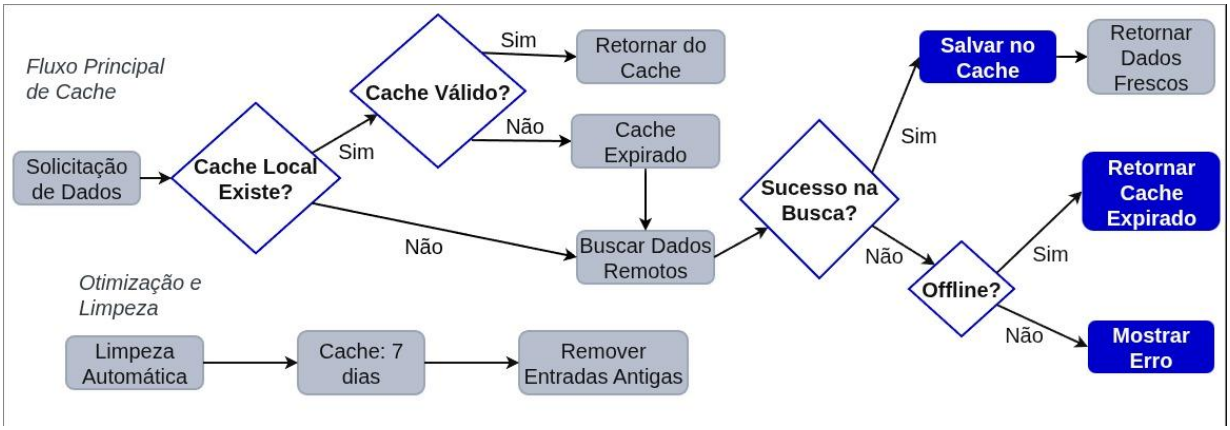
O fluxo de autenticação seguro estabelece comunicação entre quatro camadas principais: Usuário, aplicativo React Native, Supabase Auth e PostgreSQL com RLS. O processo inicia-se quando o usuário requisita autenticação via OAuth ou visitante. O Supabase Auth valida as credenciais e retorna token JWT (_JSON Web Token_) armazenado no AsyncStorage para persistência local. A cada operação subsequente, o aplicativo consulta recursos incluindo contexto do usuário através do token JWT. O Supabase Auth intercepta estas requisições e aplica automaticamente políticas RLS no PostgreSQL, garantindo que cada consulta SQL seja filtrada conforme identidade do usuário autenticado.

As políticas RLS são definidas através de regras declarativas para operações SELECT, INSERT, UPDATE e DELETE. Esta arquitetura implementa segurança em profundidade linha a linha, permitindo isolamento lógico de dados por usuário por exemplo.

3.2.4 Arquitetura de Cache e Performance

A Figura 5 demonstra como foi arquitetado o sistema de cache, que expira após 7 dias e verifica se o cache local existe e é válido, para retorná-lo em caso de uma solicitação de informações.

Figura 5 – Arquitetura do Cache



Fonte: Dados da pesquisa, 2025.

O sistema implementa estratégia de cache em múltiplas camadas para otimização de performance. O fluxo inicia-se com solicitação de dados, verificando existência de cache local. Em caso positivo, valida-se temporalmente através de comparação com timestamp de criação (política de 7 dias). Cache válido é retornado imediatamente, eliminando latência de rede. Caso inexista ou esteja expirado, executa-se busca remota no Supabase; dados obtidos são armazenados localmente antes de retorno ao usuário. Paralelamente, ocorre a limpeza automática que remove arquivos expirados.

3.5 Tecnologias e Ferramentas

A seleção do *stack* tecnológico para o desenvolvimento priorizou simplicidade, performance em dispositivos antigos e capacidade offline robusta, características essenciais para aplicações voltadas ao público idoso.

O Quadro 1 demonstra as tecnologias principais utilizadas para desenvolver o aplicativo.

Quadro 1 –*Stack* Tecnológico

Componente	**Tecnologia**	**Versão**	**Justificativa**
Framework Mobile	React Native + Expo	081.2 + SDK 54	Desenvolvimento multiplataforma com builds gerenciados
Linguagem	TypeScript	5.x	Type safety e prevenção de erros em runtime
Backend	Spabase BaaS	Latest	PostgreSQL gerenciado + Auth + Real-time
Navegação	React Navigation v6	Native Stack	Performance nativa em dispositivos antigos
Estado	React Hooks	Nativo	Simplicidade sem dependências externas
Cache Local	AsyncStorage + FileSystem	Nativo	Estratégia offline-first completa
Build/Deploy	EAS Build	Nativo	CI/CD gerenciado pela Expo

Fonte: Dados da pesquisa, 2025.

Quanto as ferramentas, foram utilizadas as seguintes para o desenvolvimento desta pesquisa:

Visual Studio Code: Editor de código principal utilizado no desenvolvimento, escolhido por sua integração nativa com TypeScript, sistema de extensões robusto e suporte específico para React Native. O VS Code oferece IntelliSense avançado que auxilia na detecção de erros em tempo de desenvolvimento e autocompletar baseado nos tipos TypeScript definidos no projeto.

