UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PAULO VICTOR CORAZZA

Um aplicativo multiplataforma desenvolvido com Flutter e NoSQL para o cálculo da probabilidade de apendicite

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives
Co-orientador: Prof. Dr. Ricardo Francalacci Savaris

Porto Alegre
2018

AGRADECIMENTOS

Aos meus familiares, especialmente a minha mãe Carmen Gatto, ao meu pai Paulo Corazza e minha tia Cláudia Gatto, pelo apoio e suporte durante a elaboração deste trabalho.

A minha namorada, Débora Ritter, futura médica, pelo incentivo e compreensão, me inspirando para a realização deste trabalho e contribuindo de modo significativo ao divulgar o questionário de avaliação para seus professores e colegas.

Aos profissionais e estudantes de medicina que responderam ao questionário de avaliação.

Aos funcionários e professores do Instituto de Informática da UFRGS, em especial ao professor Raul Weber (*in memorian*) pela disponibilidade em me auxiliar sempre que necessitei.

Ao professor Ricardo Savaris, meu coorientador, pelas informações que subsidiaram a elaboração deste trabalho e pela presteza em esclarecer as dúvidas que surgiram durante o desenvolvimento.

Um agradecimento muito especial ao meu orientador, professor Leandro Wives, pela inestimável contribuição em todas as fases do trabalho. Registro o seu empenho e dedicação na orientação.

RESUMO

O presente trabalho consiste na elaboração de um aplicativo móvel para Android e iOS para calcular a probabilidade de apendicite, com o objetivo de auxiliar médicos no diagnóstico da doença. Ao longo do trabalho são exploradas as ferramentas utilizadas no desenvolvimento, como o framework Flutter para o desenvolvimento multiplataforma e o Cloud Firestore, um banco de dados NoSQL na nuvem para o armazenamento de dados. Também é apresentado como o aplicativo foi projetado e o seu processo de desenvolvimento. O resultado é demonstrado por meio de telas que exibem as funcionalidades desenvolvidas no aplicativo. Ao final são analisados os resultados de uma avaliação de usabilidade realizada com usuários reais. Constatou-se que a interface do aplicativo apresentou facilidade no uso, com resultados positivos entre os participantes.

Palavras-chave: Aplicativo móvel. Android. iOS. Flutter. NoSQL. Apendicite.

A cross-platform application developed with Flutter and NoSQL for calculating the probability of appendicitis

ABSTRACT

The goal of this work is to develop a mobile application for Android and iOS to calculate the probability of appendicitis to assist physicians in the diagnosis of the disease. Throughout the work, the tools used in development are presented and explored, such as Flutter, a framework for cross-platform development, and Firestore, a NoSQL database in the cloud for data storage. It also shows how the application was designed and its development process. The result is demonstrated through screens that display the features developed in the application. At the end, the results of a usability evaluation with users are analyzed. It was verified that the interface of the application was easy to use, with positive results among the participants.

Keywords: Mobile Application. Android. iOS. Flutter. NoSQL. Appendicitis.

LISTA DE FIGURAS

Figura 2.1 Smartphone com Android 9.0 Pie....14
Figura 2.2 Arquitetura do Android 16
Figura 2.3 iPhone X com iOS 12 18
Figura 2.4 Arquitetura do iOS....19
Figura 2.5 Hierarquia de widgets 23
Figura 2.6 Arquitetura do Flutter 24

Figura 3.1 Aplicativo desenvolvido pelo coorientador 29
Figura 3.2 Tela do Trello após a entrega de todos os *sprints* 33
Figura 3.3 Tela do Android Studio com o projeto do aplicativo com o Flutter 34
Figura 3.4 Exemplo de dado armazenado no Firebase Cloud Firestore 35
Figura 3.5 Classe Probability 36
Figura 3.6 Tela de login 37
Figura 3.7 Tela principal para inserção dos dados do paciente....38
Figura 3.8 Tela principal para usuários registrados....39
Figura 3.9 Informação dos campos inseridos de forma inválida....40
Figura 3.10 Tela para a exibição do resultado....41
Figura 3.11 Tela exibindo um menu de navegação lateral 42
Figura 3.12 Tela com a navegação lateral para visitantes 43
Figura 3.13 Tela com a lista de resultados registrados....44
Figura 3.14 Tela para edição/remoção dos dados 45
Figura 3.15 Tela com informações sobre o aplicativo....46
Figura 3.16 Tela principal para inserção dos dados do paciente em um dispositivo iOS47

Figura 4.1 Profissão dos participantes....49
Figura 4.2 Domínio na língua inglesa dos participantes 50
Figura 4.3 Domínio na utilização de aplicativos móveis por parte dos participantes 50
Figura 4.4 Tarefa 1 51
Figura 4.5 Tarefa 2 51
Figura 4.6 Tarefa 3 52
Figura 4.7 Tarefa 4 52
Figura 4.8 Tarefa 5 52
Figura 4.9 Tarefa 6 53
Figura 4.10 Classificações em relação à pontuação SUS....54
Figura 4.11 Questionário SUS - Resultados 55

LISTA DE TABELAS

Tabela 1.1 Vendas globais de smartphones no primeiro trimestre de 2018 10

Tabela 2.1 Comparação entre as principais plataformas móveis 20

Tabela 3.1 User Story 1 30
Tabela 3.2 User Story 2 30
Tabela 3.3 User Story 3 30
Tabela 3.4 User Story 4 30
Tabela 3.5 User Story 5 30
Tabela 3.6 User Story 6 30
Tabela 3.7 User Story 7 31
Tabela 3.8 User Story 8 31
Tabela 3.9 User Story 9 31
Tabela 3.10 User Story 10 31
Tabela 3.11 User Story 11 31
Tabela 3.12 User Story 12 31
Tabela 3.13 Descrição dos dados armazenados 35

LISTA DE ABREVIATURAS E SIGLAS

AOT	Ahead-of-time
API	Application Programming
ART	Android Runtime
DSL	Domain-Specific Language
HAL	Hardware Abstraction Layer
IDE	Integrated Development Environment
JIT	Just-in-time
MDD	Model Driven Design
NoSQL	Not Only Structured Query Language
OS	Operating System
PID	Pelvic Inflammatory Disease
SDK	Software Development Kit
SQL	Structured Query Language
SUS	System Usability Scale
UI	User Interface
URL	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Motivação 11
1.2 Objetivo 12
1.3 Estrutura 12
2 FUNDAMENTAÇÃO TEÓRICA E TECNOLOGIAS UTILIZADAS 13
2.1 Android 13
2.2 iOS 17
2.3 Desenvolvimento multiplataforma 19
2.3.1 Flutter 22
2.4 NoSQL 24
2.4.1 Firebase Cloud Firestore 25
2.5 FirebaseUI 26
3 APLICATIVO PROPOSTO: PROJETO E DESENVOLVIMENTO 28
3.1 Aplicativo existente 28
3.2 User Stories 29
3.3 Processo de desenvolvimento 31
3.4 Ambiente de desenvolvimento 33
3.5 Armazenamento de dados 34
3.6 Funcionamento do aplicativo 36
3.6.1 Tela de login 37
3.6.2 Tela principal 38
3.6.3 Tela com o resultado 41
3.6.4 Telas com a navegação lateral 42
3.6.5 Tela de resultados 44
3.6.6 Tela de edição 45
3.6.7 Tela de informações 46
3.6.8 Tela principal no iOS 47
3.7 Requisitos de sistema 48
4 AVALIAÇÃO DO APLICATIVO 49
4.1 Perfil dos participantes 49
4.2 Facilidade na realização das tarefas 51
4.3 System Usability Scale 53
5 CONCLUSÃO 56
REFERÊNCIAS 57
APÊNDICE A — QUESTIONÁRIO DE AVALIAÇÃO 59
A.1 Instruções para realização do questionário 59
A.2 Perfil do participante 60
A.3 Facilidade no uso do aplicativo 61
A.4 Questionário System Usability Scale 63

1 INTRODUÇÃO

Os aplicativos móveis estão tornando-se cada vez mais populares e presentes na vida das pessoas. O número de downloads desses aplicativos tem apresentado forte crescimento e é notável o aumento do interesse das pessoas por mobilidade. Isso está ligado principalmente à constante evolução dos dispositivos móveis, que tem apresentado capacidade de processamento e armazenamento cada vez maiores.

Android e iOS predominam no mercado de sistemas operacionais para smartphones, os dispositivos móveis mais utilizados atualmente. A Tabela 1.1 demonstra que as vendas globais de smartphones equipados com Android no primeiro trimestre de 2018 representaram a imensa maioria, chegando a 85,9% do total, segundo dados da Gartner (2018). Os outros 14,1% são de smartphones equipados com iOS. Todos os outros sistemas operacionais somados registraram vendas pouco significativas.

Tabela 1.1: Vendas globais de smartphones no primeiro trimestre de 2018

Sistema operacional	**Unidades (milhões)**	**Participação no mercado (%)**
Android	329,313	85,9
iOS	54,058	14,1
Outros	0,131	0.0
Total	**383,503**	**100**

Fonte: Adaptado de (GARTNER, 2018)

Com a ascensão na venda de smartphones e a consolidação do Android e iOS como sistemas operacionais predominantes, há uma necessidade cada vez maior no desenvolvimento de aplicativos para atender a demanda das pessoas nas mais diferentes áreas.

O aplicativo proposto neste trabalho está inserido na área da saúde, onde o uso de dispositivos móveis por profissionais da área têm transformado muitos aspectos da prática clínica.

Dispositivos móveis, como smartphones e *tablets*, têm substituído os computadores *desktop* para profissionais da saúde que necessitam acesso rápido à informação no seu local de atendimento, pois combinam recursos de computação e comunicação em um único aparelho que pode ser segurado na mão ou armazenado no bolso. Isso fez com que se tornassem comuns em ambientes de saúde, levando a um rápido crescimento no desenvolvimento de aplicativos para essas plataformas (VENTOLA, 2014).

Os aplicativos contribuem com os profissionais da área em diversas tarefas, entre elas a gestão e monitoramento de pacientes e a tomada de decisão clínica, mais rápida e com uma taxa menor de erro, impactando positivamente nos resultados do tratamento dos pacientes (VENTOLA, 2014).

1.1 Motivação

O coorientador deste trabalho realizou um estudo não publicado de uma coorte prospectiva que identificou e analisou múltiplas variáveis em relação a 126 pacientes com dor abdominal na Unidade de Emergência do Hospital de Clínicas de Porto Alegre e criou um algoritmo para calcular a probabilidade de um paciente ter apendicite.

O cálculo é realizado a partir da incidência da doença e de dados clínicos: idade, sexo, número de dias com sintomas, valor dos leucócitos, presença de dor abdominal no quadrante inferior direito e sinal de Blumberg.

Para calcular, validar e armazenar os resultados dos cálculos de seus pacientes, inicialmente desenvolveu uma planilha no software Microsoft Excel. Uma vez que os pontos da curva *Receiver Operator Characteristic* utilizava vários pontos do leucograma, era inviável a utilização desse cálculo na prática diária. Sentindo a necessidade de uma maior comodidade, portabilidade e facilidade para realizar os cálculos, desenvolveu um aplicativo móvel para esse fim.

O aplicativo existente apresentava algumas limitações, entre elas, a sua disponibilidade somente para a plataforma iOS — limitando sua utilização —, uma interface muito simples — com campos de inserção de texto pouco informativos, dificultando a compreensão sobre o que deveria ser inserido —, e não foi desenvolvido utilizando as melhores práticas, metodologias e técnicas computacionais.

Para solucionar essas questões, surgiu a necessidade da criação de um novo aplicativo, que fosse disponibilizado também para a plataforma Android, de modo a atingir praticamente a totalidade dos usuários de dispositivos móveis que tenham interesse no tema. Esse aplicativo deveria contar com novas funcionalidades, como um banco de dados para o registro dos cálculos e ser desenvolvido utilizando os mais recentes princípios para a concepção de interfaces de usuários. Também deveria ter seu código centralizado em uma única base, para que novos recursos pudessem ser acrescentados com facilidade.

