INTEGRAÇÃO DE APLICATIVO MOBILE COM API REST PARA APOIO A AGENTES DE SAÚDE NA COLETA DE DADOS DE IDOSOS EM ÁREAS REMOTAS

Vinicius Sumara Fortuna[1], André Faria Ruaro[2]

Resumo: O presente trabalho aborda o problema da coleta de dados de idosos realizada por agentes de saúde e agentes de pesquisa em regiões remotas, onde o acesso à internet é limitado ou inexistente. Essa limitação compromete o registro e o envio das informações de forma ágil e confiável, ocasionando atrasos, retrabalho e falhas na comunicação entre as equipes de saúde. Como solução, desenvolveu-se um protótipo composto por um aplicativo móvel, uma aplicação web e uma *Application Programming Interface* (API) baseada em *Representational State Transfer* (REST) que realiza a integração entre ambos. O aplicativo permite o funcionamento em modo offline, armazenando localmente os dados coletados e possibilitando a sincronização com o servidor assim que a conexão é restabelecida. Já a aplicação web oferece recursos para o cadastro e gerenciamento de informações gerais, como usuários, agentes e clientes. A pesquisa buscou comprovar que o uso dessa integração tecnológica aumenta a eficiência do processo de coleta e melhora a gestão das informações. Os resultados demonstraram que o protótipo é funcional, leve e atende adequadamente às necessidades dos agentes de saúde e agentes de pesquisa. Conclui-se que a solução proposta contribui significativamente para a modernização do processo de coleta de dados, fortalecendo a atuação dos profissionais e a qualidade dos serviços prestados à população idosa.

Palavras-chave: aplicativo mobile; aplicação web; API REST; agentes de saúde; coleta de dados; sincronização offline.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), vini.s.fortuna@unesc.net

[2]Orientador. Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), andre.ruaro@unesc.net

ABSTRACT: This study addresses the problem of data collection on elderly individuals carried out by community health agents in remote regions, where internet access is limited or nonexistent. This limitation compromises the agile and reliable recording and transmission of information, causing delays, rework, and communication failures among healthcare teams. As a solution, a prototype was developed consisting of a mobile application, a web application, and a Application Programming Interface (API) based on Representational State Transfer (REST) that integrates both systems. The mobile application operates in offline mode, storing collected data locally and allowing synchronization with the server once the connection is restored. The web application provides resources for registering and managing general information, such as users, agents, and clients. The research aimed to demonstrate that the use of this technological integration increases the efficiency of the data collection process and improves information management. The results showed that the prototype is functional, lightweight, and effectively meets the needs of health agents. It is concluded that the proposed solution significantly contributes to modernizing the data collection process, strengthening professionals' performance, and improving the quality of services provided to the elderly population.

Keywords: mobile application; web application; REST API; health agents; data collection; offline synchronization.

1 INTRODUÇÃO

Em diversas regiões remotas e isoladas no Brasil, o acesso à informação é um desafio constante, essas regiões ainda possuem um grande índice de inacessibilidade digital. Segundo, Filho e Gambarato (2023, p.52), no Brasil existem aproximadamente 4,8 milhões de pessoas na faixa etária dos 9 aos 17 anos sem acesso à internet residencial, quando avaliadas regiões rurais vê-se que esse número se potencializa, a partir de tais estatísticas é fato que o acesso a internet é um problema para as famílias dessas regiões.

Atualmente, a expectativa de vida da população tem aumentado, impulsionada por fatores como a redução da taxa de natalidade, mudanças no estilo de vida e avanços na medicina. No Brasil, em 2010, a população idosa correspondia a 10% do total, e a projeção é que, até 2040, esse percentual ultrapasse um quarto da população, refletindo um cenário de envelhecimento demográfico significativo (Ramos et al., 2024).

No que diz respeito à relevância dos agentes de saúde no acompanhamento do idoso, fica evidente que sua atuação contribui significativamente para a melhoria da qualidade do atendimento e para a redução de hospitalizações evitáveis. De acordo com Litzelman et al. (2017), no Estado de Indiana, nos Estados Unidos, a participação ativa dos idosos nas decisões sobre seus próprios cuidados, juntamente com os agentes de saúde locais, teve como resultado nos atendimentos de urgência e internações hospitalares considerável redução.

Em contrapartida, o trabalho desses profissionais é muitas vezes prejudicado pela falta de acesso à internet (Bertollo, 2024), o que faz com que a coleta de dados ainda seja realizada, em muitos casos, por meio de formulários em papel, o que torna o processo mais demorado, custoso e suscetível a erros (Santos; Lima; Freire, 2020).

A transição do uso de papel para uma plataforma digital diminui o uso de recursos físicos e reduz os erros associados à transcrição manual de dados. Além disso, a centralização das informações por meio de uma API REST facilita o compartilhamento de dados entre as equipes de saúde, garantindo que os prontuários dos pacientes sejam atualizados em tempo real sempre que a sincronização for possível, o que melhora significativamente a organização e a eficiência do atendimento (Santos; Lima; Freire, 2020).

A aplicação de soluções móveis offline tem sido considerada em diversos estudos ao longo dos últimos anos. No estudo de Maran et al. (2013), os autores propõem uma arquitetura voltada para facilitar e agilizar o processo de coleta de informações pelos Agentes Comunitários de Saúde (ACS). Essa arquitetura opera de forma híbrida: os dados são inicialmente armazenados localmente, por meio do banco de dados SQLite, e posteriormente sincronizados com um servidor remoto utilizando Web Services que integram um banco de dados PostgreSQL. A arquitetura foi desenvolvida no modelo cliente-servidor e tem como base o modelo RestFul em seus Web Services, sendo específica para dispositivos Android,

A metodologia adotada envolveu o acompanhamento in loco dos agentes em suas atividades rotineiras, preparação para visita domiciliar, cadastro de famílias, visita domiciliar, agendamento e encaminhamento de consulto, com o objetivo de compreender, mapear o fluxo operacional típico e personalizar questiónarios com base na demanda dos agentes de saúde.