Expo CLI: Ferramenta de linha de comando para gerenciamento do projeto Expo, permitindo inicialização do servidor de desenvolvimento, execução em emuladores/dispositivos físicos e gerenciamento de builds. Utilizada através dos comandos npm start para desenvolvimento local e npm run android para testes em emulador Android.

EAS CLI: Ferramenta de linha de comando do Expo Application Services para gerenciamento de builds e deploys. Automatiza o processo de compilação nativa através de comandos como eas build --platform android, eliminando necessidade de configurar Android Studio ou Xcode localmente.

Git e GitHub: Sistema de controle de versão utilizado para gerenciamento do código-fonte, permitindo rastreamento de mudanças, colaboração e backup do projeto. O repositório está hospedado no GitHub em github.com/AntonyHaro/mobile-vivafit-seniors.

Android Studio (Emulador): Utilizado exclusivamente para execução do emulador Android durante desenvolvimento e testes. Embora a configuração nativa não seja necessária devido ao Expo, o emulador Android do Android Studio é utilizado para validação da aplicação.

Supabase Dashboard: Interface web administrativa do Supabase para gerenciamento do banco de dados PostgreSQL, configuração de Row Level Security (RLS),

monitoramento de autenticação e execução de queries SQL. Acessível através do console web do Supabase.

3.6 Análise de dados

A análise foi feita de forma qualitativa (através de comparação com outros aplicativos do meio *fitness*). A comparação entre os aplicativos ajudou a medir a adequação da pesquisa com o público-alvo, bem como as diferenças entre os aplicativos que já existem no mercado e o vivafit.

No Quadro 2 é apresentado o comparativo de aplicativos que já existem no mercado.

Quadro 2 – Comparativo de aplicativos do mercado

Característica	**VivaFit Seniors**	**SilverSneakers GO**	**Fitness Coach for Seniors**	**Nike Training Club**	**Adidas Training**	**FitOn**
Público-Alvo	Exclusivo 60+	Idosos (EUA)	Idosos	Geral (jovens)	Geral (jovens)	Geral
Disponibilidade Brasil	Disponível	Não disponível	Inglês apenas	Disponível	Disponível	Disponível
Modo Offline Completo	100% funcional	Parcial	Requer conexão	Parcial	Parcial	Requer conexão
Acessibilidade Senior	Touch 60px+, alto contraste	Boa	Básica	Padrão pequeno	Padrão pequeno	Média
Exercícios Adaptados	Cadeira, baixo impacto	Sim	Sim	Alta intensidade	Alta intensidade	Mix geral
Duração dos Treinos	5-15 min	10-30 min	10-20 min	15-45 min	15-60 min	5-60 min
Interface Senior-Friendly	Boa	Boa	Média	Complexa	Complexa	Média
Prevenção de Quedas	Foco em equilíbrio	Sim	Sim	Não	Não	Não
Sistema de Progressão	Gradual senior-specific	Adaptado	Básico	Competitivo	Competitivo	Genérico
Conquistas Motivacionais	Alcançáveis	Sim	Limitado	Sim (competitivo)	Sim (competitivo)	Sim
Dicas de Saúde	Nutrição + bem-estar sênior	Básicas	Básicas	Performance	Performance	Gerais
Modelo de Negócio	Gratuito 100%	Freemium (US$)	Pago (US$ 4.99)	Freemium	Freemium	Freemium
Privacidade de Dados	RLS + dados locais	Média	Média	Coleta extensiva	Coleta extensiva	Média
Integração Wearables	Não (v1.0)	Sim	Não	Sim	Sim	Sim

Vídeos Demonstrativos	Instruções detalhadas	Sim	Sim	Profissionais	Profissionais	Sim
Suporte a Limitações	Perfil personalizado	Básico	Sim	Não	Não	Não
Comunidade/Social	Não (v1.0)	Sim	Não	Sim	Sim	Sim
Lembretes Personalizados	Treinos + hidratação	Sim	Básico	Sim	Sim	Sim

Fonte: Dados da pesquisa, 2025.

4 DESENVOLVIMENTO

Durante o desenvolvimento do *software*, foi utilizado um *desktop* equipado com um processador AMD Ryzen 5600G e 16 GB de memória RAM, utilizando sistema operacional Ubuntu 24.04 LTS.

Seguindo as melhores práticas visando acessibilidade para pessoas da terceira idade, cada tela foi desenvolvida de modo que os componentes tenham tamanho maior, menos textos e menos botões para facilitar a utilização, tornando o aplicativo mais intuitivo e prático. A Figura 6 mostra a tela de *login.*

