Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Projeto de Extensão

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA MARCAÇÃO DE CONSULTAS

Cláudio Antônio Gomes[1]
Diego Aragão Dutra Borges[1]
Franciane Aprigio de Oliveira Castro
Giácopo Teixeira Campos
Gustavo Andrade Medeiros
Marília roberto da Cruz Carvalho
Maurício Reis Filho
Rafael Reis de Oliveira
Romualdo Monteiro de Resende Costa[2]
Centro Universitário Academia, Juiz de Fora, MG

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta para agendamento de consultas, sob a forma de um aplicativo móvel, desenvolvido utilizando a biblioteca *React Native*. Esta ferramenta possui várias funcionalidades, além da possibilidade de marcar e desmarcar as consultas, que incluem a validação do cadastro dos usuários, bem como a indicação de clínicas e profissionais para a realização dos procedimentos de saúde. Cada uma dessas funcionalidades está interligada a um servidor remoto, onde está disponibilizado um banco de dados que é acessado através de uma interface construída especificamente para este projeto. Os registros de acesso ficam armazenados para posterior análise de demanda e gerenciamento do aplicativo.

Palavras-chave: Marcação de Consultas, Aplicações móveis. Saúde. React Native.

[1] Discente do Curso de Engenharia de Software, do Centro Universitário Academia.
[2] Docente do Curso de Engenharia de Software, do Centro Universitário Academia. Orientador.

1 INTRODUÇÃO

Atualmente, os usuários, para realizar as suas atividades cotidianas, fazem uso de várias aplicações computacionais, muitas vezes construídas especificamente para este fim. Através do uso dessas aplicações, os usuários, frequentemente, conseguem realizar suas atividades com maior facilidade e independência. Considere, como exemplo, o uso de aplicações de comércio eletrônico. Através do uso dessas aplicações, os usuários podem escolher o item desejado, muitas vezes pesquisando em diferentes fornecedores, realizar o pedido e receber o item sem a necessidade de comparecer presencialmente às lojas. Adicionalmente, quando essas aplicações são projetadas para funcionar em dispositivos móveis, a disponibilidade do serviço oferecido aumenta, pois os usuários, através dos seus celulares, podem acessar o serviço em qualquer lugar e a qualquer momento. Nesse contexto, o serviço torna-se disponível, inclusive, para os usuários sem acesso aos computadores tradicionais.

Dessa forma, no geral, as aplicações móveis têm a capacidade de facilitar o acesso aos usuários aos mais diferentes serviços. Em particular, quando se trata de serviços essenciais, o favorecimento de acesso aos serviços se torna ainda mais importante. Este é o caso, por exemplo, dos serviços relacionados à saúde. No geral, pode-se dizer que a acessibilidade à saúde é um componente fundamental para o bem estar de uma sociedade, desempenhando um papel importante na promoção da equidade e na melhoria da qualidade de vida das pessoas.

Em (RICCIARDI, 2019) os autores destacam que a acessibilidade aos cuidados de saúde está intrinsecamente ligada à busca por serviços preventivos. Segundo esse estudo, indivíduos que enfrentam menos barreiras para acesso a serviços de saúde são mais propensos a procurar exames regulares, vacinações a adotar práticas saudáveis contribuindo, assim, para a prevenção de doenças em toda a sociedade.

Outra pesquisa (YANG, 2020) destaca a relação entre a facilidade de acesso à saúde e a eficácia do tratamento. Pacientes que enfrentam menos obstáculos no acesso a serviços médicos são mais propensos a aderir ao tratamento, realizar acompanhamentos regulares e, assim, conseguem alcançar melhores resultados.

Além disso, em (WARNER, 1979), os autores destacam as implicações econômicas relacionadas a facilitação do acesso à saúde. Populações com maior acesso a serviços médicos têm menor incidência de doenças crônicas não tratadas,

resultando, entre outras coisas, em menor absenteísmo no trabalho, o que contribui para uma economia mais produtiva.

Por tudo isto, a facilidade de acesso à saúde não é apenas um benefício individual, mas também desempenha um papel significativo na construção de uma sociedade saudável e economicamente produtiva. Nesse contexto, este trabalho tem por objetivo oferecer uma contribuição social através de uma ferramenta que permita o agendamento de procedimentos de saúde, facilitando o acesso a esses serviços, entre outras facilidades, que incluem o oferecimento de informações de localização de clínicas e consultórios médicos. Tudo isto através de uma aplicação móvel que pode ser acessada através dos dispositivos móveis como celulares. Dessa forma, esta ferramenta constitui parte da implementação de políticas e práticas que visam a redução de barreiras ao acesso a saúde pela população.

Para o desenvolvimento deste trabalho, entre outras tecnologias, foi utilizada a a biblioteca *React Native* (EISENMAN, 2018), que oferece um conjunto de componentes *Javascript* (FLANAGAN, 2020). A próxima seção apresenta algumas características dessa biblioteca, importantes para o desenvolvimento deste trabalho e que, assim, justificam sua escolha.

A seguir, a terceira seção apresenta os recursos da ferramenta desenvolvida. A quarta seção apresenta a parte da implementação que permitiu a construção da ferramenta, com seus diferentes elementos. Finalmente, os resultados alcançados com discussões correlatas são encontrados nas últimas seções, onde são apresentadas as conclusões, bem como os possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

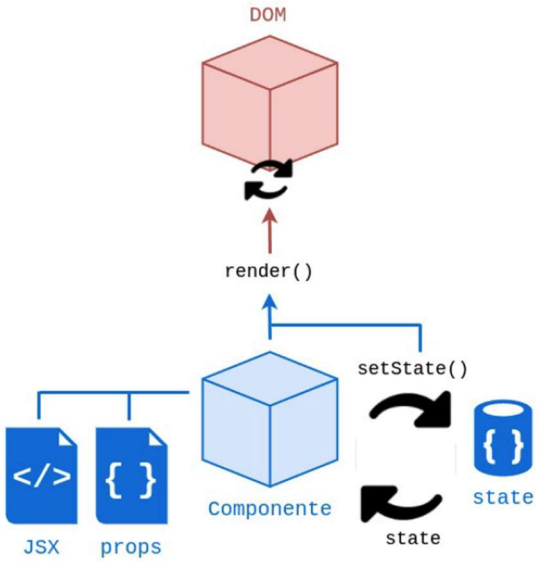
2.1 React Native

React Native é uma biblioteca para construção de aplicativos móveis baseada no *framework* definido originalmente pela biblioteca *React* (BANKS, 2017). Assim, as especificações básicas do *React Native* são as mesmas definidas no *React*, isto é, as aplicações são escritas de forma declarativa, seja usando as especificações funcionais do *Javascript*, ou usando as especificações declarativas do *JSX*[3]. De forma similar, o ciclo de vida das aplicações é o mesmo, isto é, o *framework* para a construção das aplicações permanece inalterado. A Figura 1 apresenta a

[3] https://facebook.github.io/jsx

representação básica desse framework, com destaque para seus elementos. Cabe destacar que a atualização dos componentes, no caso do *React Native*, é realizada diretamente na plataforma de execução, como será visto adiante.