1.2 Objetivo

O objetivo deste trabalho consiste no desenvolvimento de um aplicativo multiplataforma para dispositivos móveis (Android, iOS), utilizando o *framework* Flutter e um banco de dados NoSQL, oferecendo uma interface intuitiva, para que médicos possam calcular a probabilidade da presença de apendicite em seus pacientes e registrar os resultados, auxiliando no diagnóstico precoce a fim de evitar possíveis complicações de um diagnóstico tardio.

1.3 Estrutura

O trabalho está organizado em cinco capítulos. Após a introdução, os próximos capítulos seguem a seguinte estrutura:

Capítulo 2: aborda a fundamentação teórica e detalha os conceitos relacionados às escolhas tecnológicas utilizadas durante o desenvolvimento do aplicativo.

Capítulo 3: demonstra como o aplicativo foi projetado e como foi organizado o processo de desenvolvimento.

Capítulo 4: descreve a avaliação de usabilidade do aplicativo realizada por usuários.

Capítulo 5: apresenta as conclusões e questões que podem ser exploradas futuramente.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLOGIAS UTILIZADAS

Neste capítulo serão apresentados os conceitos relacionados às escolhas tecnológicas utilizadas para o desenvolvimento deste trabalho. Inicialmente, serão detalhados os sistemas operacionais aos quais o aplicativo se destina, o Android e iOS.

Na sequência serão apontadas considerações sobre o desenvolvimento multiplataforma e o Flutter, o *framework* escolhido para o desenvolvimento do aplicativo.

Ao final do capítulo, serão introduzidos os bancos de dados NoSQL e será apresentando o Firebase Cloud Firestore, escolhido para realizar o armazenamento dos dados inseridos no aplicativo.

Todas as tecnologias apresentam uma documentação detalhada e podem ser facilmente integradas, possibilitando o desenvolvimento de um aplicativo consistente.

2.1 Android

O Android é um sistema operacional projetado principalmente para executar em dispositivos móveis como smartphones e tablets. Desde sua introdução, o Android vem crescendo em termos de popularidade, se tornando atualmente o líder no mercado de dispositivos móveis, estando presente em 85,9% dos smartphones vendidos em todo o mundo no primeiro trimestre de 2018 (GARTNER, 2018).

É uma plataforma de código-aberto, o que o torna customizável para um grande número de dispositivos. Hoje o sistema pode ser encontrado em diversos outros dispositivos como TVs, reprodutores de mídia, relógios inteligentes, painéis de automóveis, dispositivos médicos, utensílios domésticos, entre outros (TANENBAUM, 2015).

Inicialmente o Android foi desenvolvido pela empresa Android Inc., sendo adquirido pelo Google em julho de 2005. Em setembro de 2008, o primeiro dispositivo comercial equipado com o sistema foi lançado, equipando a versão 1.0 (LECHETA, 2015). Com o passar dos anos, novas versões foram sendo lançadas e atualmente o Android está na versão 9.0, denominada Android 9 Pie[1] (Figura 2.1).

De acordo com Tanenbaum (2015), o Android é uma plataforma de código aberto e fortemente baseada no núcleo do Linux, utilizando a maioria dos mecanismos presentes nesse núcleo (processos, memória virtual, sistemas de arquivos...), adicionando algumas extensões. Uma quantidade considerável do sistema é escrita em uma linguagem de alto

[1] https://www.android.com/versions/pie-9-0/

Figura 2.1: Smartphone com Android 9.0 Pie

Fonte: (ANDROID, 2018a)

nível, o Java, e tende a seguir um projeto orientado a objetos conforme encorajado por essa linguagem. Já o núcleo e um grande número de bibliotecas de baixo nível são escritos em C e C++.

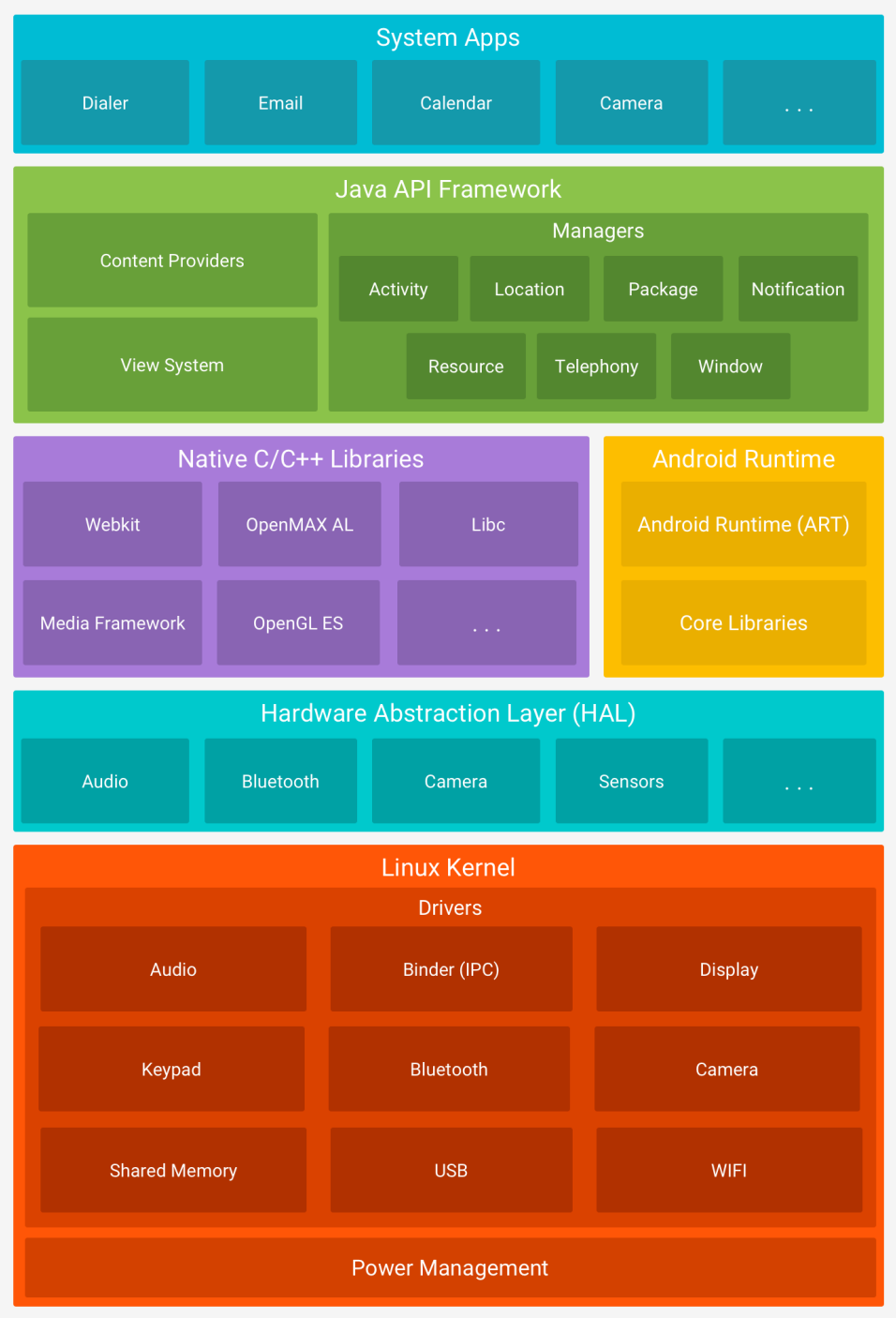
A arquitetura do Android é composta por várias camadas, cada uma com características e propósitos próprios. No entanto, nem sempre as camadas são totalmente separadas e muitas vezes se infiltram umas nas outras (GARGENTA; NAKAMURA, 2014). Na Figura 2.2, estão descritos a maioria dos componentes da plataforma Android, inseridos nas suas respectivas camadas, as quais serão detalhadas a seguir:

- **Kernel do Linux**: o Linux foi escolhido para ser a base da pilha especialmente pela sua portabilidade — é uma plataforma flexível e relativamente fácil de ser portada para várias arquiteturas de hardware —, segurança — é altamente seguro, sendo testado em diversos ambientes ao longo das décadas — e pelos seus recursos muito úteis, como por exemplo, o suporte para gerenciamento de memória e energia.
- **Camada de abstração de hardware (HAL)**: o Android foi projetado para executar nas mais variadas configurações de hardware, porém, não existe uma padronização no acesso aos *drivers* de cada dispositivo. Para evitar que um desenvolvedor de aplicativos deva se preocupar com os detalhes de cada placa, existe uma camada

responsável por fazer essa abstração. Essa camada fornece interfaces padrão que expõem os recursos de hardware do dispositivo através de vários módulos de biblioteca, cada um dos quais implementa uma interface para um tipo específico de componente de hardware, como por exemplo, a câmera.

- **Android Runtime (ART)**: é um ambiente de tempo de execução usado por aplicativos e alguns serviços do sistema no Android. É utilizado a partir da versão 5.0 do Android para substituir a máquina virtual Dalvik. É projetado para trabalhar com o Dalvik Executable (DEX), um formato de *bytecode* projetado especialmente para Android, otimizado para oferecer consumo mínimo de memória. Alguns dos principais recursos do ART são a compilação *ahead-of-time* (AOT) e *just-in-time* (JIT), a coleta de lixo otimizada e um melhor suporte a depuração, geração de relatórios de erros e diagnósticos. Além disso, a partir do Android 9, o compilador AOT otimiza os arquivos compactados DEX do pacote de um aplicativo, convertendo-os para um código de máquina mais compacto, fazendo com que os aplicativos iniciem mais rápido e consumam menos memória e espaço em disco.
- **Bibliotecas C/C++ nativas**: muitos componentes e serviços do sistema Android, como o ART e o HAL, são criados a partir de código nativo, isso requer bibliotecas nativas escritas em C e C++. Nessa camada temos APIs como a OpenGL ES, para renderização 3D, a Media Framework para processamento de áudio e vídeo e a Libc, uma implementação da biblioteca C padrão. Esse conjunto de bibliotecas, pode ser acessado a partir das APIs da camada superior.
- **Java API Framework**: todo o conjunto de recursos do sistema operacional Android está disponível por meio de APIs escritas na linguagem Java, e são a partir delas que a grande maioria dos desenvolvedores criam seus aplicativos. Assim, os desenvolvedores têm acesso total às mesmas APIs que os aplicativos de sistema do Android acessam. Nessa camada se encontram bibliotecas para criação de interfaces, comunicação com outros aplicativos, gerenciamento de recursos, gerenciamento de notificações, acesso aos dados de outros aplicativos, entre outras.
- **Aplicativos do sistema**: o Android dispõe de um conjunto de aplicativos padrão para e-mail, SMS, calendário, contatos e outros. Esses aplicativos não tem prioridades especiais em relação aos aplicativos de terceiros que o usuário escolhe instalar. Assim, um navegador desenvolvido por terceiros, por exemplo, pode inclusive substituir o aplicativo padrão do sistema para navegação na Internet.

Figura 2.2: Arquitetura do Android



Fonte: (ANDROID, 2018b)

Um aplicativo Android é por convenção um arquivo com uma extensão .apk, referente a Android Package, contendo tudo o que forma o aplicativo. Entre os conteúdos importantes desse apk está o código do aplicativo, as informações de assinatura para identificar com segurança o autor, os recursos necessários ao aplicativo como dados XML para o layout, além de um manifesto contendo um nome de pacote no estilo Java, para identifi-

car unicamente a aplicação, a descrição da aplicação e como executá-la (TANENBAUM, 2015).

O Google Play é a loja do Google para que os desenvolvedores publiquem seus aplicativos. A loja também é responsável por assegurar a compatibilidade desses aplicativos com os dispositivos que forem entregues.