A pesquisa de Vieira (2022) aborda o desenvolvimento de uma aplicação móvel voltada ao gerenciamento das vendas de cestas básicas

por empreendedores atuantes em zonas rurais. Diante das dificuldades enfrentadas por essa população, como o acesso limitado a comércios próximos e a conectividade precária, o estudo propõe uma solução tecnológica que funcione de maneira offline. A aplicação foi desenvolvida com o framework Flutter, utilizando a linguagem Dart, o que permitiu a criação de uma solução multiplataforma, compatível com os sistemas Android e iOS. O banco de dados local escolhido foi o SQLite, justamente por oferecer suporte ao uso offline. O processo de desenvolvimento do software é detalhado desde a concepção do negócio até a publicação do aplicativo na Play Store. Além disso, o trabalho inclui um relato de experiência com um usuário real, um empreendedor informal do setor de cestas básicas, o que possibilita compreender melhor os impactos práticos da aplicação proposta.

Sua concepção seguiu uma metodologia estruturada, tendo sido analisados três sistemas relacionados e dois aplicativos móveis semelhantes disponíveis na Google Play Store e na App Store, com o objetivo de compreender soluções já existentes no mercado. Na fase de análise do negócio, foi realizada uma entrevista com um empreendedor individual atuante na comercialização de cestas básicas, a fim de identificar suas principais necessidades relacionadas à tecnologia da informação.

Posteriormente, foram selecionadas as ferramentas tecnológicas mais adequadas para o desenvolvimento do sistema, considerando plataformas, sistemas operacionais, linguagens de programação e banco de dados. O projeto seguiu o modelo de desenvolvimento em cascata, com etapas bem definidas de modelagem, implementação, testes unitários e de integração. Por fim, foi realizada uma avaliação da experiência do usuário, a partir do relato de um microempreendedor que utilizou o aplicativo, proporcionando uma análise prática sobre a usabilidade e efetividade da solução proposta.

Nesse contexto, como surge a proposta de desenvolvimento de um protótipo de aplicação mobile e web utilizando React Native, Expo e NextJs, com um banco de dados local em SQLite e sincronização com uma API REST. O uso da tecnologia mobile oferece vantagens significativas, como a mobilidade, permitindo que os usuários possam acessar seus serviços no próprio local, com um aparelho que de fato já faz parte do cotidiano das pessoas desse século (Brasil; Santos; Ferenhof, 2018), dispondo de recursos offline, que é crucial em regiões onde o acesso à internet é limitado ou inexistente. (Bertollo, 2024).

O React Native é um framework open source criado pelo Face-

book em 2015 para desenvolver aplicativos híbridos usando React.js. Diferente de soluções como o Ionic, que rodam dentro de um WebView, o React Native transforma os componentes escritos em JavaScript em elementos nativos da plataforma. Isso permite que os apps tenham desempenho e aparência similares aos desenvolvidos com código nativo (Bezerra, 2021).

O expo é um framework que tem como objetivo central dar suporte ao desenvolvimento de soluções em React de maneira geral, entretanto ele é grandemente conhecido por seu uso em aplicações baseadas no React-Native. Detem um conjunto abrangente de ferramentas, metódos, bibliotecas que auxiliam no desenvolvimento das aplicações, o que consequentemente otimiza o tempo de desenvolvimento (Falcão, 2022).

O SQLite é uma biblioteca leve que funciona como um banco de dados SQL embutido, ou seja, não exige servidor separado, instalação complexa ou configuração. Ele armazena todas as informações em um único arquivo de disco, o que o torna portátil e ideal para aplicações que exigem simplicidade, como apps móveis e sistemas embarcados. É altamente confiável, amplamente testado, gratuito para uso comercial ou pessoal e utilizado em bilhões de dispositivos (Consortium, 2025). A escolha pelo SQLite como sistema gerenciador de banco de dados neste projeto se deu por suas características, características essas destacadas por Hasnat, Mahato e Halder (2023), que são leveza, simplicidade de uso e compatibilidade com a plataforma Android. Por ser uma solução embarcada, que dispensa a configuração de servidores e funciona com um único arquivo de banco de dados, o SQLite se mostra ideal para aplicações que operam offline ou em ambientes com conectividade limitada (Vieira, 2022).

No que diz respeito a API (Interface de Programação de Aplicações), é um sistema desenvolvido com o objetivo de permitir a integração entre diferentes sistemas, por meio de Web Services com requisições padronizadas responsáveis pelo envio e recebimento de dados. Os protocolos mais utilizados nesse processo são o HTTP e o formato de dados JSON ou XML (Neto, 2022), e sua arquitetura REST que padronizam os recursos de uma aplicação web, promovendo a interoperabilidade entre os componentes (Junior; Rocha; Maciel, 2021).

No nível da gestão de bancos de dados da API, a utilização de um sistema de Mapeamento Objeto-Relacional (*Object-Relational Mapping – ORM*) configura-se como uma solução eficaz para simplificar o desenvolvimento e a manutenção de aplicações. O Prisma, nesse contexto, oferece suporte a operações de criação, leitura, atualização e exclusão (CRUD)

por meio de um esquema gerado com base no banco de dados escolhido. Essa abordagem proporciona maior flexibilidade, permitindo a substituição do sistema gerenciador de banco de dados com alterações mínimas no código da aplicação. Para este trabalho, optou-se pela utilização do PostgreSQL, explorando as funcionalidades avançadas do Prisma com o objetivo de assegurar uma integração eficiente e um gerenciamento robusto dos dados (Lima et al., 2025).