Figura 6 – Tela de *login* com *Google OAuth*

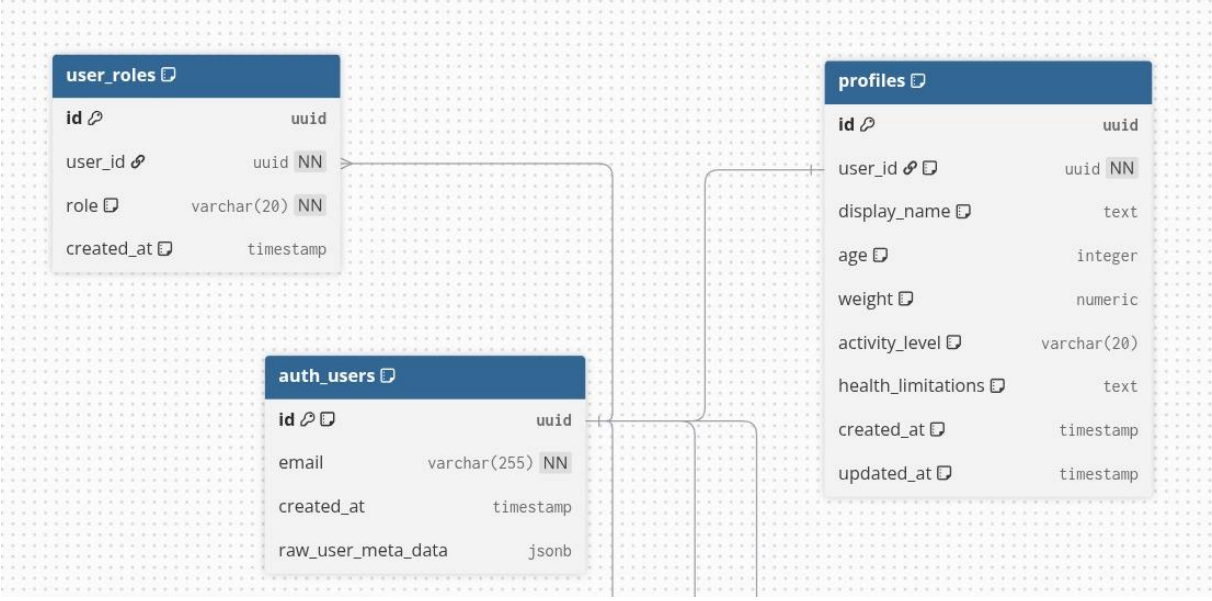
Fonte: Dados da Pesquisa, 2025.

Utilizando um padrão de maior acessibilidade para pessoas da terceira idade o modelo do *login* tem apenas 2 botões, um onde ele pode se cadastrar usando sua conta *google* e outro entrando sem cadastro, caso ele não tenha uma conta google. Ao realizar o login via google ele é enviado para o servidor *backend* através de uma requisição.

Ao receber a requisição, o servidor verifica no banco de dados hospedado no Supabase, que por sua vez está integrado diretamente a API Oauth da *google*, que faz a verificação do cadastro da conta *google*.

A Figura 7 mostra a tabela de perfil do usuário num modelo e entidade relacionamento que é salva no serviço *baas(Backend as-a-service) Supabase*.

Figura 7 – Modelo Entidade relacionamento do Perfil



Fonte: Dados da Pesquisa, 2025.

Este é o formato em que as tabelas estão salvar no supabase, que por padrão gera um schemas apenas para autenticação, possuindo a tabela auth_users(ou outro nome) e então ligamos ela a tabela profiles para conectar diretamente ao perfil de usuário do app.

A tabela *profiles* no Supabase armazena informações personalizadas dos usuários, essenciais para customização da experiência e recomendações de exercícios adequadas ao perfil individual. A estrutura de dados implementa modelo relacional com campos específicos para o contexto de atividade física. Alguns dos campos incluem o nome do usuário, que será usado para exibir nas telas, a idade e o peso. Alguns campos foram pensados para o público idoso, visando adaptação do app, eles são o nível de atividade física, que funciona em três escalas, entre baixo, médio e atleta

Políticas RLS garantem que usuários acessem exclusivamente seus próprios perfis, seguindo em conformidade com políticas de privacidade e a LGPD.

Já na Figura 8, é apresentado a tela principal apresenta cards grandes e espaçados para identificar cada tela disponível. A tela conta com uma saudação personalizada e uma dica de saúde contextualizada.

Figura 8 – Tela Principal do App

Fonte: Dados da Pesquisa, 2025.

Na Figura 9, é possível visualizar a tela de treino, que implementa a exucução guiada do exercício, com timer visual, instruções passo a passo e opção de iniciar o exercício, pausar e até pular quando desejado. A tela conta com bloqueio de gestos acidentais durante o treino.

Conta também com uma barra de progresso visual para acompanhar o avanço do treino e salvamento automático em caso de interrupção.

Na Figura 10, é apresentado o alerta de treino concluído com sucesso, possibilitando duas saídas, uma para a tela de progresso para verificar as atividades feitas ou ir direto para a tela principal selecionando ok.

Figura 9 – Tela de treino

Figura 10 – Finalização de Treino

Fonte: Dados da pesquisa , 2025.

Nas Figura 11 e 12, é apresentado a tela de Perfil, que permite personalizar as informações do usuário, incluindo nome de exibição, idade, peso, nível de atividade física e limitações de saúde.

Os dados são salvos localmente primeiro e sincronizados com Supabase quando a conectividade está disponível.