2.2 iOS

O iOS é um sistema operacional projetado e desenvolvido pela Apple para ser executado exclusivamente nos dispositivos móveis que a empresa produz, o iPhone, iPad e iPod Touch. Com o iPhone, o sistema esteve presente em 14,1% das vendas mundiais de smartphones no primeiro trimestre de 2018. (GARTNER, 2018).

Originalmente chamado de iPhone OS, sua primeira versão foi lançada no ano de 2007 exclusivamente para o iPhone, único dispositivo suportado inicialmente. No mesmo ano o sistema passou a suportar também o iPod Touch. Em 2010, a empresa introduziu o iPad que também passou a equipar o sistema. No mesmo ano, a Apple renomeou o sistema operacional para iOS para refletir a natureza unificada desse sistema, presente em todos os seus dispositivos móveis (LEVIN, 2012). A Figura 2.3 demonstra a interface do sistema, atualmente na versão 12.

O iOS é baseado em UNIX, porém, diferentemente do Android, a maioria do sistema é de código fechado, ou seja, usuários não tem acesso livre a seu código fonte.

Conforme consta em Apple (2014), a arquitetura do iOS é formada por quatro camadas. O iOS age como um intermediário entre o hardware do nível mais baixo e os aplicativos. Dessa forma, esses aplicativos não se comunicam diretamente com o hardware. Para isso, utilizam um conjunto de interfaces de sistema bem definidas que permitem a execução desses aplicativos de forma consistente em dispositivos com capacidades de hardware diferentes.

A maioria das interfaces de sistema são oferecidas através de pacotes especiais denominados *frameworks*. Um *framework* é um diretório que contém uma biblioteca dinâmica e recursos (como arquivos de cabeçalho, imagens e aplicativos auxiliares) necessários para suportar essa biblioteca. Desenvolvedores utilizam essas estruturas adicionando-as aos seus projetos de aplicativos no Xcode, a IDE fornecida pela Apple para o desenvolvimento de aplicativos para o sistema iOS.

Aos desenvolvedores se recomenda o uso de *frameworks* de nível superior, sem-

Figura 2.3: iPhone X com iOS 12

Fonte: (APPLE, 2018)

pre que possível, pois eles fornecem abstrações orientadas a objeto para construções de nível inferior, encapsulando recursos complexos e reduzindo a quantidade de código a ser escrito.

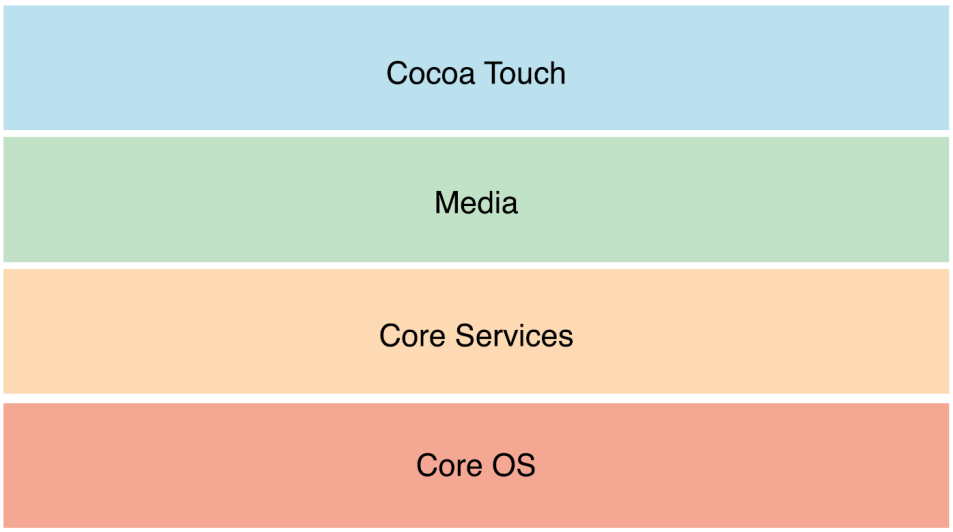
A Figura 2.4 apresenta o conjunto de camadas que compõem a arquitetura do iOS. As camadas da parte inferior contêm serviços e tecnologias fundamentais. Camadas de nível mais alto utilizam as camadas inferiores para fornecer serviços mais sofisticados. A seguir serão apresentados mais detalhes sobre essas camadas:

- **Core OS**: contém os recursos de baixo nível sob a qual a maioria das outras tecnologias são construídas. Os recursos oferecidos por essa camada são constantemente usadas por outros *frameworks*, sendo a ligação entre as outras camadas e o hardware. O desenvolvedor geralmente utiliza os *frameworks* dessa camada em situações que precisa lidar explicitamente com a segurança, ou se comunicar com um acessório de hardware externo.
- **Core Services**: contém serviços fundamentais do sistema para aplicativos. Os *frameworks* Foundation e Core Foundation e definem os tipos básicos que todos os aplicativos utilizam. Nessa camada também estão tecnologias individuais para su-

portar recursos como localização, iCloud, mídia social e rede.

- **Media**: as tecnologias disponíveis nessa camada (gráficos, áudio e vídeo) são utilizadas para implementar recursos multimídia nos aplicativos.
- **Cocoa Touch**: nessa camada estão diversos *frameworks* importantes para a criação de aplicativos para iOS. Esses *frameworks* definem a aparência, a infra-estrutura básica de aplicativos e o suporte a tecnologias como multitarefa, entrada baseada em toque, notificações *push* e outros serviços de sistema.

Figura 2.4: Arquitetura do iOS



Fonte: (APPLE, 2014)

A App Store é a loja desenvolvida e mantida pela Apple para que desenvolvedores possam disponibilizar seus aplicativos e para que usuários possam obter novos aplicativos para o iOS.

2.3 Desenvolvimento multiplataforma

Um dos principais objetivos dos desenvolvedores de aplicativos móveis, geralmente, é tornar seu aplicativo disponível para o maior número de usuários possíveis. Para atingir esse objetivo, existem, entre outras, duas abordagens frequentemente utilizadas, o desenvolvimento nativo e o multiplataforma.

No processo de desenvolvimento de aplicativos nativos o desenvolvedor dispõe de APIs completas fornecidas pela plataforma para acessar todos os recursos do dispositivo móvel (câmera, sensores, rede, GPS...). Geralmente o aplicativo desenvolvido apre-

senta desempenho superior aos desenvolvidos com outras abordagens e proporcionam uma experiência de interface nativa para a plataforma de destino pois são desenvolvidos utilizando as ferramentas para construção de interfaces específicas para cada plataforma (EL-KASSAS et al., 2017).

O ciclo de vida de desenvolvimento do aplicativo móvel nativo consiste na análise da ideia do aplicativo, projeto da interface do usuário, desenvolvimento utilizando as ferramentas e linguagens de programação da plataforma de destino, teste do aplicativo em dispositivos diferentes e, finalmente, publicação do aplicativo na loja da plataforma de destino. Esse ciclo é repetido para cada plataforma, com a exceção do primeiro passo (EL-KASSAS et al., 2017).

Devido as particularidades de cada plataforma, esse processo exige que o desenvolvedor compreenda e utilize os recursos específicos fornecidos por cada plataforma como IDEs, linguagens de programação e APIs. Além disso, restrições associadas a cada plataforma também podem ser uma dificuldade no desenvolvimento (ex.: a IDE Xcode só está disponível para dispositivos da Apple, obrigando o desenvolvedor a ter acesso a esses dispositivos para produzir o aplicativo). A Tabela 2.1 demonstra algumas diferenças que devem ser consideradas no desenvolvimento de um aplicativo para as duas principais plataformas móveis.

Tabela 2.1: Comparação entre as principais plataformas móveis

	Android	**iOS**
Empresa	Google, Open Handset Alliance	Apple
Licença	Livre e código aberto	Proprietária
Linguagem de programação	Java, Kotlin e C++	Swift
IDE	Android Studio	Xcode
Plataformas	Windows, Linux, e Mac	Mac
Dispositivos	Diversos	Apple
Loja de aplicativos	Google Play	App Store

Fonte: (ANDROID, 2018a), (APPLE, 2018)

Devido a essa complexidade no desenvolvimento nativo, muitos desenvolvedores tem optado por *frameworks* multiplataformas, com o propósito de desenvolver o aplicativo somente uma vez, através de uma única base de código e executá-lo nas diferentes

plataformas.

Bjorn-Hansen, Groenli and Ghinea (2018) apontam que a adoção dessas ferramentas tem apresentado um crescimento nos últimos anos, principalmente pelo fato de reduzirem o custo, tempo e conhecimento necessário para o desenvolvimento de aplicativos móveis.

Ainda segundo os autores, essas ferramentas podem ser classificadas em diferentes abordagens de desenvolvimento:

- **Híbrida**: Nesta abordagem são utilizadas tecnologias web como o HTML, CSS e JavaScript, que são executadas e renderizadas por um componente WebView integrado a um aplicativo nativo de cada plataforma. Assim, o desenvolvedor pode programar a WebView para renderizar uma página HTML específica e programar o que é exibido para o usuário. Como os aplicativos híbridos são websites apresentados em um aplicativo nativo, pode haver dificuldade no desenvolvimento de uma UI seguindo os guias de desenvolvimento de interfaces disponibilizados pelas plataformas, como o Android Material Design ou o Apple Human Interface Guidelines. Para facilitar o processo de desenvolvimento de interface, existem diversos *frameworks*, entre os mais famosos está o Ionic. Nessa categoria, se destaca o *framework* Apache Cordova, que se tornou popular por simplificar o processo de desenvolvimento de aplicativos híbridos, podendo ser utilizado para inicializar e configurar o aplicativo nativo com a WebView incluída para todas as plataformas, e por possuir plugins para acessar componentes do dispositivo móvel, como câmera ou GPS.
- **Interpretada**: Como na abordagem híbrida, permitir que desenvolvedores utilizem uma linguagem de programação como o JavaScript para desenvolver os aplicativos. Porém os aplicativos interpretados não são executados em uma WebView e sim através de um interpretador JavaScript no dispositivo móvel (ex.: JavaScriptCore no iOS e V8 no Android). Uma das grandes vantagens dessa abordagem em relação a híbrida é que a interpretação torna possível a renderização de componentes de interface nativos na tela do usuário. Entre os *frameworks* mais utilizados nessa abordagem, está o React Native do Facebook.
- **Compilação multiplataforma**: Nesta abordagem, um único código do aplicativo, que pode ser desenvolvido em uma linguagem de programação como C#, é compilado pelo *framework* em um código de máquina nativo, para ser executado em cada plataforma. O acesso aos recursos do dispositivos são disponibilizados por SDKs

de cada plataforma. Entre as vantagens dessa abordagem em relação às anteriores, se destaca o desempenho do aplicativo desenvolvido e a forma como os componentes nativos da UI são renderizados. O Xamarin da Microsoft é um dos *frameworks* mais populares nessa categoria.

- **Model-Driven**: Baseada na metodologia de desenvolvimento de software *Model-Driven Developmen*t (MDD). *Frameworks* dessa abordagem facilitam a geração de interfaces de usuário e a lógica do aplicativo com base em modelos e *templates*. O código do aplicativo é desenvolvido utilizando uma linguagem de domínio específico (*Domain-Specific Language* - DSL) disponibilizado pelo *framework*, que é convertido em código nativo para as plataformas de destino. Isso habilita usuários que não tenham experiência em desenvolvimento móvel a criarem aplicativos com base em uma linguagem textual ou gráfica. Um exemplo de *framework* nessa categoria é o MD^2.
- **Progressive Web Apps**: Esta abordagem permite o desenvolvimento de aplicativos web que podem ser acessados a partir de uma URL em um navegador. Podem ser instalados por usuários e utilizados offline, sendo adicionados na tela de aplicativos do usuário, similarmente com o que acontece com os instalados pela loja oficial da plataforma. A interface é desenvolvida de forma similar aos aplicativos híbridos, através de HTML e CSS. Diferente do que acontece se acessados por um navegador, ao serem executados, não exibem barra de endereços e outros componentes de um navegador web, melhorando a experiência do usuário. O Angular é um exemplo de *framework* disponível para a construção desses aplicativos.