FIGURA 1: Elementos de um Componente *React Native* definidos pelo Framework



Fonte: PONTES, 2018.

Na Figura 1, o método *render* aparece em destaque. Esse é o único método obrigatório em todo componente. Quando o componente é construído, esse método é chamado e o seu conteúdo define a forma de apresentação desse componente. Além de ser executado no processo de construção do componente, esse método também deve ser executado por ocasião de modificações sobre o seu estado. O estado, definido também em destaque na Figura 1 (*state*), representa um conjunto de valores que, toda vez que são alterados, provocam, novamente, a execução do método *render*. Para destacar as alterações no estado, um método especial, conhecido como *setState*, normalmente é empregado.

Além do *render*, o ciclo de vida dos componentes do *React Native* é completado pelos métodos apresentados na Tabela 1.

TABELA 1 - Métodos do Ciclo de Vida da Construção de um Componente *React Native*

Método	Contexto
constructor()	*Método invocado quando o componente é montado*
getDerivedStateFromProps()	*Método invocado antes da renderização*
render()	*Método invocado toda vez que o estado muda*
componentDidMount()	*Método invocado depois da renderização*
componentWillUnmount()	*Método invocado quando o componente é removido*

Fonte: do autor.

O método construtor (*constructor*) é usado para a inicialização do componente. É nesse método que o estado é inicializado. Esse método tem acesso as propriedades do componente e, se necessário, pode utilizar esses valores para definir seu estado inicial. Logo após a obtenção das propriedades e inicialização do estado, o método *getDerivedStateFromProps* é executado. Esse método é chamado antes da apresentação do componente e pode ser utilizado, por exemplo, para a requisição de conteúdos e configurações adicionais que precisem ser realizadas antes da apresentação. Também é possível, ainda nesse método, realizar modificações sobre o estado. Esse método recebe o estado como um argumento e retorna um objeto com as mudanças no estado.

Após a apresentação do componente, o método *componenteDidMount* é chamado. Esse método pode ser utilizado para fazer requisições de conteúdos, porém, toda vez que o estado for modificado nesse método o ciclo de vida será recomeçado e o componente novamente apresentado. Finalmente, o método *componentWillUnmount* é chamado antes do componente ser removido.

Voltando à Tabela 1, os métodos nela apresentados são chamados em decorrência de ações no ciclo de vida de um componente durante a sua construção. Atualizações nesse componente, por outro lado, decorrentes de mudanças no seu estado ou devido ao recebimento de novas propriedades definem a chamada dos métodos de atualização, que são apresentados na Tabela 2.

TABELA 2. Métodos do Ciclo de Vida da Atualização de um Componente *React Native*

Método	Contexto
getDerivedStateFromProps()	*Método inicial invocado quando o componente é atualizado*
sholdComponentUpdate()	*Método invocado antes da renderização*
render()	*Método invocado toda vez que o estado muda*
getSnapshotBeforeUpdate()	*Método invocado depois da renderização*
componentDidUpdate()	*Método invocado quando o componente é removido*

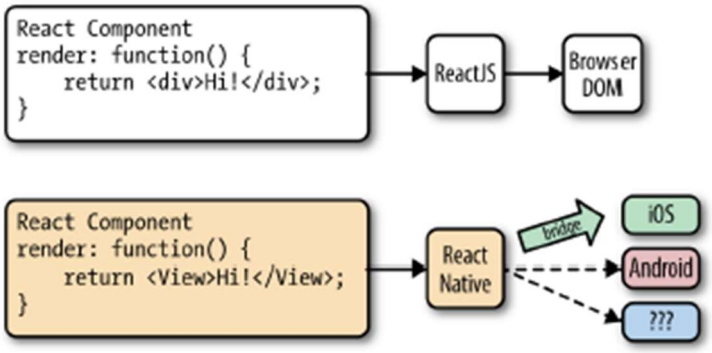
Fonte: do autor.

Quando um componente é atualizado, o primeiro método a ser chamado é o *getDerivedStateFromProps*. Esse método recebe o estado e as propriedades de um componente e retorna as modificações que forem necessárias. Nesse método é possível realizar modificações necessárias para preservar valores no estado de um componente, apesar das modificações realizadas. Após a atualização, mas ainda antes da apresentação, o método *sholdComponentUpdate* é chamado. Esse método, quando usado, pode retornar um valor booleano que especifica se o componente deve

ser atualizado, sendo o valor padrão verdadeiro. O método *getSnapshotBeforeUpdate* é utilizado nos casos em que é necessário acessar os valores das propriedades e do estado de um componente antes da atualização. Assim, através desse método, mesmo após uma atualização, é possível verificar os valores anteriores do estado do componente. Finalmente, o método *componentDidUpdate* é chamado após a apresentação do componente, isto é, depois que o componente foi novamente apresentado a partir de modificações no seu estado.

O método *render*, apresentado nas Tabelas 1 e 2, é o único, entre todos esses métodos, obrigatório na concepção do componente. Esse método, como já mencionado, realiza a apresentação visual do componente. Particularmente, no caso do *React Native*, a apresentação visual é realizada por componentes próprios dessa biblioteca, como representado na Figura 2. Na parte superior dessa figura é representada a apresentação tradicional do *React*, onde os componentes são apresentados no navegador Web. Na parte inferior, por outro lado, é representada a apresentação dos componentes em diferentes plataformas, como acontece no React Native.

FIGURA 2. Comportamento da Apresentação da Aplicação React Native em relação ao React.



Fonte: EISENMAN, 2018.

Como apresentado na Figura 2, as modificações nos componentes da aplicação *React Native*, através de uma ponte (*bridge*, na Figura 2), são traduzidas em modificações para a apresentação da plataforma alvo. Dessa forma, aplicações do *React Native* podem ser apresentadas em qualquer plataforma, desde que seja construída uma interface intermediária que realize a tradução das modificações do componente considerado para as representações na plataforma destino.

Uma vez que as especificações dos componentes no *React Native* podem ser convertidas em apresentações diversas dependendo da plataforma de destino, ao

invés de especificar elementos do HTML para apresentação, como é o caso do *React*, nessa outra biblioteca são definidos elementos com semântica de apresentação própria (EISENMAN, 2018). A Tabela 3 apresenta, como exemplo, alguns desses elementos.