Figura 11 – Tela de Perfil

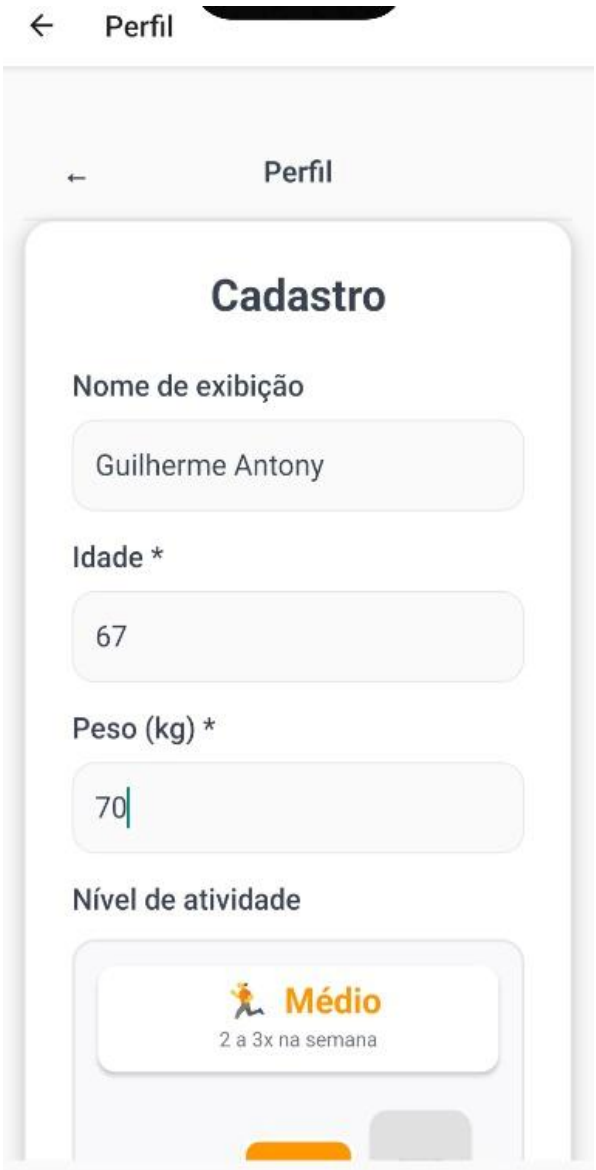
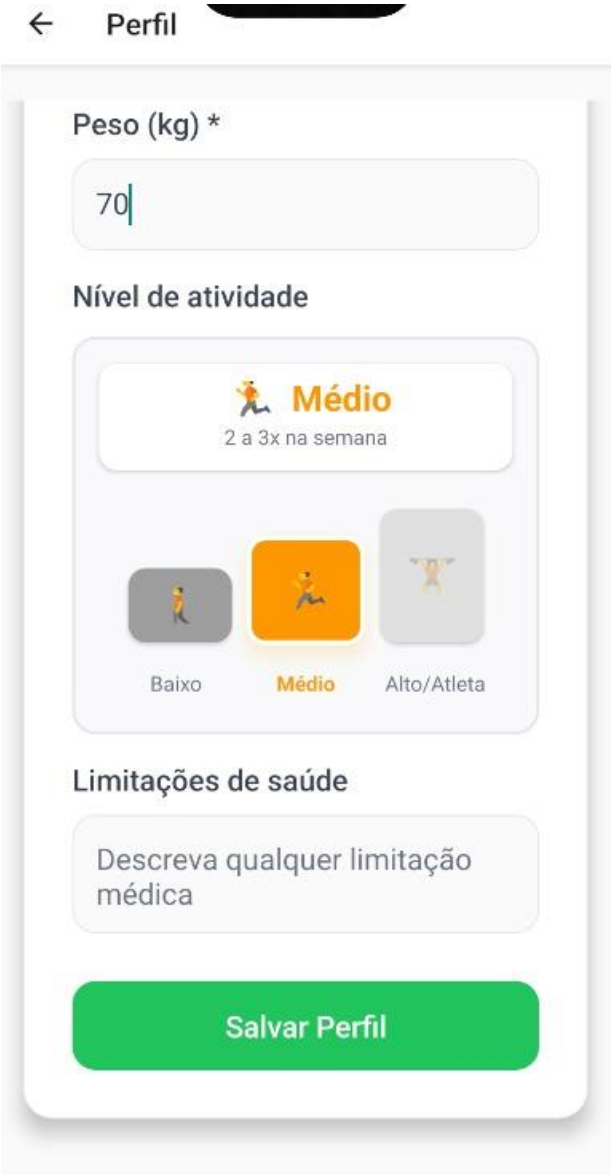


Figura 12 – Tela de Perfil



Fonte: Dados da pesquisa, 2025.

Na Figura 13, é apresentado a tela de exercícios, catálogo principal do aplicativo, apresentando todas as atividades físicas disponíveis organizadas de forma intuitiva para o público idoso. Nesta tela foi implementada a estratégia *offline-first*, garantindo acesso completo aos exercícios mesmo sem conexão a internet.

A tela implementa um sistema de filtros por categoria e dificuldade, facilitando localizar exercícios adequados às limitações do usuário.

Na Figura 14, é possível ver as informações contidas no card de cada exercício, com o tempo de execução, benefícios, classificação e botões de iniciar o exercício e de detalhas para demonstrar como executá-lo.