Na solução web, destaca-se o uso do Next.js, um framework de aplicação desenvolvido pela Vercel que amplia as funcionalidades do React, tornando-o mais eficiente e estruturado. O Next.js oferece recursos como Renderização no Lado do Servidor (SSR), que melhora o carregamento das páginas e contribui para o SEO, além da Geração de Sites Estáticos (SSG), ideal para conteúdos pouco dinâmicos. Também simplifica o desenvolvimento por meio de rotas baseadas em arquivos, além de disponibilizar otimizações automáticas de imagens, pré-carregamento de links e divisão de código. Enquanto o React fornece a base para a construção de interfaces, o Next.js agrega produtividade e desempenho ao processo, formando uma tecnologia amplamente indicada tanto para projetos pessoais quanto para aplicações de grande escala (Vercel, 2024).

Nesse sentido, a combinação entre React Native e Expo para o desenvolvimento mobile, juntamente com o Next.js para a aplicação web, possibilita uma solução híbrida e integrada, contemplando tanto a mobilidade em campo dos agentes de saúde e agentes de pesquisa, quanto a gestão e análise centralizada em plataforma web.

Diante de todo esse cenário, este trabalho tem como objetivo geral aperfeiçoar o acesso à informação por parte dos agentes de saúde e agentes de pesquisa por meio da integração de tecnologia mobile com uma aplicação web. Para alcançar esse propósito, busca-se compreender os métodos de elaboração e aplicação de questionários utilizados pelos agentes, validar o modelo proposto, desenvolver um protótipo de API e aplicação mobile, aplicar o conceito de sincronização entre os sistemas e realizar testes que subsidiem uma validação posterior da solução.

O trabalho segue uma estrutura com Introdução, onde é contextualizado o problema bem como suas tecnologias, Materiais e Métodos, onde é explicado o passo a passo do desenvolvimento e pesquisa, Discussão e Resultados e Conclusão.

2 MATERIAIS E MÉTODOS

O desenvolvimento do protótipo de aplicação destinado aos Agentes Comunitários de Saúde (ACS) caracteriza-se como uma pesquisa aplicada, de base tecnológica. O principal objetivo foi aprimorar o acesso à informação por parte desses profissionais, por meio da integração entre tecnologias mobile e web. A solução proposta contempla a funcionalidade de operação em modo offline, aliada a um sistema de sincronização de dados, permitindo que os formulários sejam preenchidos em campo, mesmo sem acesso à internet, e posteriormente sincronizados pelo usuário assim que a conectividade for restabelecida.

2.1 APPLICATION PROGRAMMING INTERFACE (API)

Com o objetivo de centralizar as informações na web, foi desenvolvida uma API hospedada na plataforma Render, a qual fornece suporte a web services e bancos de dados em nuvem, reduzindo a necessidade de gerenciamento de infraestrutura por parte do cliente. Dessa forma, garantiu-se maior escalabilidade e disponibilidade do sistema.

Para a construção da aplicação foram utilizadas dependências que possibilitaram a transpilação do código-fonte em TypeScript para JavaScript e a geração do build final. As bibliotecas empregadas foram instaladas por meio do comando:

```
npm install -D typescript @types/node tsx tsup
```

2.1.1 Base de Dados

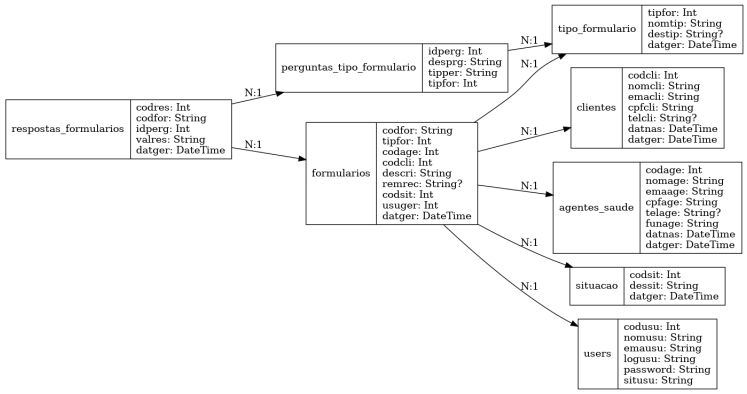
No acesso à base de dados foi utilizado o framework Prisma para a interação com o PostgreSQL, que utiliza o modelo orientado à objetos para a interação com a base, minimizando a escrita de códigos SQL, o schema do banco pode ser visualizado na Figura 1.

Para viabilizar o acesso às tabelas e a realização de operações de manipulação de dados (DML), foi necessária a execução do comando `npx prisma generate`, responsável pela geração do *Prisma Client*, o qual possibilita o correto funcionamento dos métodos implementados na aplicação. Após essa etapa, foi criado o arquivo `.env`, que contém as definições de acesso ao banco de dados (URL) utilizadas no arquivo `schema.prisma`.

No que se refere à criação das *migrations*, foi empregado o comando `npx prisma migrate dev`, que gera as migrações com base nas classes e estruturas definidas no `schema.prisma`. Em ambiente de

produção, fora utilizado o comando `npx prisma migrate deploy`, cuja finalidade é aplicar as migrações de forma definitiva na base de dados em ambientes de produção.

Figura 1 - Diagrama Base de Dados API



Fonte: do Autor.

2.1.2 Controle de Requisições

O Fastify foi adotado para o controle das requisições da interface, desempenhando o papel de controlador de requisições HTTP. Entre suas funcionalidades estão o gerenciamento de rotas, o processamento de dados de entrada e o envio de respostas aos clientes. O mapeamento das rotas ocorre por meio do método `register`, no qual são especificadas as rotas e o caminho dos arquivos que as implementam.