TABELA 3 - Alguns Elementos definidos no *React Native* e sua equivalência com HTML.

Componente	HTML	Objetivo
<View>	* *	*Organizar a disposição dos elementos*
<Text>	**	*Especificar informações textuais*
<FlatList>	* *	*Criar uma lista com informações*
<Image>	*<img>*	*Apresentar uma imagem*
<TextInput>	*<input>*	*Receber entrada de dados dos usuários*

Fonte: do autor.

Para organizar a apresentação dos componentes, o *React Native* define o elemento <*View*>. Esse elemento funciona de maneira semelhante ao <*div*> do HTML permitindo organizar a representação visual dos seus elementos relacionados. Durante a apresentação da aplicação esse elemento deverá ser traduzido para um elemento nativo da plataforma. No caso do sistema operacional *Android* (LECHETA, 2015), por exemplo, esse elemento será traduzido para o elemento homônimo <*View*>. Já no caso do sistema *iOS* (iOS, 2021), por outro lado, esse elemento será traduzido para o componente *UIView*.

A Tabela 3 apresenta, como exemplo, alguns outros elementos do *React Native*, como o elemento <*Text*> que permite a apresentação de conteúdos textuais, o elemento <*FlatList*> para a construção e apresentação de listas. O elemento <*Image*>, para a apresentação de imagens e o elemento <*TextInput*> que permite ao usuário preencher um campo com informações no formato texto. Diversos outros elementos são oferecidos por essa biblioteca (EISENMAN, 2018).

Cada um dos elementos apresentados, bem como os demais elementos existentes na biblioteca *React Native* são implementados como componentes. Essa biblioteca, portanto, é baseada em componentes visuais, alguns dos quais serão vistos nas próximas seções.

3 DESENVOLVIMENTO

Para o desenvolvimento da aplicação móvel foi utilizada uma abordagem baseada em componentes. Todas as telas foram construídas como componentes que são

formados, internamente, por outros componentes. A especificação visual de cada tela foi realizada de forma colaborativa, utilizando a ferramenta figma (www.figma.com). Essa ferramenta, apesar de facilitar o processo de elaboração das telas, que ocorre de maneira visual, não foi utilizada para a codificação, cuja especificação foi realizada manualmente, utilizando a biblioteca *React Native*, descrita na seção anterior. A próxima subseção descreve em detalhes as principais funcionalidades da aplicação, através de figuras das suas telas, e é seguida por uma breve explicação da parte que funciona do lado do servidor, descrita na subseção 3.2

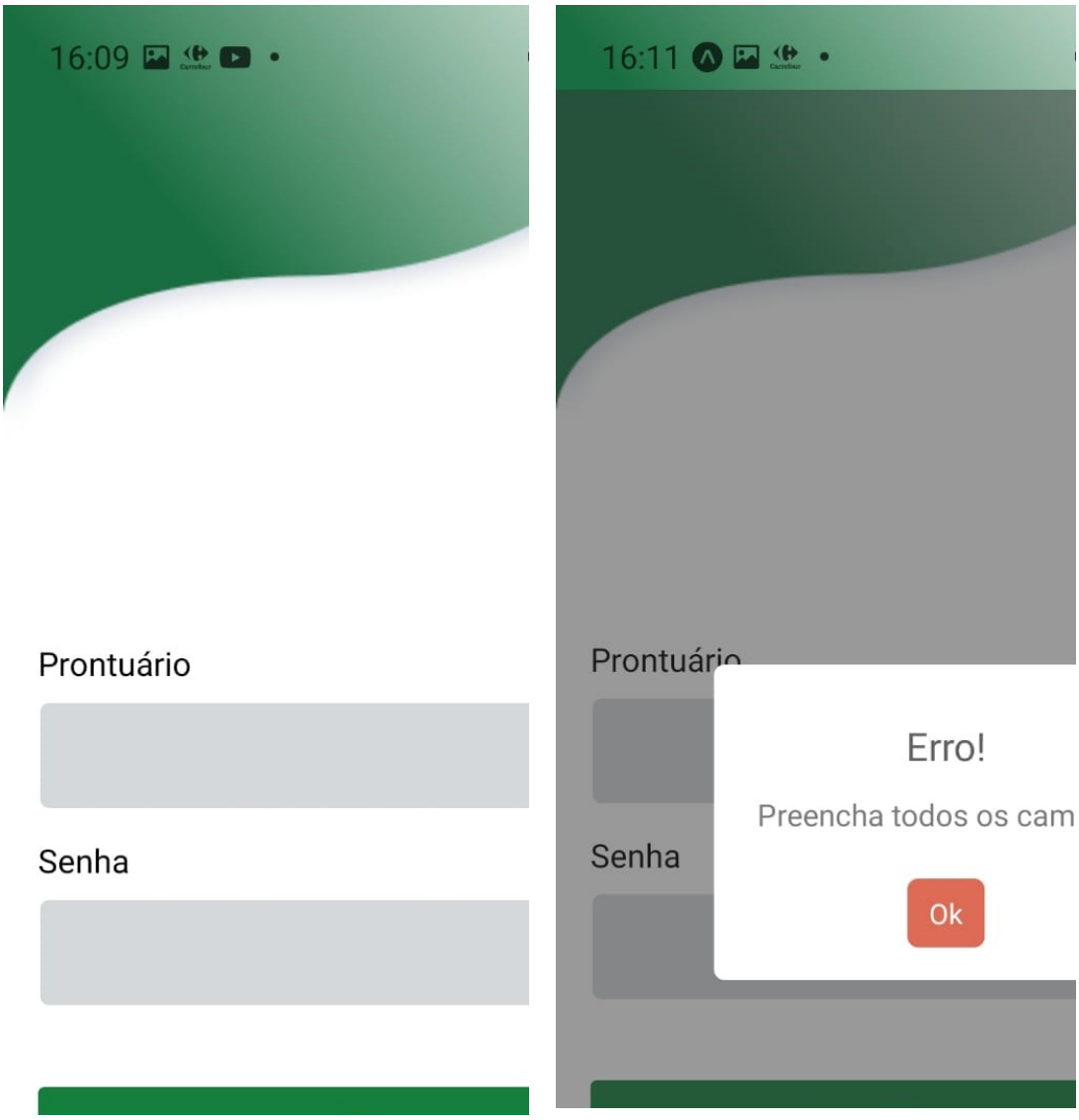
2.1 APLICAÇÃO MÓVEL

De forma similar a diversos aplicativos, a tela inicial do aplicativo desenvolvido consiste em uma área de autenticação, apresentada na Figura 3 (a). Nesta tela deve ser fornecido o código do usuário previamente cadastrado e a sua senha. Nessa operação, caso o usuário forneça dados inconsistentes, como por exemplo, deixe de preencher o seu código ou mesmo a sua senha, uma mensagem específica de erro é fornecida. Um exemplo de uma mensagem de erro, relativa ao não preenchimento dos campos é apresentada na Figura 3 (b).