Figura 13 – Tela de Exercícios

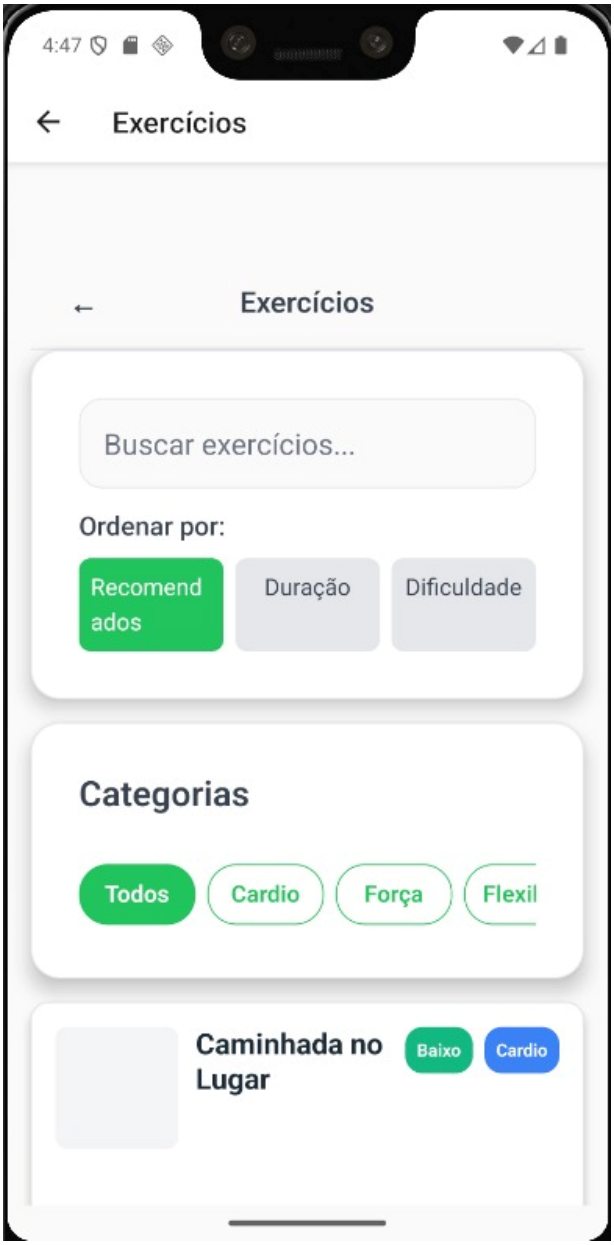
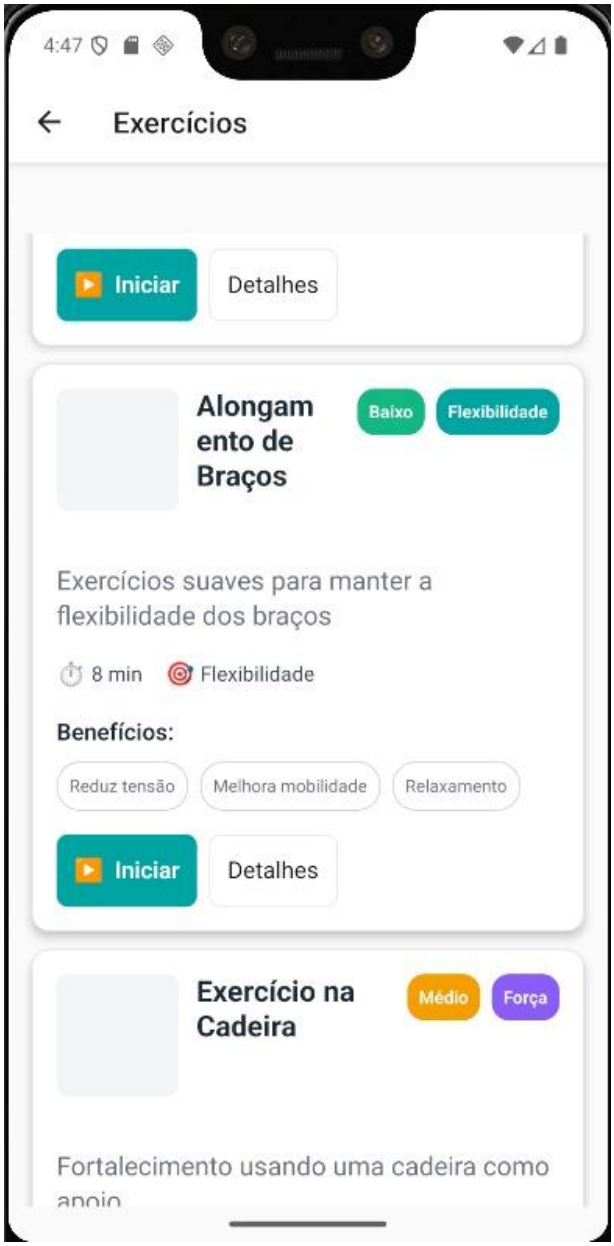