Quanto à segurança das requisições, foi elaborado um *middleware* destinado à verificação de tokens em cada interação com a API. Tais tokens são emitidos no processo de autenticação, na rota de login, utilizando-se a biblioteca `jsonwebtoken`, o que assegura que apenas usuários autenticados possam acessar os recursos da aplicação.

As rotas foram organizadas de forma modular, de modo que cada arquivo concentra as rotas correspondentes ao seu respectivo módulo. Essa abordagem favorece a organização e a manutenção do código a longo prazo. Além disso, nesses mesmos arquivos são realizadas as interações com a base de dados, o que mantém a estrutura do sistema mais coesa e compreensível.

2.1.3 Deploy da Plataforma Render

Para o deploy da aplicação, inicialmente foi criado um banco de dados PostgreSQL disponibilizado pela própria plataforma Render, onde são definidas as configurações básicas, como nome e plano. Considerando

que o objetivo do projeto foi o desenvolvimento de um protótipo, optou-se pelo plano gratuito, que apresenta algumas limitações, como a expiração periódica dos dados e um alto tempo de espera em sua primeira requisição após 15 minutos de inatividade.

Após a criação do banco, a Render fornece uma URL externa de acesso, a qual deve ser vinculada às variáveis de ambiente do serviço web. Para esse fim, foi criada a variável DATABASE_URL. A criação do serviço web segue a mesma lógica de configuração do banco de dados, em que são definidos parâmetros básicos, como nome e demais informações necessárias para a execução do serviço.

2.2 APLICAÇÃO WEB

A aplicação web foi desenvolvida utilizando o framework Next.js, o qual é baseado em React. Nesse ambiente foram implementadas as funcionalidades de cadastro dos módulos utilizados no aplicativo, bem como o cadastro dos formulários propriamente ditos. Além disso, foi incorporado o recurso de exportação dos dados referentes aos formulários aplicados.

O projeto foi iniciado utilizando o comando `npx create-next-app`, durante a execução, algumas perguntas são apresentadas, incluindo a escolha da linguagem de programação e da estrutura de pastas. Para o desenvolvimento, foi selecionado o JavaScript, e a organização do projeto seguiu a estrutura baseada na pasta `src`, que funciona como a raiz do projeto, abrigando as rotas, componentes e demais configurações essenciais.

2.2.1 Gerenciamento de Acesso

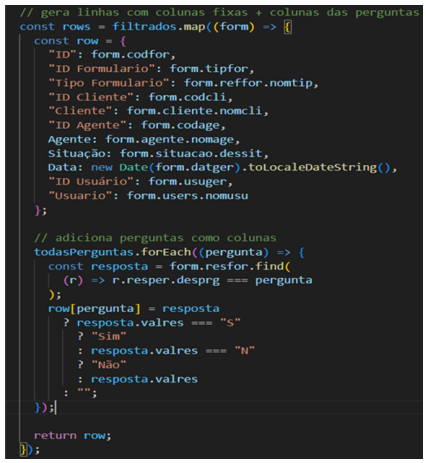
O controle de acesso na aplicação foi estruturado por meio de funções de verificação, sendo a principal a função `verify`. Esta função realiza, a cada acesso a uma página, uma requisição à API com o objetivo de validar o status do token do usuário. Caso o token tenha expirado, o usuário é automaticamente redirecionado para a tela de login. Os tokens emitidos pela API no momento do login são armazenados no `localStorage` do cliente e recuperados utilizando o método `getItem`.

Para garantir consistência em todas as operações de CRUD da aplicação, foi desenvolvido o arquivo `fetchWithAuth.js`, responsável por padronizar as requisições, incluindo a configuração dos cabeçalhos de autorização e das credenciais necessárias para a comunicação com a API.

2.2.2 Exportação de Formulários

A exportação dos formulários aplicados foi implementada utilizando a biblioteca SheetJS, que permite gerar arquivos em formato de planilha (Excel) a partir de estruturas de objetos em JavaScript que em seguida são convertidos em planilhas pelo SheetJS (Figura 2).

Figura 2 - Função Exportação Formulários



Fonte: Do Autor.

2.3 APLICATIVO MÓVEL

O aplicativo móvel foi desenvolvido com o *Expo*, com foco em oferecer uma solução multiplataforma que funcionasse de forma integrada à API e, ao mesmo tempo, garantisse suporte ao modo offline. Para a criação do projeto em expo utilizou-se o comando `npx create-expo-app`, onde são definidas as pastas e estruturas iniciais do projeto.

2.3.1 Base de Dados

A definição do banco de dados offline, que em grande parte reflete a estrutura do banco da API, foi implementada a partir do arquivo `migrations.js`. Esse arquivo é executado durante a inicialização do aplicativo e contém os comandos responsáveis pela criação do banco e pela definição das tabelas locais. Para isso, utiliza-se a biblioteca expo-sqlite, que disponibiliza o método `execAsync`, por meio do qual é possível executar os blocos de código necessários para a definição da base. A estrutura das tabelas pode ser visualizada na Figura 3.