Após o preenchimento dos campos, com valores válidos, o usuário seleciona a opção entrar e os dados são enviados, de forma criptografada, através do uso do protocolo HTTPS (TANENBAUM, 2021), para o servidor, onde os valores serão verificados na base de dados. Se os valores fornecidos estiverem corretos, o usuário tem acesso ao menu principal da aplicação onde as opções gerais, relativas aos serviços oferecidos, podem ser acessadas. Caso contrário, uma mensagem de erro é fornecida. Nesse caso, o usuário tem a opção de tentar gerar uma nova senha, através da opção “Esqueci minha senha”, localizada abaixo do campo senha, conforme apresentado na Figura 3.

Uma vez autenticado corretamente, os dados do usuário são salvos localmente no cache do dispositivo. Dessa forma, o usuário não precisa realizar, novamente, o processo de autenticação quando for utilizar novamente a aplicação. Caso o usuário deseje encerrar esta seção, basta utilizar a opção sair que será apresentada junto ao menu principal de opções.

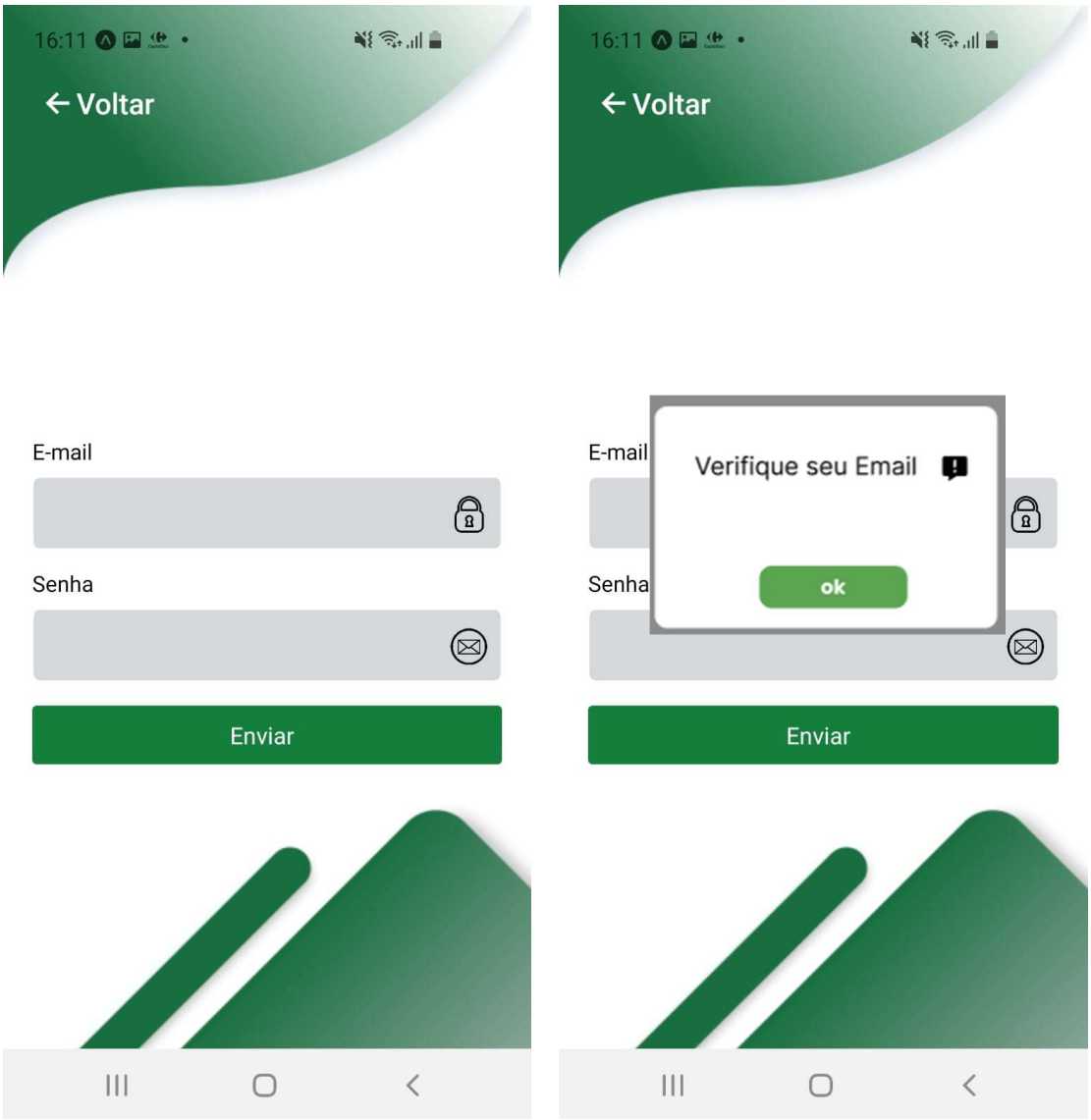
Figura 3: Telas de Autenticação da Aplicação



(a) Fonte: do autor (2023) (b)

Como mencionado, caso o usuário não consiga realizar a autenticação no sistema, é possível tentar recuperar sua senha através da opção apresentada na tela inicial. Neste caso, o usuário é direcionado a tela apresentada na Figura 4 (a). Ao fornecer seu email, cadastrado no sistema, e a senha que o usuário acredita ser válida, essas informações são enviadas para o servidor que realiza a validação do email e, se o mesmo estiver correto no sistema, envia uma mensagem para a atualização de senha na caixa de email do usuário. O usuário é informado dessa operação através da mensagem apresentada na Figura 4 (b). Ao atualizar a sua senha e entrar no sistema, o usuário receberá as informações do menu principal da aplicação, apresentado na Figura 5.