Figura 14 – Tela de Exercícios



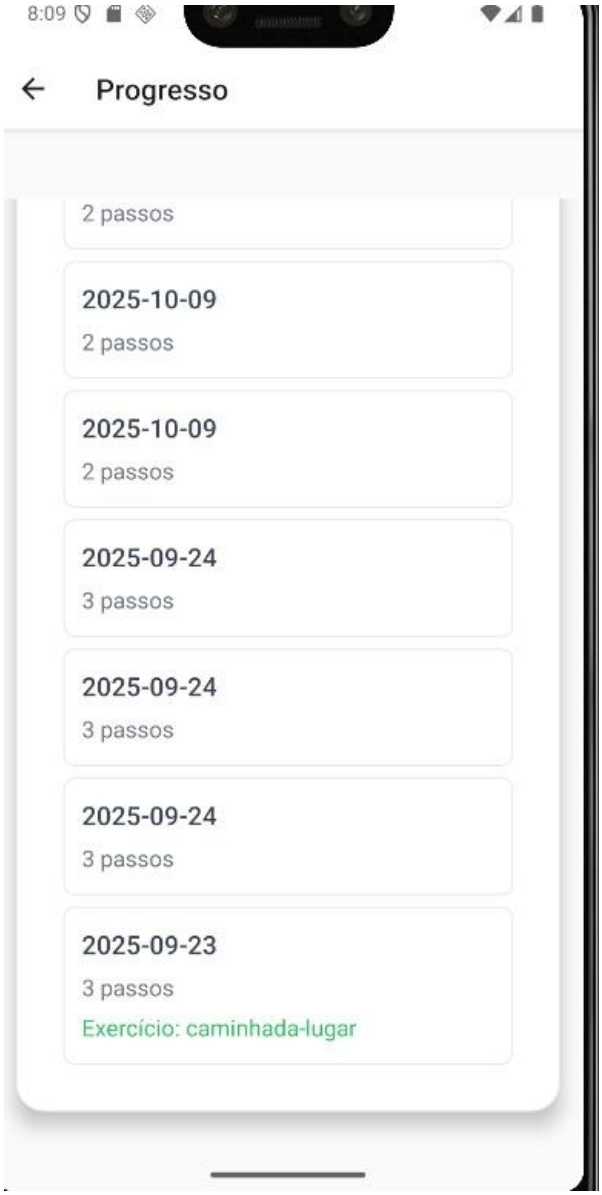
Fonte: Dados da pesquisa, 2025.

Nas Figuras 15 e 16, podemos ver a parte de progresso que apresenta estatísticas sobre os treinos feitos.

Os treinos são apresentados no formato de lista cronológica reversa dos treinos recentes, do mais recente ao mais antigo. Cada um exibe a data que foi feito o treno, numero de passos (ou repetições), e nome do exercício executado.