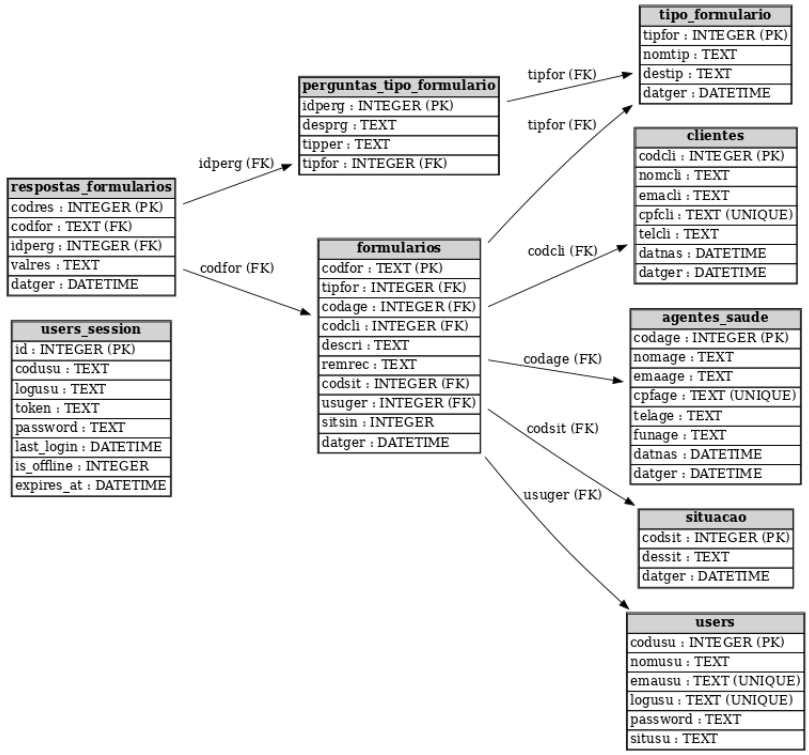
Na tabela `users session` são registradas as sessões dos usuários, possibilitando o controle de autenticação durante o uso do aplicativo em modo offline. Já o campo `sitsin`, presente na tabela `formularios`, é responsável pela identificação do status de sincronização dos formulários,

indicando se os registros já foram enviados para a API ou permanecem apenas no armazenamento local.

As interações com o banco foram organizadas em arquivos de service para cada módulo. Nessas funções estão centralizadas as operações que são exportadas para utilização nas rotas da aplicação. Por meio delas, são definidos o método de acesso, a ação (INSERT, UPDATE, SELECT ou DELETE), o tipo de consulta e a origem da chamada, garantindo maior modularidade e separação de responsabilidades no aplicativo.

Para comandos de alteração de dados foi empregado o método `runAsync`, enquanto as consultas foram realizadas com o método `getAllAsync`, utilizado para a recuperação de múltiplos registros, e o método `getFirstAsync`, aplicado em buscas específicas por identificador.

Figura 3 - Diagrama Banco Offline



Fonte: Do Autor.

2.3.2 Requisições da API

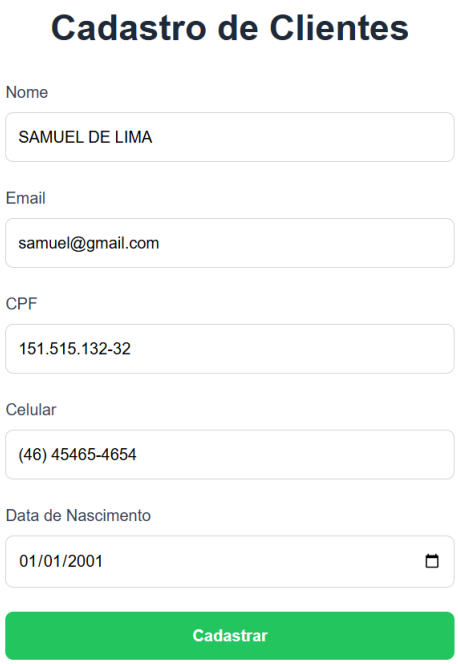
As requisições foram estruturadas de maneira modular, considerando os parâmetros de método, os dados transmitidos, o tipo de ação (consulta geral ou específica) e, quando aplicável, o identificador do registro. Para garantir a padronização dos cabeçalhos e do corpo das requisições, desenvolveu-se a função `fetchWithAuth`, a qual centraliza o tratamento

das credenciais de autenticação.

2.3.3 Sincronização do APP X API

As informações básicas são registradas na aplicação web por meio de módulos específicos, incluindo clientes, agentes, usuários, situações e tipos de formulários, na Figura 4 pode ser visualizada a interface do cadastro de clientes.

Figura 4 - Cadastro de Cliente - WEB

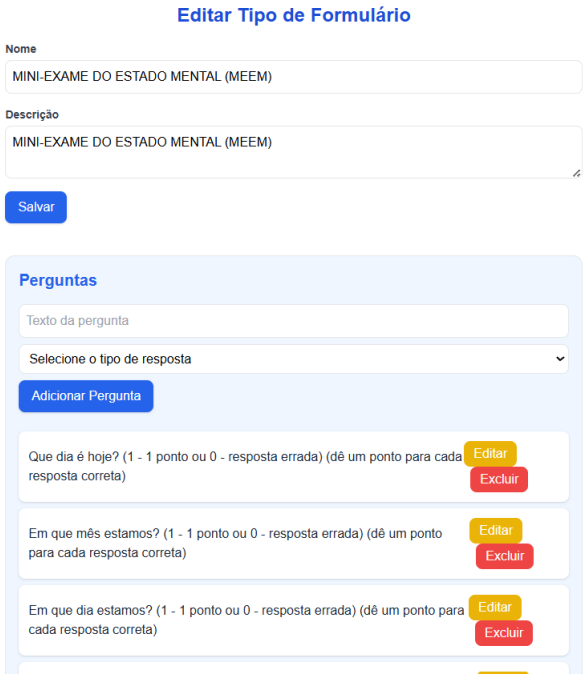


Fonte: Do Autor.