Figura 4: Tela de recuperação de senha



(a) Fonte: do autor (2023) (b)

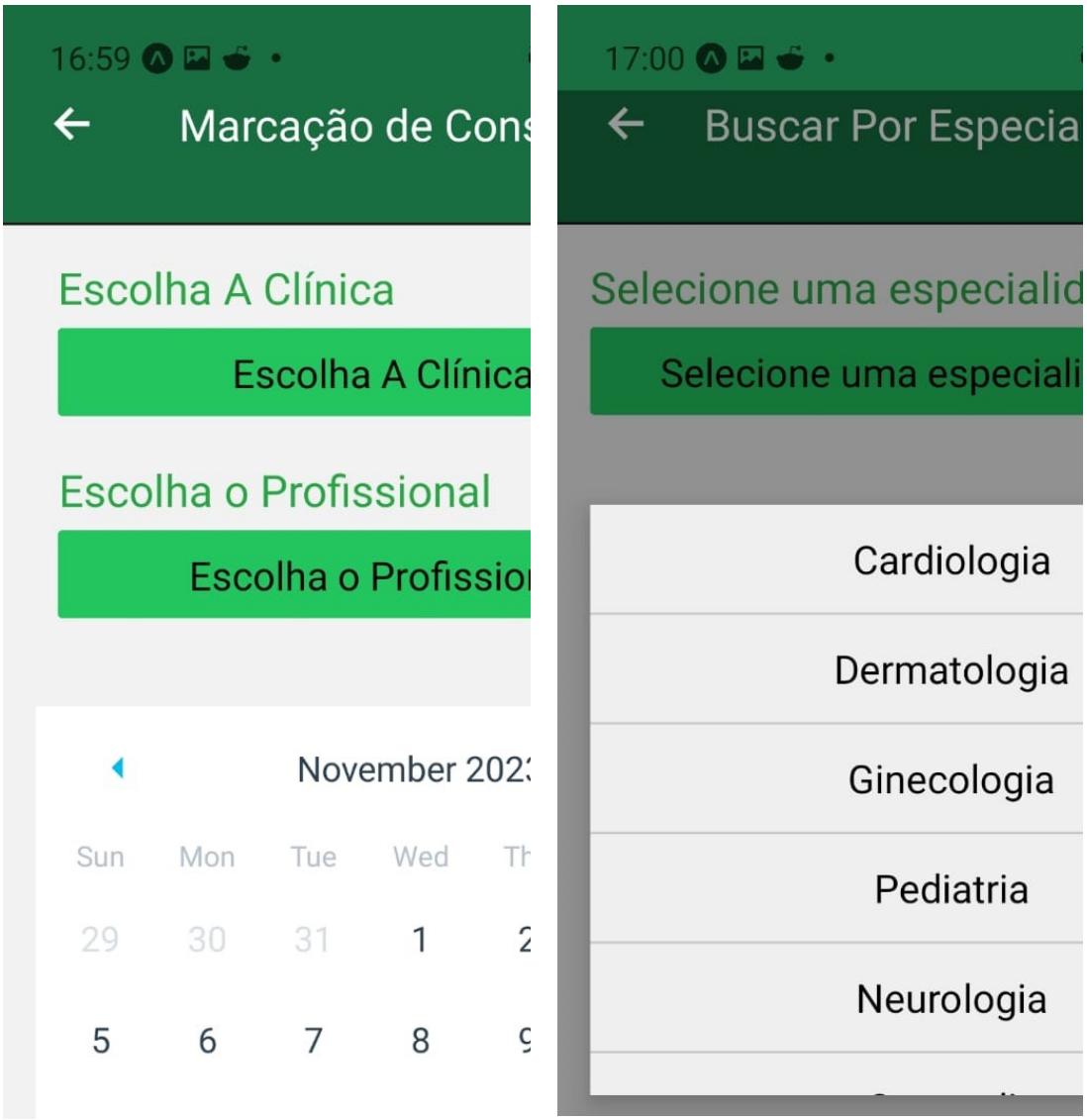
Na tela do menu principal da aplicação, apresentada na Figura 5, são oferecidas as opções de navegação para acesso aos serviços disponíveis. Nessa tela, na parte superior, aparecem as informações de boas-vindas com o convênio do usuário, caso exista esta informação. Os serviços podem ser acessados através de ícones disponíveis na tela, conforme apresentado na Figura 5 (a). Também é possível acessar os serviços através de um menu flutuante, localizado na parte superior direita. Ao clicar nesse menu, as opções dos serviços disponíveis são apresentadas, conforme listado na Figura 5 (b). Na parte inferior são apresentadas três opções, a primeira volta ao menu principal, enquanto as outras duas acessam os serviços considerados principais, o acesso as informações da rede credenciada e o acesso à marcação de consultas.

Figura 5: Tela do Menu Principal da Aplicação

(a) Fonte: do autor (2023) (b)

Ao acessar a opção de marcar uma consulta, o usuário deve escolher a clínica desejada (cardiologia, dermatologia, ginecologia etc) e, posteriormente, escolher o profissional desejado que trabalha nessa clínica. A escolha dessas informações é apresentada na Figura 6. Escolhidas a clínica e o profissional, um calendário é apresentado com as datas disponíveis para o agendamento, conforme apresentado na Figura 6 (a). A partir da escolha da data, são apresentados os horários disponíveis. Basta o usuário selecionar um horário e confirmar para o agendamento ser realizado. Algumas restrições são realizadas nessa operação. Como exemplo, um mesmo usuário não pode realizar dois agendamentos para a mesma especialidade.