Como o propósito do aplicativo desenvolvido neste trabalho é oferecer aos usuários uma forma rápida e informativa de efetuar os cálculos, foi definida que a abordagem adotada seria a **compilação multiplataforma** com o Flutter, um *framework* introduzido recentemente pelo Google, com foco na construção de interfaces, possibilitando assim a construção de um aplicativo com bom desempenho e uma UI de qualidade.

2.3.1 Flutter

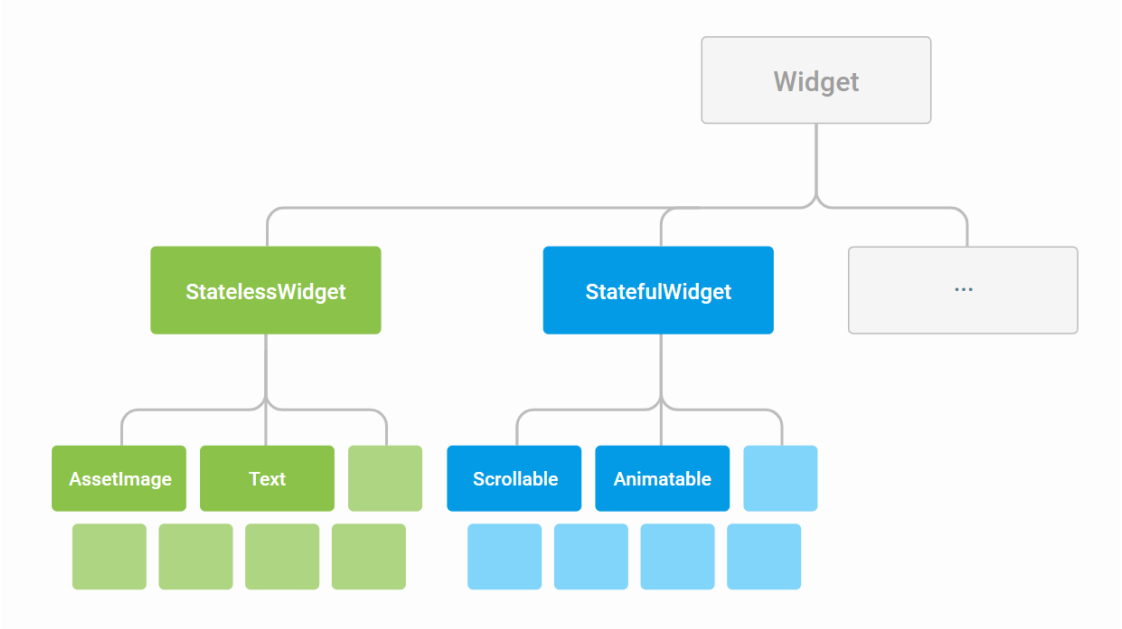
O Flutter[2] é o SDK de código aberto do Google que permite o desenvolvimento de aplicativos que executem tanto no Android quanto no iOS a partir de uma única base

[2]Disponível em: https://flutter.io/

de código. Seu objetivo é permitir que os desenvolvedores criem aplicativos de alta performance com uma experiência nativa em ambas as plataformas.

Seu fluxo de desenvolvimento é orientado ao design e os *widgets* são os blocos básicos da interface de usuário de um aplicativo Flutter. Assim, existem *widgets* para definir elementos estruturais (botões, menus...), elementos de estilo (fontes, cores...), aspectos de layouts (margens, espaçamentos...), além de *widgets* com design específico para a plataforma Android (Material Components) e iOS (Cupertino). Além disso, o Flutter foi projetado para facilitar a criação de novos *widgets* e a personalização dos existentes. Os *widgets* formam uma hierarquia baseada na composição onde cada *widget* herda propriedades de seu superior. A Figura 2.5 demonstra essa hierarquia.

Figura 2.5: Hierarquia de widgets



Fonte: https://flutter.io/technical-overview/

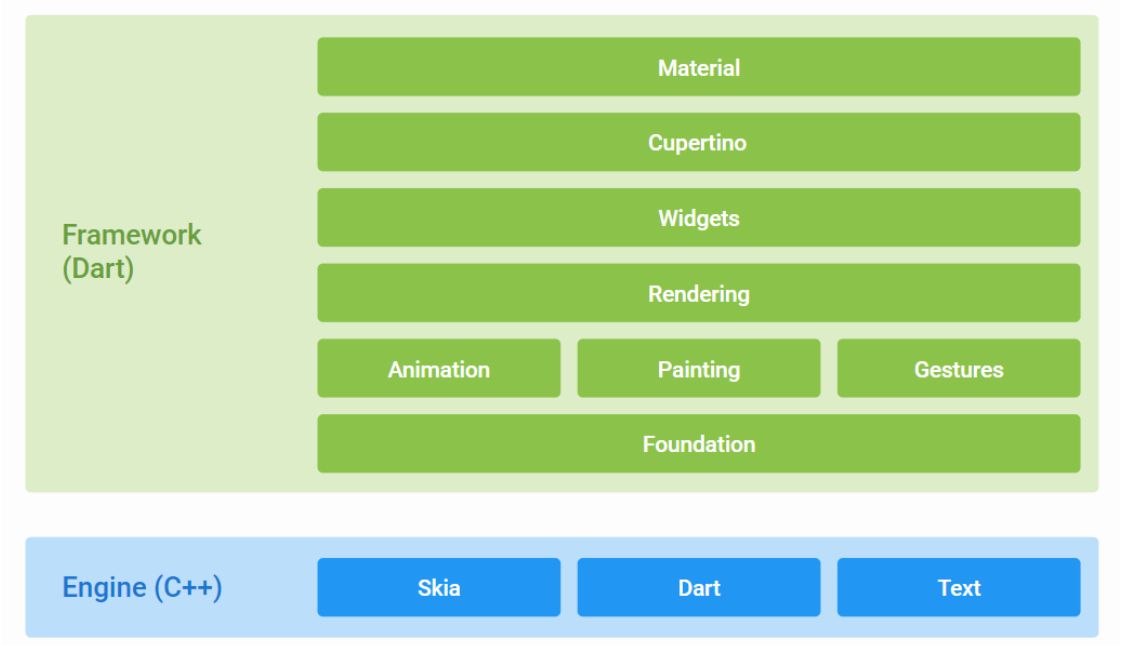
O que torna o Flutter diferente da maioria das outras opções para criar aplicativos móveis, é que ele não utiliza os *widgets* fornecidos com o dispositivo. Em vez disso, utiliza o seu próprio mecanismo de renderização de alto desempenho para desenhar *widgets*.

A Figura 2.6 representa a arquitetura do Flutter, que é composta pela *engine*, uma camada fina de código C/C++, e pelo *framework*, organizado em uma série de camadas, cada uma construída sobre a anterior e onde está implementada a maior parte de seu sistema (composição, gestos, animação, estrutura, *widgets*...). Essa implementação é feita em Dart[3], uma linguagem de programação moderna, concisa e fortemente tipada e orien-

[3]Disponível em: https://www.dartlang.org/

tada a objetos.

Figura 2.6: Arquitetura do Flutter



Fonte: https://flutter.io/technical-overview/

2.4 NoSQL

Ao iniciar o desenvolvimento de um aplicativo, um detalhe importante a ser considerado é sobre como armazenar os dados. Neste trabalho foi utilizado um banco de dados NoSQL pela grande facilidade de integração com o Flutter. Os conceitos relacionados a essa escolha serão apresentados a seguir.

As tecnologias de banco de dados podem ser divididas historicamente em três grandes gerações segundo Harrison (2015):

- A primeira geração foi impulsionada pelo surgimento do computador e de uma série de sistemas de bancos de dados que foram evoluindo cada vez mais.
- A segunda geração iniciou com o surgimento dos bancos de dados relacionais após a definição do modelo relacional em 1970 a partir das ideias de Edgar Codd. Nesse período quase todos os sistemas de bancos de dados significativos compartilhavam a mesma arquitetura. Os três pilares dessa arquitetura foram o modelo relacional, transações ACID e a linguagem SQL.
- A terceira geração surgiu no início dos anos 2000 motivada pela popularização

da internet e pelas novas demandas dos aplicativos modernos. Isso resultou em uma grande quantidade de novos sistemas de banco de dados que não aderiram às implementações relacionais tradicionais, entre eles os bancos de dados NoSQL.

Sadalage and Fowler (2012) sustentam que o NoSQL nasceu devido a necessidade das aplicações lidarem com grandes quantidades de dados (*big data*), difíceis de serem modeladas com bancos de dados relacionais.

Ainda segundo os autores, existem dois motivos principais que levam as pessoas a considerar a utilização de bancos de dados NoSQL.

O primeiro é a produtividade. Muito esforço no desenvolvimento de aplicativos é gasto no mapeamento de dados da memória para um banco de dados relacional. Um banco de dados NoSQL pode fornecer um modelo de dados que melhor atenda às necessidades do aplicativo, resultando em menos código para escrever, depurar e evoluir.

O segundo são os dados em grande escala. Para as organizações é importante capturar e processar uma grande quantidade de dados rapidamente. Elas consideram custoso fazer isso com bancos de dados relacionais. Isso se deve ao banco de dados relacional ser projetado para rodar em uma única máquina. Geralmente é mais econômico processar grandes quantidades de dados em *clusters* com diversas máquinas menores e mais baratas. Muitos bancos de dados NoSQL são projetados explicitamente para rodar em *clusters*, tornando-os melhor adaptados a esse cenário.

Bancos de dados NoSQL podem ser classificados de acordo com o seu modelo de dados. Os quatro principais são: chave-valor, documentos, família de colunas e grafo.

Para a elaboração do aplicativo, foi utilizado o Firebase Cloud Firestore, um banco de dados NoSQL disponibilizado recentemente pelo Google, cujo modelo é o de documentos.

2.4.1 Firebase Cloud Firestore

O Firebase Cloud Firestore[4] é um banco de dados NoSQL hospedado na nuvem, designado para aplicativos móveis, Web e servidores a partir do Firebase e do Google Cloud Platform. Permite que aplicativos iOS, Android e Web o acessem diretamente por meio de SDKs nativos.

O modelo de dados NoSQL do Cloud Firestore é o documento, onde dados são

[4]Disponível em: https://firebase.google.com/docs/firestore/

armazenados em documentos que contêm mapeamentos de chave para valores. Esses documentos são armazenados em coleções, utilizadas para organizar os dados e criar consultas. Os documentos são compatíveis com diversos tipos de dados diferentes, entre eles, strings, inteiros, números de ponto flutuante, booleanos, além de elementos complexos e aninhados. Os documentos podem conter subcoleções e é possível a criação de estruturas de dados hierárquicas que podem ser escalonadas à medida que o banco de dados cresce.

No Cloud Firestore é possível criar consultas superficiais para recuperar dados no nível do documento sem precisar recuperar a coleção inteira ou qualquer subcoleção aninhada. Isso garante uma grande eficiência na recuperação dos dados. Nas consultas podem ser adicionadas classificação, filtragem e limites. As consultas são indexadas por padrão, portanto, o desempenho é proporcional ao tamanho do conjunto de resultados, e não ao conjunto de dados.

O banco garante a sincronia dos dados em aplicativos clientes por meio de *listeners* em tempo real. Assim, os dados são mantidos atualizados instantaneamente nos aplicativos sem recuperar todo o banco de dados sempre que houver mudanças, recuperando somente as alterações.