O cadastro de clientes segue a mesma lógica adotada nos demais módulos, porém o módulo de tipos de formulários apresenta características específicas. Conforme ilustrado na Figura 5, inicialmente são inseridas as descrições do formulário e, posteriormente, as perguntas, que podem ser configuradas como texto livre ou respostas do tipo "Sim/Não", permitindo maior flexibilidade na coleta de informações.

Para garantir a sincronização adequada entre o aplicativo e a API, foram desenvolvidos métodos específicos que podem ser executados sempre que o usuário acessa a tela inicial do sistema, através do botão sincronizar dados. Inicialmente, é realizada a verificação da conexão de rede por meio da biblioteca `NetInfo`, disponibilizada pelo pacote `react-native-community`, a qual fornece o método `isConnected`, responsável por retornar um valor booleano de acordo com o status da conexão.

Figura 5 - Edição de Tipo Formulário - WEB



Fonte: Do Autor.

Após confirmada a disponibilidade da rede, o botão de sincronização é habilitado. Ao ser acionado, o sistema executa automaticamente uma sequência de rotinas responsáveis por atualizar os dados locais do aplicativo com as informações mais recentes provenientes da API. Esse processo é ilustrado na Figura 6.

Figura 6 - Home APP - Mobile

Fonte: Do Autor.

Cada função de sincronização segue um modelo padronizado: inicialmente, realiza a verificação dos registros armazenados localmente e os compara com os dados da API. Caso determinado registro não exista mais na API, este é removido da base local (com exceção dos formulários ainda não sincronizados, que são preservados). Em seguida, é feita

uma requisição `GET` ao respectivo módulo (Clientes, Agentes, Usuários, Formulários etc.), percorrendo os registros retornados. Para cada item, é consultado o service do banco local a fim de verificar se já existe um registro correspondente; se existir, é acionada a função de atualização, caso contrário, um novo registro é criado com base nos dados da API.

Além da sincronização ao acessar a tela inicial, foi implementada uma segunda forma de sincronização vinculada especificamente à tela de atualização de formulários. Nessa abordagem, o usuário dispõe de um botão dedicado para sincronizar, individualmente os formulários desejados, foi gerado a partir de um componente `BotaoSync` (com paramêtros como método, tabela e dados).

O botão é exibido apenas havendo conexão com a internet, após essa verificação é realizada uma requisição à API de maneira semelhante às funções de sincronização geral, porém restrita apenas ao formulário selecionado. Durante esse processo, os dados do formulário são enviados para a API e, após o retorno bem-sucedido, o registro correspondente na base local é atualizado, alterando o campo `sitsin` para o valor 2, através do services de atualização de formulários, indicando que o formulário foi devidamente sincronizado.

2.3.4 Geração do Build da Aplicação

Para a etapa de geração do build da aplicação, foi inicialmente realizada a instalação do *EAS CLI*, ferramenta que possibilitou a interação com os serviços disponibilizados pelo EAS. A instalação foi efetuada por meio do comando `npm install -g eas-cli`. Em seguida, realizou-se a autenticação utilizando uma conta Expo, habilitando o acesso às funcionalidades de compilação. Concluídas essas etapas, o build foi gerado por meio do comando `npx eas build -p android -profile preview`.

2.4 MÉTRICAS DE AVALIAÇÃO

Para a avaliação da usabilidade e da satisfação dos usuários em relação ao aplicativo desenvolvido, foi utilizado um questionário baseado na Escala de Likert, juntamente com questões descritivas. Essa escala é amplamente empregada em pesquisas científicas por permitir a quantificação de percepções e opiniões subjetivas, facilitando a análise estatística de dados qualitativos.

A Escala de Likert consiste em uma série de afirmações nas quais os participantes indicam seu grau de concordância ou satisfação em

uma escala ordinal de cinco pontos, como 1 (Muito Ruim) a 5 (Excelente). Essa metodologia possibilita captar nuances nas respostas dos usuários, indo além de avaliações dicotômicas (Likert, 1932).

O questionário contemplou itens relacionados à facilidade de uso, funcionalidade, design da interface, eficiência e satisfação geral com a aplicação. As respostas obtidas foram posteriormente tabuladas e analisadas de forma descritiva, permitindo identificar aspectos positivos e pontos de melhoria no sistema, como apresentado no Apêndice A. O questionário foi aplicado na própria plataforma web.

3 RESULTADOS E DISCUSSÃO

Os resultados obtidos neste trabalho foram gerados a partir de questionários baseados na escala de Likert, abrangendo aspectos como usabilidade, desempenho, integração e satisfação geral com o sistema (aplicativo móvel e aplicação web), foram aplicados à 5 participantes, agentes de pesquisa do DNG - Grupo de pesquisa em Doenças Neurodegenerativas, do Laboratório de Neurologia Experimental - UNESC, que possuem diferentes níveis de familiaridade com tecnologias digitais, onde foram realizados cadastros de formulários pelo aplicação móvel e web. Além de avaliarem o sistema por meio da própria interface web.

Na aplicação web, observou-se que 90% dos participantes da avaliação atribuíram a nota máxima para o quesito usabilidade, enquanto o desempenho obteve 100% de avaliações com nota máxima. Já em relação à aplicação móvel, verificou-se que 94% dos participantes concordaram totalmente e 6% concordaram parcialmente quanto à usabilidade e desempenho. Os resultados detalhados podem ser visualizados nas Tabelas 1 e 2.

Tabela 1 - Questionário Aplicação WEB

	Usabilidade	**Desempenho**
Discordo Totalmente	0%	0%
Discordo	0%	0%
Neutro	0%	0%
Concordo	10%	0%
Concordo Totalmente	90%	100%

Fonte: Elaborado pelo autor.