Figura 6: Telas de Marcação de Consultas

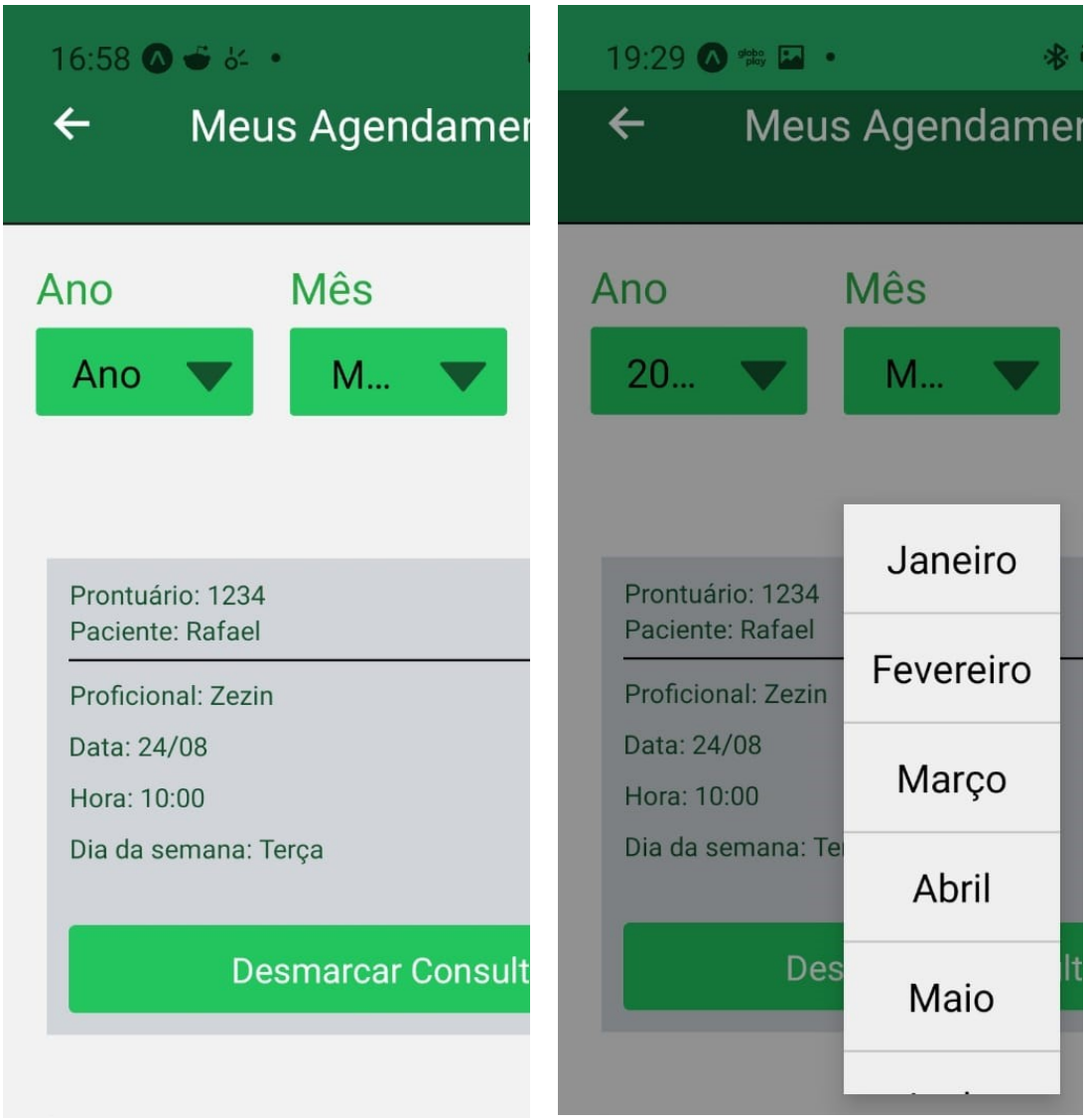


(a) Fonte: do autor (2023) (b)

É importante destacar que, mesmo que não existam vagas disponíveis em uma determinada especialidade as opções de clínicas cadastradas são apresentadas aos usuários, com o objetivo de registrar a demanda existente. Assim, toda vez que um usuário procura por vagas em uma determinada clínica essa procura é registrada no sistema com o objetivo de orientar eventuais futuros investimentos no aumento de vagas para uma determinada especialidade.

Além da opção de marcar uma consulta ou procedimento, o usuário também pode visualizar as suas consultas agendadas através do sistema. Dessa forma, ele pode se preparar adequadamente para a consulta ou, caso não possa comparecer, desmarcar com antecedência. As telas correspondentes a essa opção são apresentadas na Figura 7.

Figura 7: Telas de Verificação/Desmarcação de Consultas

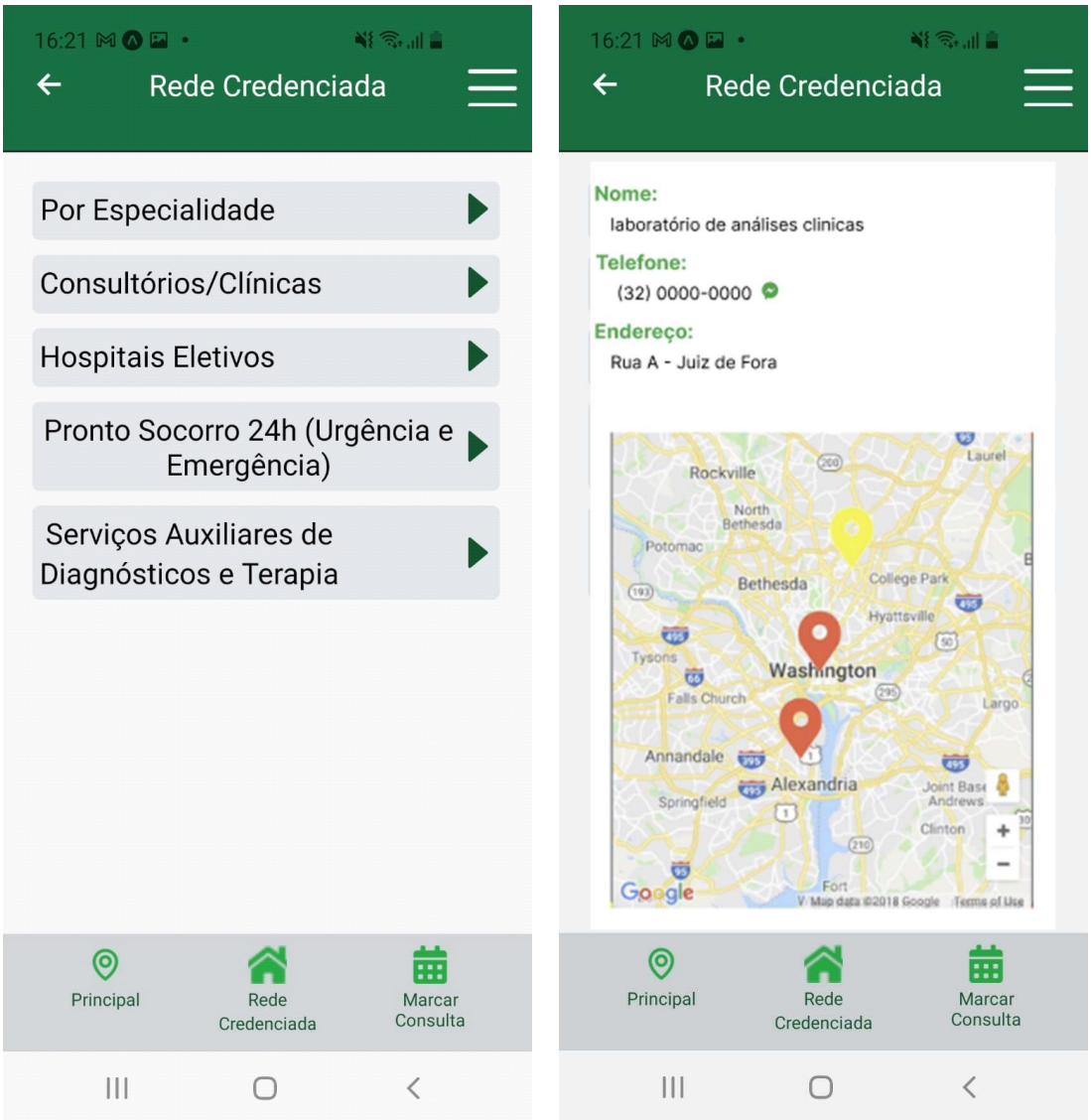


(a) Fonte: do autor (2023) (b)

A Figura 7 apresenta a tela onde o usuário tem acesso ao seu histórico de consultas agendadas. Inicialmente, são apresentadas as consultas futuras, isto é, as consultas cuja data de realização ainda não foi alcançada. Para essas consultas é oferecida a opção de desmarcação, conforme apresentado na Figura 7 (a). Através dessa opção, quando uma consulta é desmarcada, ela volta a ser oferecida, diminuindo a chance da vaga ser perdida. Caso o usuário precise verificar as consultas antigas é possível aplicar o filtro de data apresentado na Figura 7 (b). Nesse caso, as consultas antigas não possuem a opção de desmarcação, uma vez que é necessário preservar o histórico do usuário.