Figura 15 – Tela de Progresso

Figura 16 – Tela de Progresso

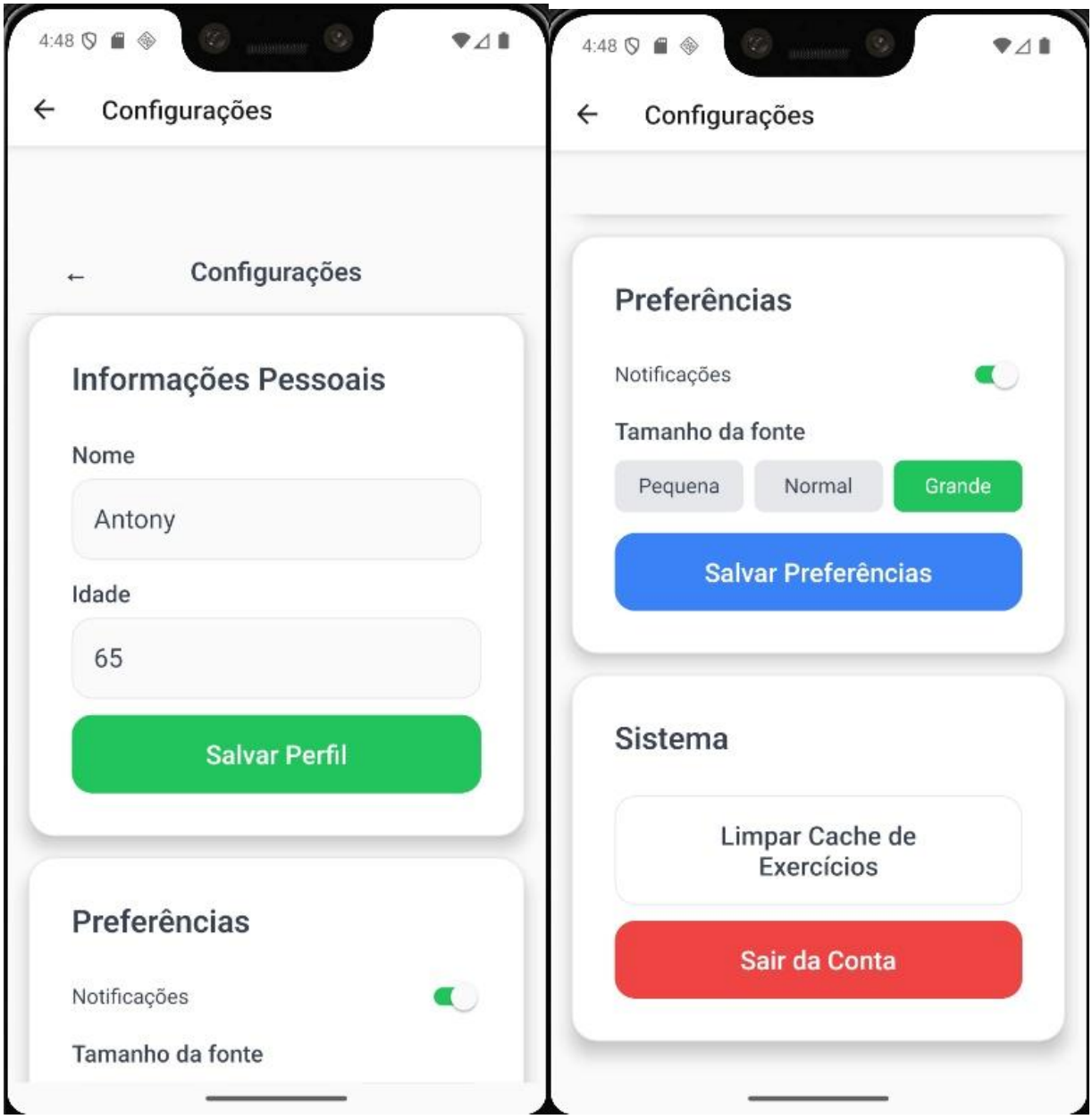


Fonte: Dados da Pesquisa, 2025.

Nas Figuras 17 e 18, é apresentado o painel de Opcões que centraliza as preferencias do aplicativo e ações do sistema. Oferecendo controle da notificações e seleção do tamanho da fonte, funcionalidade essencial para acessibilidade sênior.

Figura 17 – Tela de Opções

Figura 18 – Tela de Opções



Fonte: Dados da Pesquisa, 2025.

Há também a opção de limpeza de cache, que libera espaço em armazenamento utilizado pelos exercícios salvos e faz logout removendo dados da sessão local. Por fim a opção de sair da conta, caso queira sair para trocar a conta google utilizada no aplicativo.

5 CONCLUSÃO

Os resultados obtidos durante o desenvolvimento do VivaFit Seniors demonstram que a

aplicação apresenta uma arquitetura técnica sólida e uma interface de fácil utilização, especialmente projetada para pessoas idosas com conhecimento tecnológico limitado, permitindo que realizem exercícios físicos de maneira simples, segura e eficaz.

Este *software* proporciona funcionalidades capazes de gerenciar e monitorar a saúde física de usuários com 60 anos ou mais, oferecendo exercícios adaptados às limitações individuais, sistema de progressão gradual e acompanhamento de histórico de treinos. A implementação da estratégia *offline-first* garante que o aplicativo funcione completamente sem conexão com a internet, eliminando barreiras tecnológicas comuns enfrentadas pelo público idoso. A aplicação permite personalização do perfil de atividade física, visualização de exercícios categorizados por dificuldade e duração, além de oferecer flexibilidade para adaptar treinos conforme as condições de saúde de cada usuário.

O aplicativo *mobile* desenvolvido oferece uma solução inovadora para promoção de atividade física entre idosos brasileiros, utilizando tecnologias modernas como *React Native*, Expo SDK 54, *TypeScript* e *Supabase*. O maior desafio técnico foi implementar o modelo de dados relacional com *Row Level Security* (RLS) garantindo isolamento completo de informações entre usuários, além de desenvolver uma interface que atendesse aos critérios de acessibilidade WCAG 2.1 nível AA, com *touch targets* de 60 pixels, alto contraste e fontes ampliadas. Esse processo aprimorou as funcionalidades de cache local, sincronização bidirecional de dados e performance em dispositivos antigos, resultando em um *software* que tem potencial para atender significativamente a população idosa brasileira, estimada em mais de 30 milhões de pessoas segundo o IBGE.

Em estudos futuros, este *software* pode ser aprimorado para incluir funcionalidades como integração com dispositivos *wearables* (*smartwatches* e sensores de saúde), sistema de comunidade social para interação entre usuários, videoconferências para treinos em grupo, e parcerias com profissionais de educação física e fisioterapeutas para acompanhamento remoto. A flexibilidade da arquitetura baseada em *Supabase* permite escaloná-la para um público mais amplo através de integrações com secretarias municipais de saúde, centros de convivência para idosos e planos de saúde, tornando-a uma solução versátil que contribui para o envelhecimento ativo e saudável da população brasileira, alinhando-se às diretrizes da Política Nacional de Saúde da Pessoa Idosa.

A contribuição científica deste trabalho reside na demonstração de que tecnologias modernas, quando adequadamente adaptadas com foco em acessibilidade e usabilidade *senior-friendly*, podem efetivamente promover inclusão digital e melhorar a qualidade de vida de idosos através da atividade física regular, estabelecendo um *framework* técnico replicável para

futuras aplicações de *e-health* voltadas a este público.

REFERÊNCIAS

ANDROID DEVELOPERS. Android App Bundles & APKs. 2024. Disponível em: https://developer.android.com/guide/app-bundle. Acesso em: 2 novembro. 2025.

Bachina, BHARGAV. Efficient RESTful API Development with NodeJS, Express, and PostgreSQL. European Journal of Advances in Engineering and Technology, v. 10, n. 6, p. 26-37, 2023. Disponível em: https://ejaet.com/PDF/10-6/EJAET-10-6-26-37.pdf. Acesso em: 14 maio 2025.

EAS. EAS Build Documentation. 2025. Disponível em: https://docs.expo.dev/eas-build/overview/. Acesso em: 6 nov. 2025.