Tabela 2 - Questionário Aplicação Móvel

	Usabilidade	Desempenho
Discordo Totalmente	0%	0%
Discordo	0%	0%
Neutro	0%	0%
Concordo	6%	6%
Concordo Totalmente	94%	94%

Fonte: Elaborado pelo autor.

Considerando o nível geral de satisfação com as ferramentas, identificou-se que 80% dos participantes marcaram “Concordo totalmente” e 20% “Concordo”, indicando um alto grau de aprovação do sistema desenvolvido. No que se refere à recomendação da ferramenta para outros projetos, 80% dos respondentes afirmaram que recomendariam o uso da aplicação, enquanto 20% indicaram que talvez a recomendariam, o que demonstra boa aceitação geral entre os usuários.

A média geral das avaliações foi de 4,92, indicando um alto nível de aceitação e satisfação por parte dos usuários. O desvio-padrão geral obtido foi de 0,10, demonstrando que as respostas apresentaram baixa dispersão, ou seja, uma percepção homogênea e consistente entre os avaliadores em relação ao desempenho e à experiência de uso da aplicação.

De modo geral, os participantes não relataram dificuldades no uso do aplicativo móvel ou da aplicação web, destacando a boa funcionalidade, usabilidade e sincronização offline. As sugestões de melhoria concentraram-se em ajustes pontuais, como correções de nomenclaturas, inclusão de novos campos cadastrais (ex.: “sexo”), exportação simultânea de questionários e possibilidade de organizar projetos em abas separadas. Também foram citadas ideias para aumentar a agilidade e segurança, como alertas de sincronização e otimização do preenchimento de planilhas.

O tempo de resposta da aplicação web foi considerado adequado pela maioria dos participantes, tendo em média 195 ms. Entretanto, observou-se uma limitação decorrente do uso do plano gratuito da plataforma Render, responsável pela hospedagem do sistema. Nesse plano, a aplicação é colocada em modo de suspensão (sleep) após 15 minutos de inatividade, resultando em um atraso na primeira solicitação subsequente. Segundo a documentação da Render, esse processo de reativação pode levar até 60 segundos, o que impacta o tempo de resposta apenas na primeira requisição após o período de ociosidade (Render, 2025).

No que se refere à geração dos pacotes de instalação, o aplica-

tivo móvel foi disponibilizado apenas para o sistema operacional Android. A geração do arquivo executável para iOS não foi realizada devido às restrições impostas pela Apple, que exigem a utilização de um ambiente macOS e a vinculação a uma conta de desenvolvedor paga para a assinatura e distribuição do aplicativo. Segundo a documentação oficial do Expo (Expo, 2025), o processo de compilação para iOS requer certificados digitais e perfis de provisionamento gerenciados pela Apple, o que inviabilizou a publicação nesta plataforma durante o desenvolvimento deste trabalho. No entanto o aplicativo móvel pode ser executado no IOS por meio da plataforma Expo Dev (em ambiente de desenvolvimento).

Ao comparar com o estudo de Maran et al. (2013), observa-se que o aplicativo analisado pelos autores exigia conexão contínua com a internet para o funcionamento adequado e o carregamento de módulos, como mapas, além de utilizar formulários pré-definidos. Em contraste, o aplicativo desenvolvido neste trabalho adota uma abordagem mais flexível e adaptável, permitindo cadastros remotos e sincronização com a API.

A arquitetura dos formulários foi projetada de forma dinâmica, possibilitando que o agente de saúde, por meio da interface web, configure perguntas e tipos de resposta (texto livre, “Sim/Não” ou múltipla escolha). Essa característica amplia a autonomia e a capacidade de personalização do sistema, tornando-o mais adequado a diferentes contextos de coleta de dados e pesquisa em campo.

O estudo de Vieira (2022) apresentou uma abordagem voltada para um aplicativo offline que demonstrou atender adequadamente às necessidades do usuário. Entretanto, o armazenamento das informações ocorria apenas de forma local, o que gerava preocupação quanto à perda de dados em caso de danos ao dispositivo ou desinstalação do aplicativo. Como trabalho futuro, o autor destacou a implementação do armazenamento em nuvem, visando à centralização e maior segurança das informações.

4 CONCLUSÃO

Este trabalho teve como objetivo geral aprimorar o acesso à informação por parte dos agentes comunitários de saúde, por meio da integração entre uma aplicação móvel e uma aplicação web, vinculadas a uma API REST. O protótipo desenvolvido demonstrou-se eficaz ao incorporar mecanismos de armazenamento local e em nuvem, garantindo o funcionamento mesmo em regiões com acesso limitado à internet e permitindo a

sincronização dos dados assim que a conectividade é restabelecida.

A aplicação proposta contribui significativamente para a modernização e a eficiência do trabalho dos agentes de saúde, oferecendo maior autonomia e praticidade na coleta, registro e transmissão das informações. O uso de questionários dinâmicos e personalizáveis permite adaptar o sistema às necessidades específicas de cada comunidade atendida, tornando o processo de atendimento mais ágil, confiável e humanizado. Para os pacientes em áreas remotas, a solução representa um importante avanço na continuidade do cuidado, pois assegura o registro seguro dos dados e possibilita o acompanhamento mais preciso das condições de saúde, mesmo em locais com infraestrutura precária de comunicação.

Entre as limitações observadas, destaca-se o tempo de inatividade da aplicação, que ocasiona pequena lentidão na primeira requisição após períodos prolongados sem uso. Apesar disso, a performance geral mostrou-se satisfatória dentro dos objetivos propostos. Inicialmente, também não foram efetuadas tratativas para permissão de rotas com base no usuário de acesso, sendo estes aspectos passíveis de otimização e alteração em versões futuras.