Uma outra funcionalidade do sistema consiste na consulta das clínicas credenciadas. Através desta opção o usuário tem acesso aos hospitais, clínicas e consultórios onde os procedimentos podem ser realizados. O acesso a essas informações é realizado através da seleção do tipo de serviço desejado, conforme apresentado na Figura 8 (a). A partir da seleção do tipo de serviço, o usuário tem acesso a um conjunto de prestadores de serviço de saúde e, ao selecionar um desses prestadores, o usuário obtém as informações de contato, conforme apresentado na Figura 8 (b). As informações de contato incluem o endereço, o telefone para contato, bem como o mapa da localização.

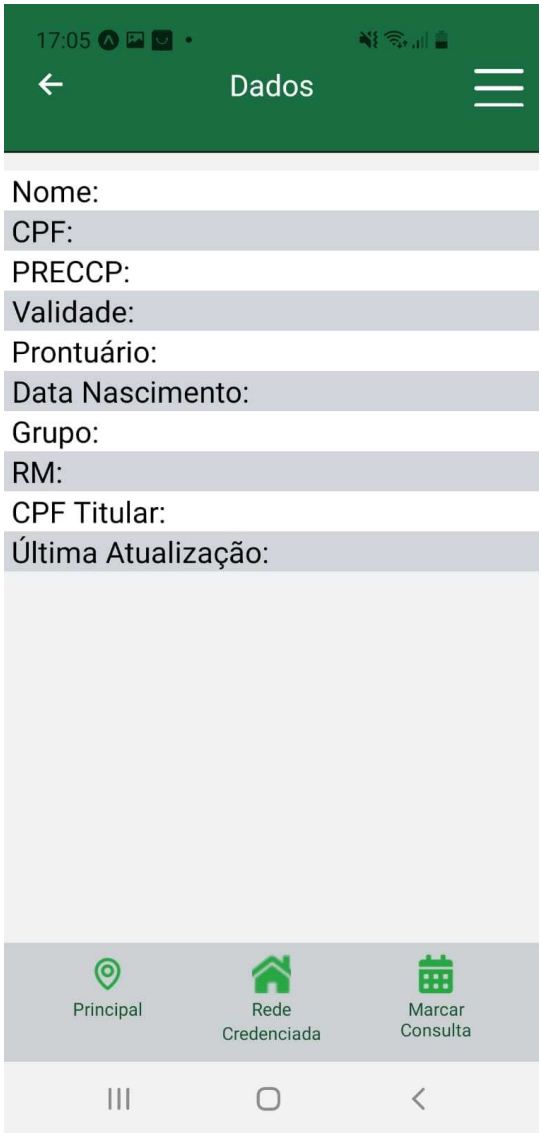
Figura 8: Telas de Consulta da Rede Credenciada



(a) Fonte: do autor (2023) (b)

, Entre os demais serviços oferecidos pela aplicação, pode ser destacado as informações a respeito dos dados do paciente. Essas informações incluem os seus dados pessoais, bem como os dados referentes ao eventual plano de saúde do qual ele faz parte. Essas informações podem ser necessárias para realização de um atendimento. Dessa forma, é importante que os usuários possam verificar e, eventualmente validar essas informações, que são apresentadas como mostrado na Figura 9. Em caso de divergências, os usuários podem solicitar atualizações através da opção referente ao atendimento.

Figura 9: Telas de Consulta dos Dados do Usuário



Fonte: do autor (2023)

3.1 SERVIDOR

Para atender aos diversos serviços relacionados ao aplicativo móvel, alguns dos quais foram listados na subseção anterior, é necessário oferecer recursos que possam ser acessados remotamente. Embora alguns desses recursos possam ser implementados sobre serviços já disponíveis na Internet, a opção utilizada neste aplicativo foi a de implementar todos os serviços remotos no servidor próprio. Essa solução permite uma integração com soluções proprietárias, como com sistemas de marcação de consultas que já existem, mas que não oferecem interfaces interativas com o usuário.

Considere, como exemplo, a utilização da autenticação integrada, utilizando, por exemplo, o endereço de email. Esse modelo de autenticação, à primeira vista, é bastante conveniente, principalmente para as aplicações lançadas recentemente e que, portanto, ainda não possuem uma base de usuários consolidada. Essa solução, no entanto, apresenta dificuldades de integração com os sistemas proprietários, além de promover uma dependência de plataformas de terceiros. Essa dependência envolve também riscos à privacidade e a segurança que, de maneira geral, foram considerados na decisão de implementar todos os serviços em servidor próprio.

Para a implementação do servidor foi utilizada uma base de dados MySQL (www.mysql.com) que consiste em uma solução de gerenciamento de base de dados relacional. Para o acesso aos dados, por sua vez, foi implementa uma API (*application program interface*) REST (*representational state transfer*). Nessa proposta, cada recurso, no caso, o acesso os dados da aplicação, seja para consulta ou manipulação, é identificado por uma URI (*Uniform Resource Identifier*). O objetivo na elaboração das URIs é que elas sejam claras e significativas pois, assim, através da sua leitura é possível identificar os recursos que elas representam. Considere, como exemplo, a Figura 10, que apresenta os serviços relacionados à autenticação do usuário.

Figura 10: Rotas de serviços de autenticação do usuário

```
authRouter.get("/users", requireAuth, getAllUsers);
authRouter.get("/users/:id", requireAuth, getUserById);
authRouter.post("/users", requireAuth, createUser);
authRouter.post("/login", login);
authRouter.get("/logout", requireAuth, logout);
```

Fonte: do autor (2023)

Na Figura 10, é possível observar que o caminho, identificado pelo recurso “/users”, quando recebido pelo servidor, irá fornecer, como resultado, uma lista de todos os usuários cadastrados no banco de dados. De maneira similar se, após o recurso “/users”, for fornecido o identificador único do usuário como, por exemplo, “/users/1”, apenas os dados referentes ou usuário desse identificador serão fornecidos.