Oferece suporte *off-line* para dispositivos móveis e Web para que os aplicativos funcionem independentemente de problemas na rede ou falta de conexão com a Internet, armazenando em cache os dados utilizados pelo aplicativo. Dessa maneira, o aplicativo poderá escrever, ler, detectar e consultar dados, mesmo que o dispositivo esteja sem Internet. Quando o dispositivo se conecta novamente, as alterações locais são sincronizadas novamente no Cloud Firestore.

2.5 FirebaseUI

O FirebaseUI[5] é um conjunto de bibliotecas de código aberto para o Firebase, que permite conectar rapidamente elementos de interface do usuário ao banco de dados do Firebase para armazenamento de dados, permitindo que as visualizações sejam atualizadas em tempo real conforme vão mudando e fornecendo interfaces simples para tarefas comuns como exibir listas ou coleções de itens (FIREBASE, 2018).

Além disso, o FirebaseUI simplifica a autenticação do Firebase ao fornecer métodos de autenticação fáceis de usar que se integram a provedores de identidade comuns como Facebook, Twitter e Google, além de permitir que os desenvolvedores usem uma

[5]Disponível em: https://opensource.google.com/projects/firebaseui

interface do usuário incorporada para facilitar o desenvolvimento (FIREBASE, 2018).

Neste trabalho foi utilizada uma versão do FirebaseUI adaptada para o Flutter, o ”flutter_firebase_ui” disponibilizado no *Dart Packages*[6], respositório de APIs oficial para o Flutter na linguagem Dart.

[6]Disponível em: https://pub.dartlang.org/packages/flutter_firebase_ui

3 APLICATIVO PROPOSTO: PROJETO E DESENVOLVIMENTO

Neste capítulo serão abordadas a metodologia utilizada no desenvolvimento do aplicativo, o aplicativo desenvolvido pelo coorientador que foi utilizado como base para o projeto do novo aplicativo, os requisitos para o desenvolvimento (demonstrados através de *user stories*), a forma como aplicativo foi sendo desenvolvido, bem como a apresentação de telas demonstrando as funcionalidades implementadas.

O aplicativo foi desenvolvido com base na metodologia ágil *Scrum* conforme apresentado em Schwaber (2004), porém, utilizando somente alguns aspectos relevantes ao desenvolvimento individual de software. Para isso, foram criadas *user stories* e a partir delas foi definido um *product backlog* com as funcionalidades a serem desenvolvidas, separadas por prioridades.

O projeto do aplicativo foi dividido em três *sprints* de 21 dias, espaços de tempo projetados para o desenvolvimento das funcionalidades presentes no *sprint backlog*, um subconjunto de itens selecionados do *product backlog*. Ao final de cada *sprint* uma versão funcional do aplicativo era disponibilizada, contendo as novas funcionalidades implementadas.

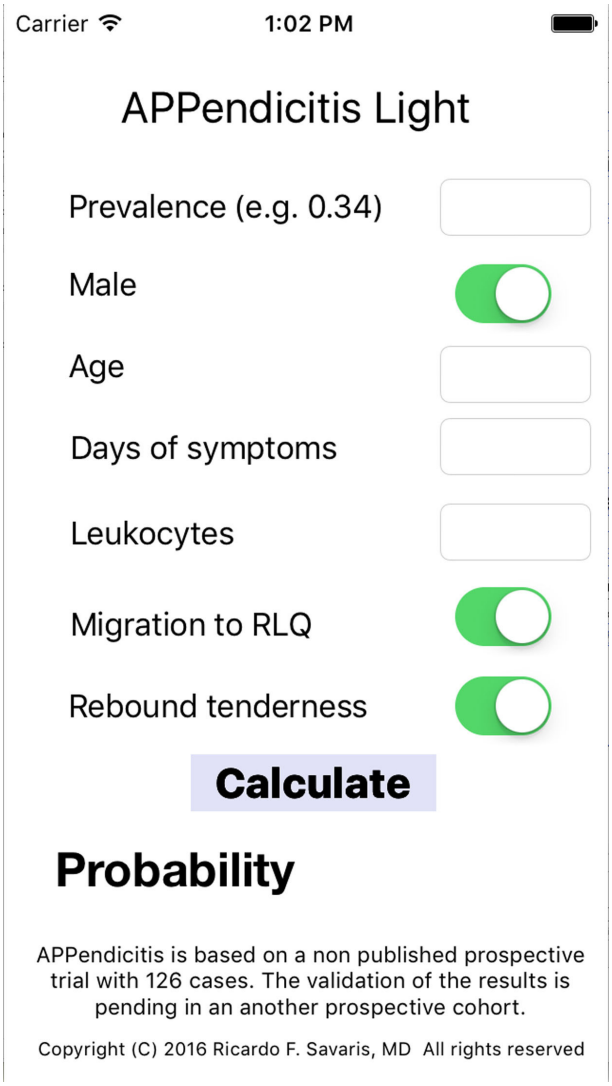
Para o gerenciamento do projeto e acompanhamento das *sprints* foi utilizada a ferramenta Trello[1], uma aplicação web desenvolvida com base no sistema Kanban, que utiliza cartões de sinalização para visualização do fluxo de trabalho (KNIBERG; SKARIN, 2010). A ferramenta disponibiliza um quadro contendo listas para a organização de tarefas. Os cartões exibem as descrições das tarefas e podem ser movidos entre as listas.

3.1 Aplicativo existente

O aplicativo desenvolvido pelo coorientador era composto somente por uma tela e apresentava algumas limitações. Uma delas é a simplicidade da interface, não apresentando valores de referência para todos os campos e não validando os dados inseridos pelo usuário, ou seja, caso dados incorretos fossem inseridos, resultados inconsistentes seriam exibidos. A interface desse aplicativo é demonstrada na Figura 3.1.

[1]Disponível em: http://trello.com/

Figura 3.1: Aplicativo desenvolvido pelo coorientador



Fonte: (APPSTORE, 2018)

O aplicativo desenvolvido neste trabalho foi projetado para superar as limitações do aplicativo existente. Além disso, novos recursos como sistema de login e armazenamento dos dados na nuvem deveriam ser implementados. Esses e outros requisitos foram modelados no formato de *User Stories*.

3.2 User Stories

Em um processo de desenvolvimento de um novo software, deve haver uma comunicação entre quem solicita esse software e o responsável por desenvolvê-lo. As *User Stories* (histórias de usuários) são utilizadas com frequência nesse processo pois descre-

vem de forma simples as funcionalidades que são importantes para um usuário final do sistema (COHN, 2004).

Neste trabalho, as *User Stories* seguiram o modelo descrito por Cohn (2008): *Como um <tipo de usuário>, eu quero <algum objetivo> para <algum motivo>*, sendo a última cláusula opcional.

A seguir estão descritas as *User Stories* criadas e validadas com o co-orientador durante o processo de desenvolvimento.

Tabela 3.1: User Story 1

Como um **usuário**, eu quero **acessar o aplicativo no meu dispositivo Android**.

Tabela 3.2: User Story 2

Como um **usuário**, eu quero **acessar o aplicativo no meu dispositivo iOS**.

Tabela 3.3: User Story 3

Como um **usuário**, eu quero **visualizar uma tela principal** para **inserir os dados do paciente**.

Tabela 3.4: User Story 4

Como um **usuário**, eu quero **visualizar textos informativos em cada campo de inserção** para **que fique claro o que deve ser inserido**.

Tabela 3.5: User Story 5

Como um **usuário**, eu quero **visualizar uma tela informando o resultado do cálculo** para **que eu possa vê-lo de forma clara**.

Tabela 3.6: User Story 6

Como um **usuário**, eu quero **fazer login no aplicativo** para **armazenar os meus cálculos**.

Tabela 3.7: User Story 7

Como um **usuário**, eu quero **cadastrar o meu email** para **poder fazer login no aplicativo**.

Tabela 3.8: User Story 8

Como um **usuário**, eu quero **utilizar minha conta do Facebook para fazer login** para **que não seja necessário me cadastrar**.

Tabela 3.9: User Story 9

Como um **usuário**, eu quero **utilizar minha conta do Google para fazer login** para **que não seja necessário me cadastrar**.

Tabela 3.10: User Story 10

Como um **usuário**, eu quero **acessar o aplicativo como visitante** para **que não seja necessário me cadastrar**.

Tabela 3.11: User Story 11

Como um **usuário**, eu quero **editar os dados armazenados** para **corrigir possíveis erros**.

Tabela 3.12: User Story 12

Como um **usuário**, eu quero **visualizar um menu** para **escolher entre a tela de cálculo e resultados**.

3.3 Processo de desenvolvimento

No início do processo de desenvolvimento, foram destinadas quatro semanas ao estudo dos conceitos relacionados às tecnologias utilizadas no aplicativo. Foram reali-

zadas diversas leituras para a decisão da adoção do Flutter como o *framework* de desenvolvimento multiplataforma utilizado no trabalho. Na sequência foi realizado um curso sobre esse *framework*, para adquirir domínio e prática na tecnologia. Finalmente foi efetuada uma pesquisa pela documentação do Firebase Cloud Firestore para compreender seu funcionamento e como utilizá-lo no aplicativo.

Após, deu-se início ao primeiro *sprint*, que teve por objetivo a entrega de uma versão básica do aplicativo. Nesse período foram implementadas as *user stories* 3, 4 e 5. Inicialmente foi criada uma tela para inserção dos dados do paciente e as entradas de texto com informações para o usuário sobre o que deve ser inserido, com exemplos. Na sequência foi criada uma classe com métodos para o cálculo do resultado com base nos dados inseridos e uma tela para exibir esse resultado para o usuário.

Para o segundo *sprint* foram acrescentadas tarefas baseadas em sugestões recebidas do co-orientador e foi priorizado o desenvolvimento da persistência dos dados do aplicativo. Fez-se necessária a mudança do campo de inserção originalmente chamado de “Prevalence” para “Incidence” e presença de uma descrição mais detalhada sobre os dados a serem inseridos nesse campo. Ao final do *sprint*, as *user stories* 6 e 7 estavam implementadas, o aplicativo tinha um sistema de login básico com e-mail e estava se comunicando com o banco de dados de forma a inserir e recuperar os dados.

No período referente ao *sprint* final foram implementadas as *user stories* 8, 9, 10, 11 e 12. Após novas conversas com o co-orientador, foram acrescentadas as opções para o usuário fazer login com o Facebook e Google, possibilitando uma alternativa ao registro de e-mail no aplicativo. Também foram acrescentados dois novos campos de inserção na tela principal para usuários logados, o primeiro para que o usuário possa definir o desfecho do diagnóstico após confirmação por patologia e outro para inserir observações.

Neste último *sprint*, também houve a necessidade de criar uma nova tela para exibir os resultados recuperados do banco de dados registrados pelo usuário e outra para tornar possível a edição de cada resultado, além de um menu lateral para acessar as telas do aplicativo.

Para tornar mais fácil o acesso aos usuários que não tem interesse em registrar os resultados dos cálculos, foi criada na tela inicial uma opção para o acesso como visitante, ou seja, sem a necessidade do registro por parte do usuário. Nesse caso, o aplicativo apresenta a funcionalidade básica de somente calcular e exibir o resultado.