EXPO. Expo Documentation — Overview. 2025. Disponível em: https://docs.expo.dev. Acesso em: 2 novembro 2025.

Feijó, Diego N.; ALMEIDA, Carlos D. A.; ROCHA, Lincoln S. An Investigation of confusing code patterns in JavaScript. Empirical Software Engineering, v. 30, p. 107, 2025. Disponível em: https://www.researchgate.net/profile/Ed-Canedo/publication/370651048_An_Investigation_of_confusing_code_patterns_in_JavaScript/links/6462b385f43b8a29ba52764c/An-Investigation-of-confusing-code-patterns-in-JavaScript.pdf. Acesso em: 14 maio 2025.

Geeksforgeeks. How to Design a Database for Health and Fitness Tracking Applications. 2023. Disponível em: https://www.geeksforgeeks.org/how-to-design-a-database-for-health-and-fitness-tracking-applications/. Acesso em: 14 maio 2025.

Github. Fitness Tracker Database. 2025. Disponível em: https://github.com/makoohara/health_and_fitness_db. Acesso em: 14 maio 2025.

IJRPR. Express.js: A scalable framework for backend development. Disponível em: https://ijrpr.com/uploads/V5ISSUE4/IJRPR25041.pdf. Acesso em: 14 maio 2025.

LAKATOS, E.M; Gil, A. M. METODOLOGIA CIENTIFICA - 8ªED. São Paulo: Atlas, 1998. 2022. Disponível em:

https://docentes.ifrn.edu.br/olivianeta/disciplinas/copy_of_historia-i/historia-ii/china-e-india/at_download/file. Acesso em 22 jun. 2025

Lima, G. *Designing Scalable Database Systems with Sequelize*. International Journal of Database Systems, v. 8, n. 2, p. 99-107, 2021. Disponível em: https://moldstud.com/articles/p-designing-scalable-database-systems-in-technical-architecture. Acesso em: 14 maio 2025.

Lucas, Walter; NUNES, Rafael; BONIFÁCIO, Rodrigo; *et al*. Understanding the adoption of modern Javascript features: An empirical study on open-source systems. Empirical Software Engineering, v. 30, p. 107, 2025. Disponível em: https://link.springer.com/article/10.1007/s10664-025-10663-9. Acesso em: 14 maio 2025.

MD. MYTH DIGITAL. The Best Mobile App Development Frameworks in 2023. Disponível em: https://myth.digital/mobile-app-development-blog/the-best-mobile-app-development-frameworks-in-2023/. Acesso em: 14 maio 2025.

MDN. Introdução Express/Node Express.js Documentation. WEB DOCS. 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction. Acesso em: 14 maio 2025.

MP. META PLATFORMS, INC. React Native. 2025. Disponível em: https://reactnative.dev/. Acesso em: 14 maio 2025.

NATARAJAN, THAMIZHINIYAN; PICHAI, SHANMUGAVADIVU. Behaviour-driven development and metrics framework for enhanced agile practices in scrum teams. Information and Software Technology, v. 170, p. 107435, 2024. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0950584924000405. Acesso em: 14 mai. 2025.

NodeJS. How to read environment variables from Node.js. 2025. Disponível em: https://nodejs.org/en/learn/command-line/how-to-read-environment-variables-from-nodejs. Acesso em: 14 mai. 2025.

NodeJS. Introduction to Node.js. 2025. Disponível em: https://nodejs.org/pt/learn/getting-started/introduction-to-nodejs. Acesso em: 14 mai. 2025.

PALOPAK, YULIANUS; HUANG, SUN-JEN. Perceived impact of agile principles: Insights from a survey-based study on agile software development project success. *Information and Software Technology*, v. 176, p. 107552, 2024. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0950584924001575. Acesso em: 14 mai. 2025.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL Documentation. 2025. Disponível em: https://www.postgresql.org/docs/. Acesso em: 28 set. 2025.

SUPABASE. Supabase Documentation. 2025. Disponível em: https://supabase.com/docs. Acesso em: 28 set. 2025.

Shukla, Abhishek. Modern JavaScript Frameworks and JavaScript's Future as a *Full-stack* Programming Language. Journal of Artificial Intelligence & Cloud Computing, v. 2, n. 4, p. 2-5, 2023. Disponível em: https://www.researchgate.net/profile/Abhishek-Shukla-41/publication/377629693_Modern_JavaScript_Frameworks_and_JavaScript%27s_Future_as_a_*Full-stack*_Programming_Language/links/65bbf3fd1bed776ae31ccd6c/Modern-JavaScript-Frameworks-and-JavaScripts-Future-as-a-*Full-stack*-Programming-Language.pdf. Acesso em: 14 maio 2025.

TMD. THE MORROW DIGITAL. React Native Year in Review 2023 - Key Updates & Highlights. 2023. Disponível em: https://www.themorrow.digital/blog/react-native-year-in-review. Acesso em: 14 maio 2025.

USTD. UPSTACK STUDIO. The Ultimate Guide To Choosing React Native For Your Mobile App in 2023. Disponível em: https://upstackstudio.com/blog/react-native/. Acesso em: 14 maio 2025.

ANEXO A – TÍTULO DO ANEXO