Nesse modelo, as requisições são realizadas através do protocolo HTTPS o que garante a confidencialidade dos dados durante a transmissão, uma vez que o protocolo se encarrega da tarefa de criptografia, que é realizada na origem, no dispositivo móvel do usuário ou no servidor, e descriptografada apenas no destino.

Além de operações de retorno dos dados, como já mencionado, este modelo permite a execução de serviços no servidor. Este é o caso, por exemplo, dos recursos “/login”, que tenta fazer a autenticação do usuário a partir dos dados fornecidos e do recurso “/logout” que, ao contrário, informa ao servidor que o usuário não deseja mais, no momento, utilizar a aplicação e que, portanto, sua sessão deverá ser encerrada. Ambos os recursos citados podem ser visualizados na Figura 10.

Seguindo as diretrizes de uma API Rest, todas as operações realizadas retornam os dados sobre um formato JSON (*javascript object notation*) (FLANAGAN, 2020). Nesse formato os dados possuem um formato semiestruturado, sobre a forma de um objeto formado por campos. Esse formato é particularmente útil para a descrição de conteúdo, como é o caso do retorno das informações de um determinado usuário. Quando os recursos não produzem um resultado específico, é retornado um JSON mencionando se a operação foi realizada com sucesso ou não. Nesse último caso, os problemas encontrados são informados, complementarmente na resposta.

Uma API Rest, por definição, é independente de estado, como normalmente é esperado do protocolo HTTPS. Isso significa que cada requisição é independente das demais. Portanto, todas as requisições devem conter todas as informações necessárias para serem corretamente executadas. Essa abordagem é importante a fim de manter a implementação simples e escalável, permitindo que o número de requisições cresça sem comprometer o desempenho do sistema. No entanto, no contexto da aplicação objeto deste trabalho, torna-se necessário implementar medidas de segurança a fim de evitar que as requisições possam ser realizadas sem a devida autenticação.

Considere, por exemplo, que o processo de marcação ou cancelamento de uma consulta, que também é realizado por recursos acessados no servidor, possa ser realizado através de uma única chamada, que é justamente o caso. Nesse contexto, se não forem tomadas medidas de segurança, poderia ser possível marcar e desmarcar consultas sem que o usuário estivesse autenticado.

A fim de garantir que as operações sobre a base de dados somente sejam realizados por usuários autenticados, foi implementado no servidor a geração de tokens de forma dinâmica para os usuários autenticados. Dessa forma, as requisições, independente do seu objetivo, devem conter o token para autenticação.

4 RESULTADOS E DISCUSSÃO

Os alunos dos cursos de Engenharia de Software e de Sistemas de Informação estudam todas as tecnologias empregadas neste trabalho ao longo de diferentes períodos dos cursos. Dessa forma, realizar um trabalho como este é uma oportunidade de integrar diferentes conhecimentos adquiridos ao longo do curso implementando, em conjunto, aquilo que foi aprendido de maneira segmentada.

A aplicação desenvolvida ao longo deste trabalho pode ser colocada em funcionamento em diferentes clínicas e hospitais, facilitando o acesso a saúde que, como já mencionado, é uma questão fundamental da sociedade. Essa ação de implementação para a colocação em funcionamento, por si só, já é um trabalho que merece ser explorado em maiores detalhes, mas que somente pode ser realizado a partir do ponto em que a aplicação se encontra.

Evidentemente, a aplicação ainda apresenta algumas limitações importantes, pois são necessário trabalhos de implementação para a sua finalização. Isto não compromete, no entanto, a sua utilização em um ambiente real. Na verdade, a aplicação pode ser beneficiada a partir da utilização em um ambiente real com o objetivo de aprimorar o seu desenvolvimento.

Devido a sua importância social, todo o trabalho desenvolvido está disponível para acesso pela sociedade podendo, livremente, ser copiado e modificado desde que o objetivo seja a utilização sem fins comerciais em ambientes que favoreçam o desenvolvimento da saúde como bem estar social.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvida uma aplicação para marcação de consultas que pode ser utilizada em hospitais e clínicas facilitando o acesso a serviços de saúde aos usuários que, muitas vezes, podem ter dificuldade de acesso a esses serviços.

Como trabalho futuro, além da continuidade no desenvolvimento da aplicação, inserindo novas funcionalidades, pretende-se implantar o serviço desenvolvido em uma unidade de saúde real. Essa implementação, além de permitir uma experiência relacionada a real usabilidade do sistema, poderá identificar novas necessidades para desenvolvimento de futuros serviços.

ABSCTRACT

This work presentes the development of a tool for scheduling appointments, in the form of a mobile application, developed using the React Native library. This tool has several functionalities, in addition to the possibility of scheduling and canceling appointments, which include validating user registration, as well as recommendaing clinics and professionals to carry out heath procedures. Each of these functionalities is linked to a remote server, where a database is available and accessed through na interface built specifically for this project. Access records are stored for later demand analysis and application management.

Keywords: Scheduling. Mobile Applications. Health. React Native.

REFERÊNCIAS

BANKS, A.; PORCELLO, E. **Learning React: Functional Web Development With React and Redux**. ISBN 978-1-491-95462-1. O'Reilly Media. 2017.

EISENMAN, B. **Learning React Native**. ISBN 978-1-491-98914-2. O'Reilly Media. 2018.

FLANAGAN, D. **JavaScript: The Definitive Guide**. ISBN 978-05-96805-52-4. O'Reilly Media. 2020.

iOS. **Build Your Apps for iOS 12.** Disponível em <https://developer.apple.com/ios>. Acesso em: 20 de Novembro de 2023.

LECHETA, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. ISBN 978-8575224687. Novatec. 2015.

PONTES, G. **Progressive Web Apps: Construa Aplicações Progressivas com React**. ISBN 978-85-94188-56-4. Casa do Código. 2018.

RICCIARDI, W.; BARROS, P.P. et all. **How to govern the digital transformation of heath services**. European Journal of Public Health, Vol 29. Outubro 2019. https://doi.org/10.1093/eurpub/ckz165

TANENBAUM, A. **Redes de Computadores**. ISBN 978-85-8260-56-08. Bookman, 6ª Edição. 2021.

WARNER, K. **The economic implications of preventive health care**. Social Science & Medicine. Part C: Medical Economics, Vol 23. Dezembro 1979. Disponível em <https://sciencedirect.com/science/article/abs/pii/0160799579900042>. Acesso em: 20 de Novembro de 2023.

YANG, X. **Health expenditure, human capital, and economic growth: na empirical study of developing countries**. International Journal of Health Economics and Management, Vol 20. Outubro 2019. Disponível em <https://link.springer.com/article/10.1007/s10754-019-09275-w>. Acesso em: 20 de Novembro de 2023.