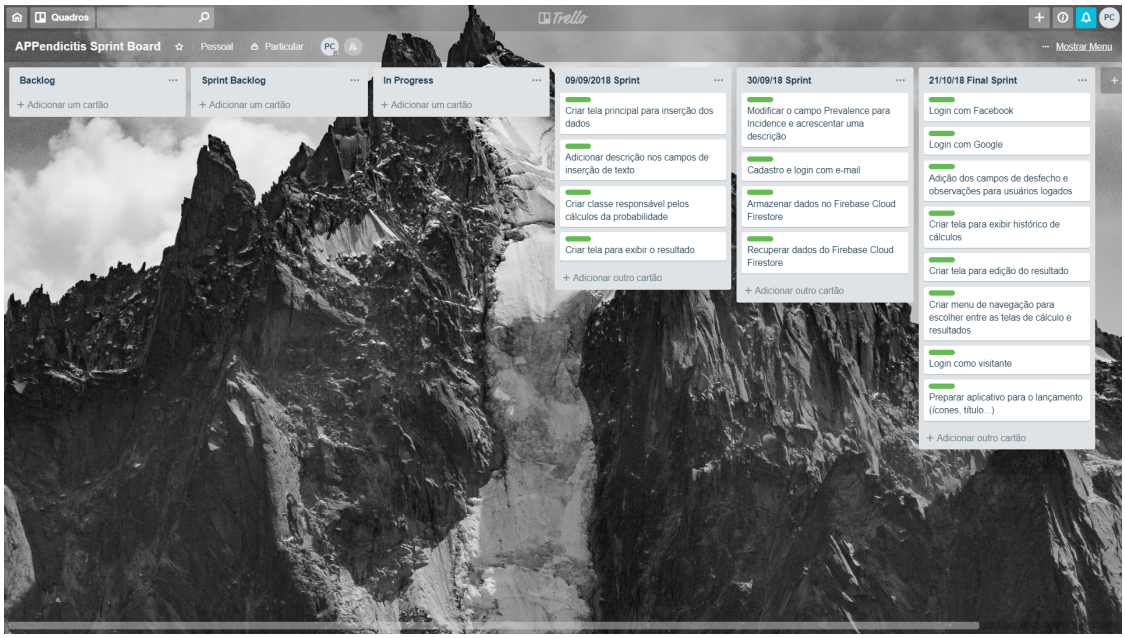
Ao final, o aplicativo foi preparado para o lançamento, com a revisão de aspectos de segurança, adição de ícones para a tela inicial e descrições mais detalhadas para o

usuário.

O aplicativo foi validado com uma série de dados reais, coletados pelo professor Ricardo Savaris, coorientador, os quais foram aplicados em seu modelo e os resultados aferidos.

Todo esse processo de desenvolvimento foi efetuado com o auxílio da ferramenta Trello. A Figura 3.2, demonstra a situação do processo após sua conclusão, assim como as funcionalidades disponíveis a cada data de entrega.

Figura 3.2: Tela do Trello após a entrega de todos os *sprints*



Fonte: Autor

3.4 Ambiente de desenvolvimento

Apesar de ser possível criar aplicativos com o Flutter utilizando qualquer editor de texto combinado com ferramentas de linha de comando, é recomendado o uso de *plug-ins* do Flutter desenvolvidos para IDEs específicas para uma melhor experiência. Esses *plug-ins* fornecem recursos como auto-completar, realce de sintaxe, assistências de edição de *widgets*, suporte a execução e depuração, entre outros (FLUTTER, 2018).

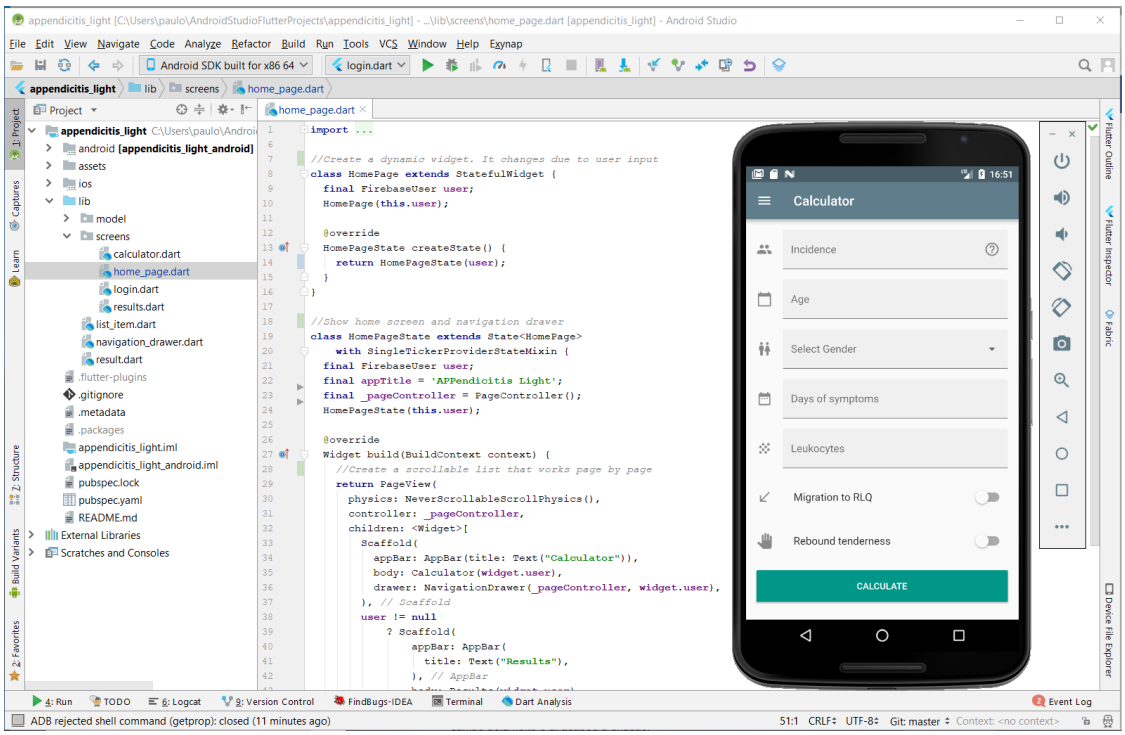
Uma das IDEs que suporta esse *plugin* é o Android Studio, a IDE oficial do Google para o desenvolvimento de aplicativos para Android, que com o auxílio de *plugins*, oferece uma experiência completa e integrada para o desenvolvimento com Flutter (FLUTTER, 2018).

Além disso, o Android Studio possui um emulador integrado para Android. Ao ser instalado em um MAC, é possível executar o aplicativo no simulador iOS. Isso torna muito rápido e simples os testes no aplicativo.

A IDE também possui uma fácil integração com o GitHub[2] para o versionamento do código.

Todos esses motivos fizeram com que o Android Studio fosse escolhido como ambiente de desenvolvimento para o aplicativo multiplataforma proposto nesse trabalho. A Figura 3.3 mostra o ambiente Android Studio configurado para o desenvolvimento com Flutter, e o aplicativo rodando no emulador Android.

Figura 3.3: Tela do Android Studio com o projeto do aplicativo com o Flutter



Fonte: Autor

3.5 Armazenamento de dados

Os dados são armazenados no aplicativo através de mapas chave-valor. A Tabela 3.13 descreve os campos e os tipos de dados que são armazenados no Firebase Cloud Firestore.

[2]Disponível em: https://github.com/

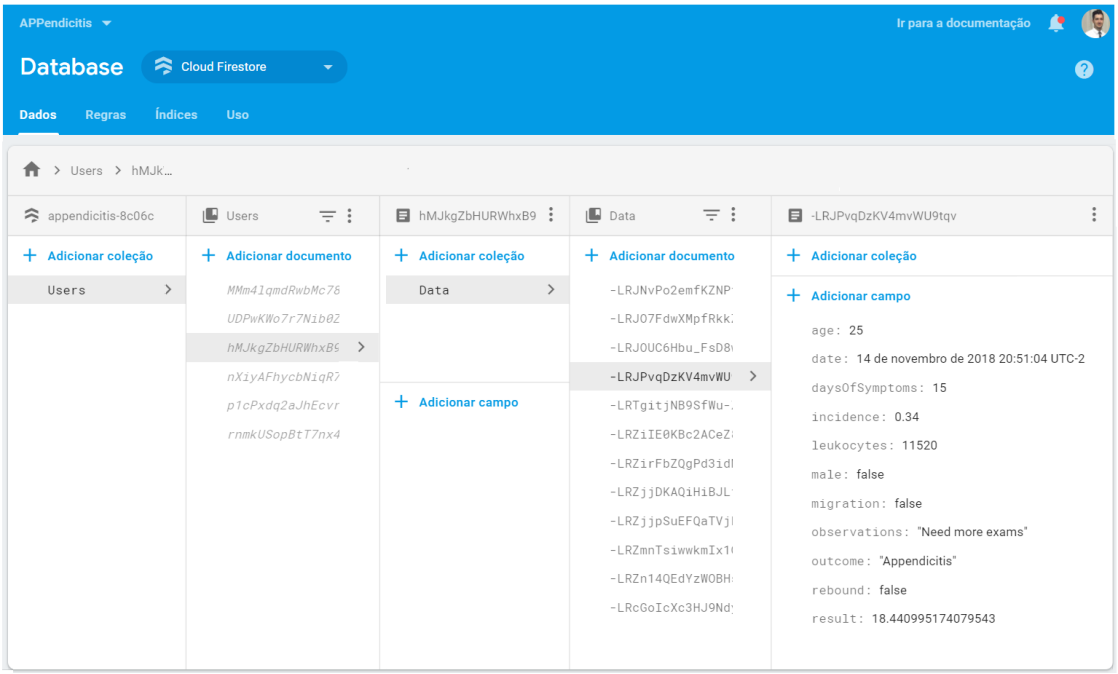
Tabela 3.13: Descrição dos dados armazenados

Campo	**Tipo**	**Descrição**
age	number	Idade
date	timestamp	Data do armazenamento
daysOfSymptoms	number	Dias apresentando sintomas
incidence	number	Incidência da doença
leukocytes	number	Número de leucócitos
male	boolean	Gênero
migration	boolean	Migração da dor abdominal
observations	string	Observações
outcome	string	Diagnóstico
rebound	boolean	Sinal de Blumberg
result	number	Resultado da probabilidade

Fonte: Autor

A forma como os dados estão armazenados no Firebase Cloud Firestore é representada pela Figura 3.4. A coleção *Users* é composta por documentos identificados pelo ID dos usuários do aplicativo. Cada usuário possui uma coleção *Data*, composta por documentos contendo os cálculos realizados.

Figura 3.4: Exemplo de dado armazenado no Firebase Cloud Firestore



Fonte: Autor

A classe *Probability* (Figura 3.5) gerencia os dados inseridos pelo usuário do aplicativo. Nela estão contidos os atributos a serem armazenados no banco de dados e todos os métodos para realizar os cálculos necessários para exibir o resultado para o usuário.

Figura 3.5: Classe Probability

probability.dart

```
1    class Probability {
2      double incidence;
3      bool male;
4      int age;
5      int daysOfSymptoms;
6      int leukocytes;
7      bool migration;
8      bool rebound;
9      double result;
10     String observations;
11     String outcome;
12
13     Probability() {...}
17
18     double calculateProbability() {...}
33
34     double calculateGenderLR(bool male) {...}
40
41     double calculateMigrationLR(bool painMitigation) {...}
47
48     double calculateBlumbergLR(bool rebound) {...}
54
55     double calculateLeukocytesLR(int leukocytes) {...}
160
161    double calculateDaysLR(int daysOfSymptoms) {...}
190
191    double calculateAgeLR(int age) {...}
256  }
257
```

Fonte: Autor

Após a finalização do preenchimento do formulário com os dados necessários para a realização do cálculo, um objeto da classe Probability é criado e os atributos desse objeto são armazenados no Firebase Cloud Firestore.