Como perspectivas para trabalhos futuros, recomenda-se a ampliação dos estudos voltados à avaliação do impacto da ferramenta no cotidiano dos agentes de saúde e na qualidade dos dados coletados. Sugere-se, ainda, a realização de testes em larga escala, abrangendo diferentes realidades socioeconômicas e regionais, de modo a validar a robustez e a aplicabilidade da solução em contextos diversos. Por fim, propõe-se o aprofundamento em técnicas de análise de dados e inteligência artificial que possam subsidiar tomadas de decisão mais assertivas e fortalecer as políticas públicas direcionadas à atenção básica em saúde.

REFERÊNCIAS

BERTOLLO, M. **O imprescindível acesso à internet no campo: a capilarização da informação no brasil e na frança**. Confins. Revue franco-brésilienne de géographie/Revista franco-brasilera de geografia, 2024, Théry, Hervé,

BEZERRA, F. S. R. **Desenvolvimento nativo vs ionic vs react native: uma análise comparativa do suporte à acessibilidade em android**.

BRASIL, S. B.; SANTOS, B. P. dos; FERENHOF, H. A. **Mobile learning: um estudo exploratório sobre aprendizagem com mobilidade no brasil**. International Journal of Knowledge Engineering and Management, 2018, v. 7, n. 19,

CONSORTIUM, S. **About SQLite**. 2025. Acesso em: 19 abr. 2025. Disponível em: <https://sqlite.org/about.html>.

EXPO. **Building for iOS — Expo Documentation**. 2025. Acesso em: 10 nov. 2025. Disponível em: <https://docs.expo.dev/tutorial/eas/ios-development-build-for-devices/>.

FALCÃO, F. D. **Desenvolvimento do aplicativo Turistando Beberibe utilizando React Native**. Fortaleza: Universidade Federal do Ceará, 2022. Trabalho de Conclusão de Curso.

FILHO, W. de J. F.; GAMBARATO, V. T. S. **Aplicação de aula remota durante a pandemia de covid-19: Um estudo de caso**. Tekhne e Logos, 2023, v. 14, n. 1,

HASNAT, A.; MAHATO, N.; HALDER, S. **Basics of Android Application Development**. Berhampore, West Bengal, India: Government College of Engineering and Textile Technology, 2023.

JUNIOR, E. A. G.; ROCHA, R. D.; MACIEL, R. de S. **Desenvolvimento de api rest com spring boot**. [S.l.]: Revista Científica do UniRios, 2021.

LIKERT, R. **A technique for the measurement of attitudes**. Archives of Psychology, 1932, v. 22, n. 140,

LIMA, J. L. A. d. et al. **Projeto web ceua-implementação de melhorias com foco em qualidade de software**. 2025, Universidade Federal do Ceará, Sobral.

LITZELMAN, D. K. et al. **Impact of community health workers on elderly patients' advance care planning and health care utilization: moving the dial**. Medical Care, 2017, LWW, v. 55, n. 4,

MARAN, V. et al. **Uma arquitetura movel para auxílio as atividades do programa de agentes comunitários da saúde do sus**. 2013, Simpósio Brasileiro de Tecnologia da Informação, Recife, Pernambuco.

NETO, R. H. **API de leitura, compressão e disponibilização de dados para business intelligence**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2022.

RAMOS, M. R. et al. **Demência em idosos: um relato de caso**. Brazilian Journal of Health Review, 2024, v. 7, n. 5,

Render. **Deploy for Free – Render Docs**. 2025. <https://render.com/docs/free>. Acesso em: 30 out. 2025.

SANTOS, A. B. C. dos; LIMA, H. S.; FREIRE, S. P. **Modernização dos registros de visita domiciliar**. Semana da Diversidade Humana (ISSN: 2675-1127), 2020, v. 3,

Vercel. **Next.js: What is Next.js?** [S.l.], 2024. Acesso em: 24 abr. 2024. Disponível em: <https://nextjs.org/docs#what-is-nextjs>.

VIEIRA, C. C. **Análise e desenvolvimento de uma aplicação móvel offline para gerenciamento das vendas de cestas básicas em zonas rurais.** 2022, Universidade Federal de Ouro Preto, João Molevade.

A APÊNDICE - QUESTIONÁRIO APLICADO

Este apêndice apresenta o questionário utilizado para a coleta de dados junto aos participantes da pesquisa.

1. O tempo de resposta da aplicação web é satisfatório. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

2. As funcionalidades do aplicação web e do aplicativo móvel estão integradas de forma eficiente. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

3. Qual o seu nível geral de satisfação com a ferramenta (móvel + web)? 1 – Muito insatisfeito 2 – Insatisfeito 3 – Neutro 4 – Satisfeito 5 – Muito satisfeito

4. Você recomendaria o uso dessa ferramenta para outros projetos de pesquisa? 1 - Sim 2 - Talvez 3 - Não

5. Quais dificuldades ou limitações você percebeu ao usar o app ou a aplicação web?

6. Que melhoria você considera mais importante para o app ou a aplicação web?

7. Há alguma funcionalidade adicional que você gostaria de ver implementada?

8. O aplicativo móvel é fácil de usar durante as visitas domiciliares. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

9. A interface do aplicativo móvel (menus, botões, ícones) é clara e intuitiva. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

10. O tempo de preenchimento dos formulários é adequado à rotina de trabalho. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

11. O aplicativo móvel funciona corretamente em modo offline e sincroniza os dados quando há conexão. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

12. O desempenho do aplicativo móvel (velocidade, estabilidade) atende às expectativas. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

13. A aplicação web é fácil de navegar e possui uma organização lógica das informações. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente

14. Os relatórios e estatísticas disponíveis facilitam o acompanhamento das coletas. 1 – Discordo totalmente 2 – Discordo 3 – Neutro 4 – Concordo 5 – Concordo totalmente