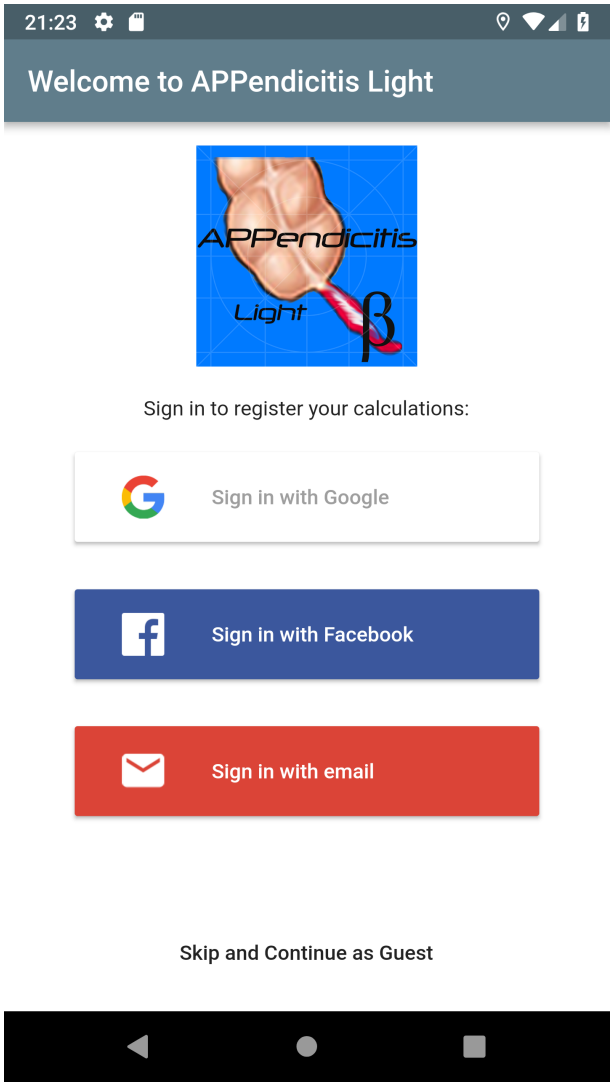
3.6 Funcionamento do aplicativo

Nesta seção será apresentada a interface do aplicativo, com a exibição das suas telas, seguidas de uma explicação sobre seu funcionamento. Optou-se por demonstrar essas telas executando em um dispositivo Android, pois a interface do aplicativo é similar

no Android e iOS. Por último, para demonstrar essa similaridade, será exibida a tela principal do aplicativo executando em um dispositivo iOS.

3.6.1 Tela de login

Figura 3.6: Tela de login



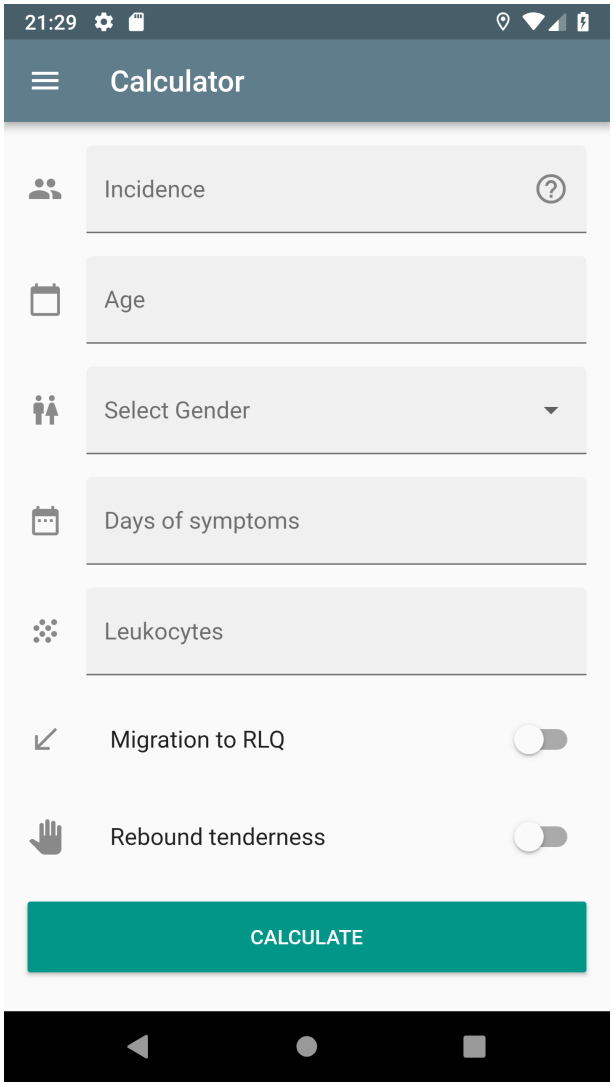
Fonte: Autor

Quando o usuário inicializa o aplicativo pela primeira vez, a tela da Figura 3.6 é exibida, oferecendo quatro possibilidades de acesso. Nesta tela é informado que o usuário deve fazer login para poder registrar os cálculos que efetuar no aplicativo. Para isso, tem a opção de fazer login utilizando os provedores Facebook e Google, ou registrando seu e-mail. Caso o usuário não tenha interesse em registrar seus cálculos, tem a opção de

acessar o aplicativo como visitante através do botão *Skip and Continue as Guest* (Pular e Continuar como Visitante), exibido na parte inferior da interface.

3.6.2 Tela principal

Figura 3.7: Tela principal para inserção dos dados do paciente



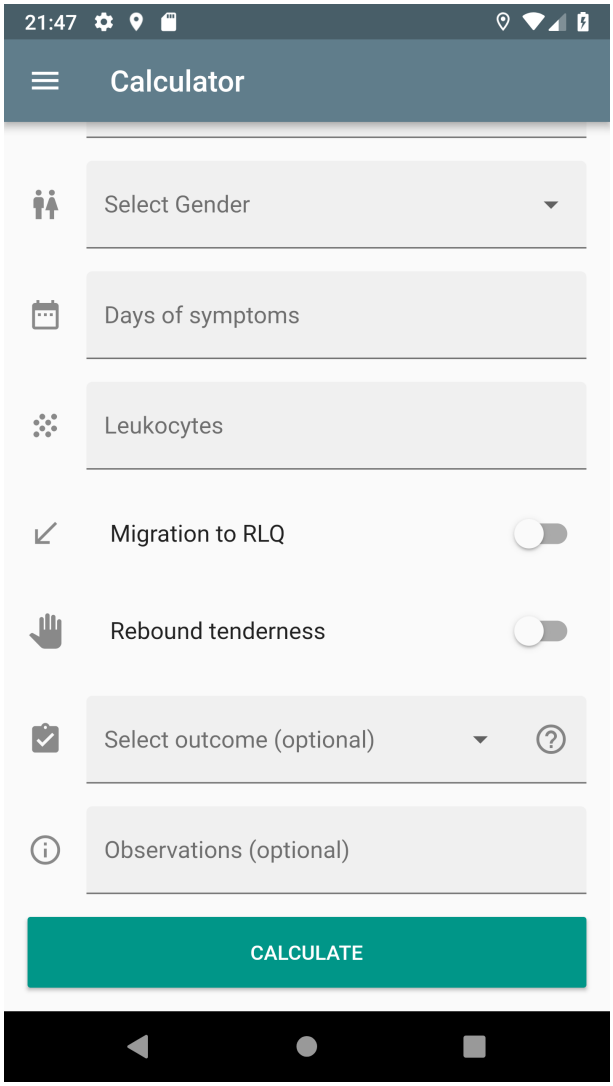
Fonte: Autor

Na tela principal (Figuras 3.7), o usuário pode inserir dados referentes ao paciente, nos seguintes campos:

- **Incidence**: Incidência de apendicite no local de trabalho do usuário em formato decimal, no intervalo de 0 a 1. Por exemplo, 0.34 representa 34% de incidência.
- **Age**: Idade do paciente.

- **Select Gender**: Gênero do paciente. Ao clicar neste campo é exibida a opção para selecionar entre Male (masculino) ou Female (feminino).
- **Days of symptoms**: Número de dias que o paciente tem apresentado sintomas.
- **Leukocytes**: Número de leucócitos do paciente, obtidos através de exame.
- **Migration to RLQ**: Permite indicar se a dor abdominal no paciente migrou para o quadrante inferior direito.
- **Rebound tenderness**: Sinal clínico que é observado durante o exame físico no abdômen de um paciente.

Figura 3.8: Tela principal para usuários registrados

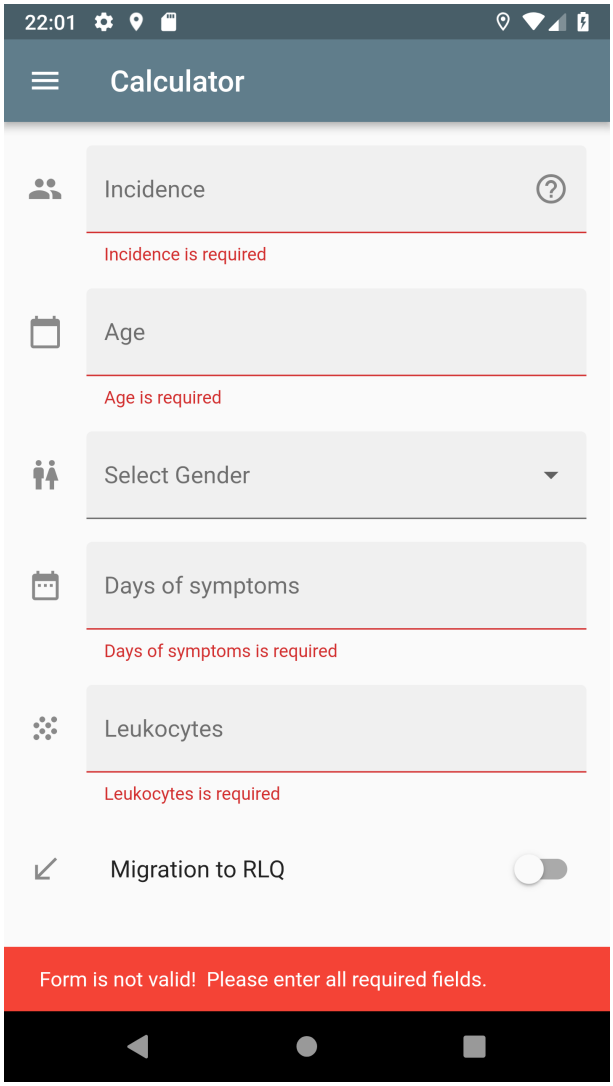


Fonte: Autor

No caso do usuário acessar o aplicativo de forma registrada, a tela principal ainda exibe os seguintes campos opcionais de inserção (dois últimos campos da Figura 3.8):

- **Select outcome**: Desfecho do diagnóstico do paciente. Pode ser acrescentado quando o diagnóstico foi confirmado por patologia. Ao clicar nesse campo são exibidas opções referentes a *Appendicitis* (Apendicite), *PID* (Doença inflamatória pélvica) e *Other* (Outra), no caso de não ter sido confirmada nenhuma das doenças anteriores.
- **Observations**: Destinado a inserção de informações relevantes para o registro do cálculo, como por exemplo, identificação do paciente.

Figura 3.9: Informação dos campos inseridos de forma inválida

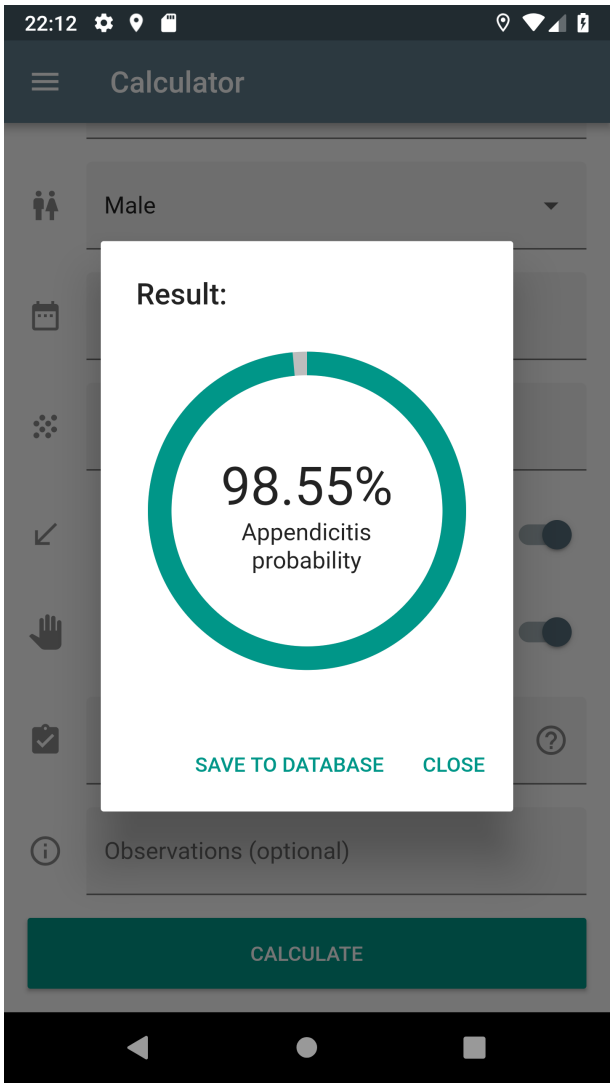


Fonte: Autor

Para evitar a inserção incorreta de dados, ou a falta de algum dado obrigatório, foi implementada uma validação nos campos ao clicar no botão ”CALCULATE”, indicando ao usuário quais campos apresentam problemas (Figura 3.9).

3.6.3 Tela com o resultado

Figura 3.10: Tela para a exibição do resultado



Fonte: Autor

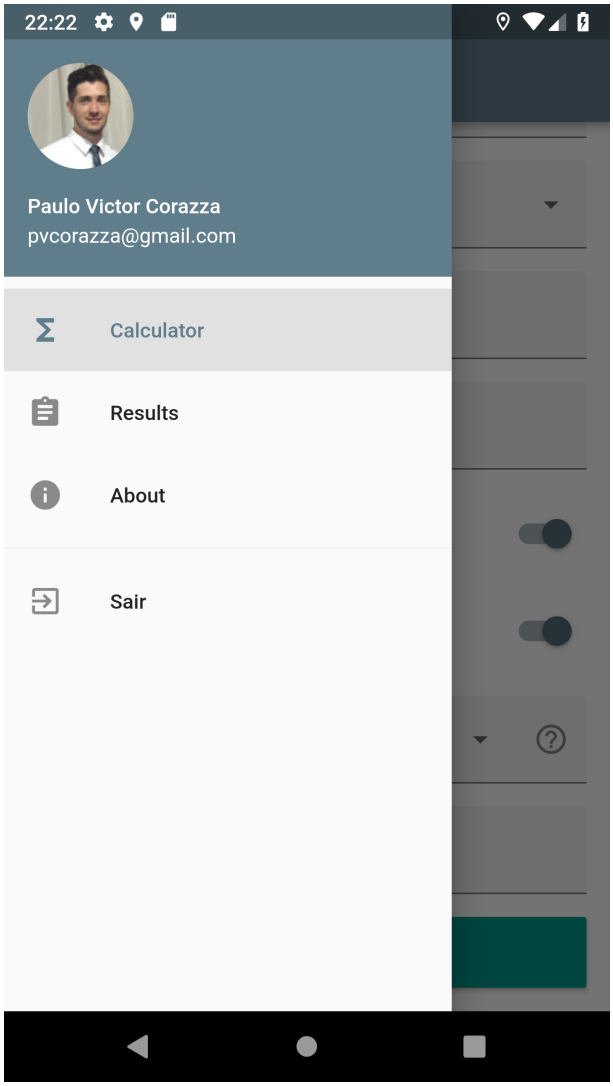
Para exibir o resultado com a probabilidade do paciente ter apendicite, uma tela é sobreposta a tela principal. O resultado é exibido de duas formas, através da indicação textual centralizada com duas casas decimais, e através de um círculo externo para indicar o progresso da porcentagem.

Esse resultado pode servir de apoio para auxiliar o médico na escolha do melhor tratamento para seu paciente. Por exemplo, uma probabilidade elevada de apendicite como a demonstrada na Figura 3.10, pode levar o médico a realizar exames complementares relacionados a doença.

O usuário registrado ainda tem a opção de registrar os dados inseridos e o resultado, salvando no banco de dados. Essa opção não é exibida para o usuário visitante.

3.6.4 Telas com a navegação lateral

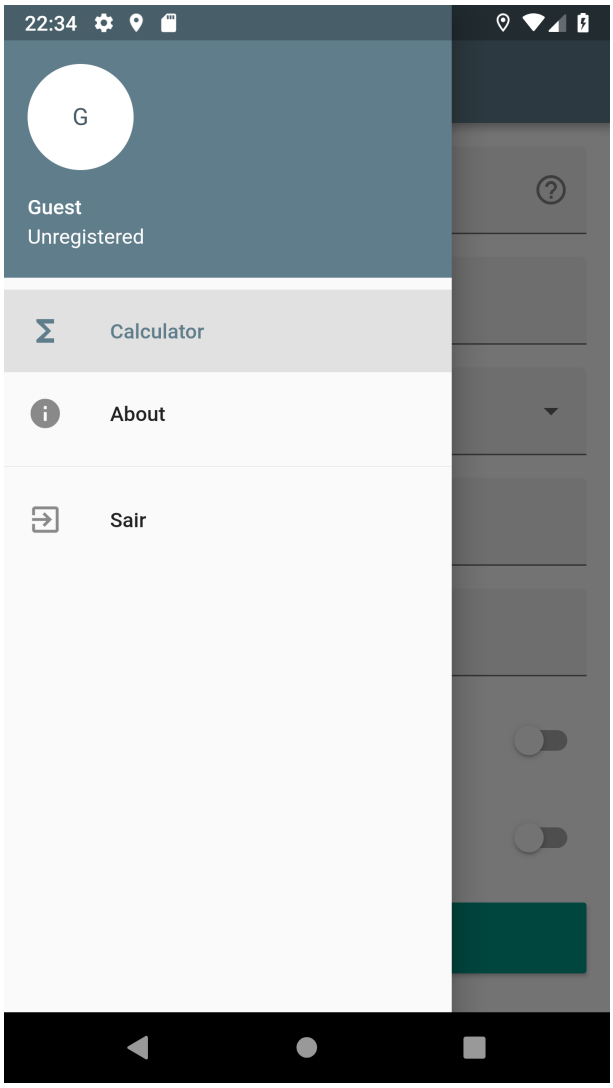
Figura 3.11: Tela exibindo um menu de navegação lateral



Fonte: Autor

Para navegar entre as telas do aplicativo, o usuário dispõe de um menu lateral que é expandido ao clicar no botão superior esquerdo da tela principal. Na parte superior é exibida a foto, nome e e-mail do usuário, obtidas pelo provedor que o usuário usou para se registrar, no caso do Facebook ou Google, ou conforme os dados informados no registro com e-mail (Figura 3.11).

Figura 3.12: Tela com a navegação lateral para visitantes

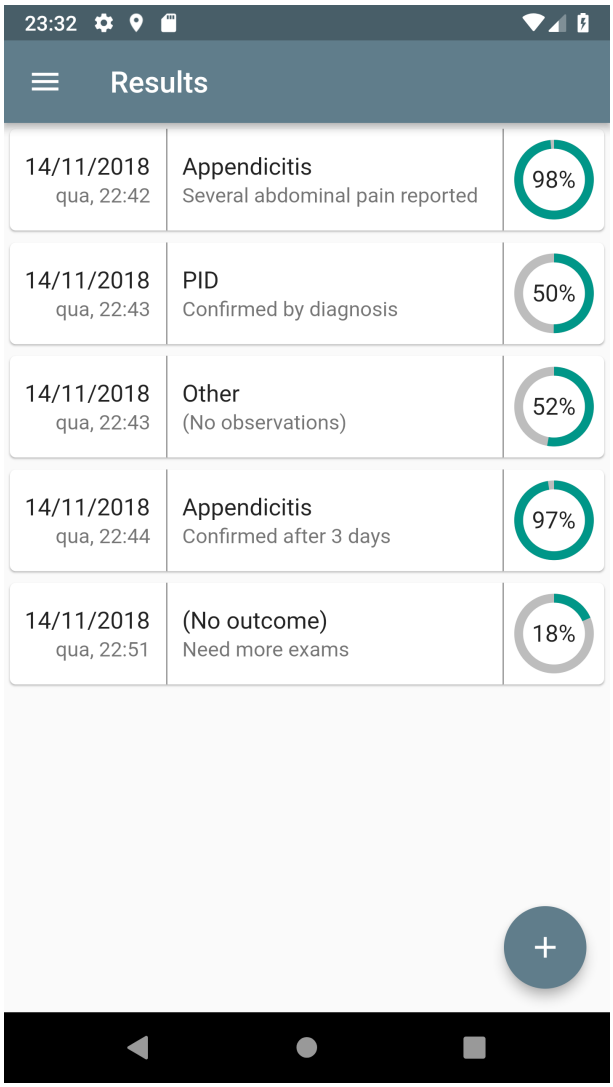


Fonte: Autor

No caso de o usuário acessar o aplicativo como visitante, a opção referente ao acesso a tela de resultados registrados fica oculta, conforme mostra a Figura 3.12.

3.6.5 Tela de resultados

Figura 3.13: Tela com a lista de resultados registrados



Fonte: Autor

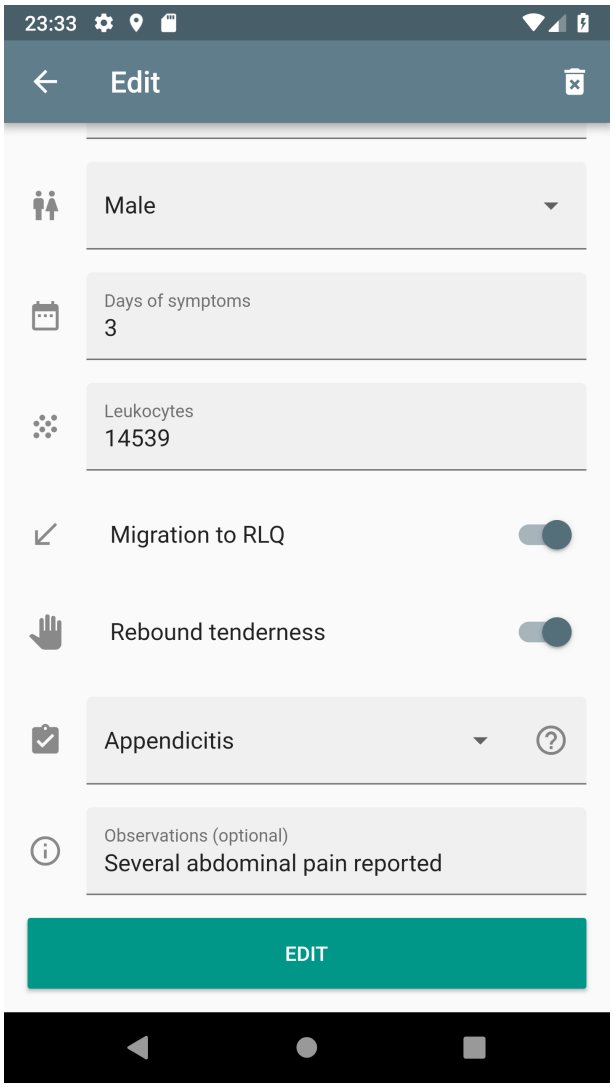
A tela de resultados pode ser acessada pelos usuários registrados ao clicar no segundo item da barra de navegação lateral.

Nesta tela o médico tem uma visão geral do histórico de seus pacientes, ordenado pela data de registro através de uma lista com os dados que optou por armazenar. Cada item da lista apresenta na esquerda, a data de inserção no banco de dados, no centro, as informações de diagnóstico e as observações, e na direita, a probabilidade de apendicite.

Na parte inferior direita, existe um botão para a rápida adição de novos cálculos.

3.6.6 Tela de edição

Figura 3.14: Tela para edição/remoção dos dados

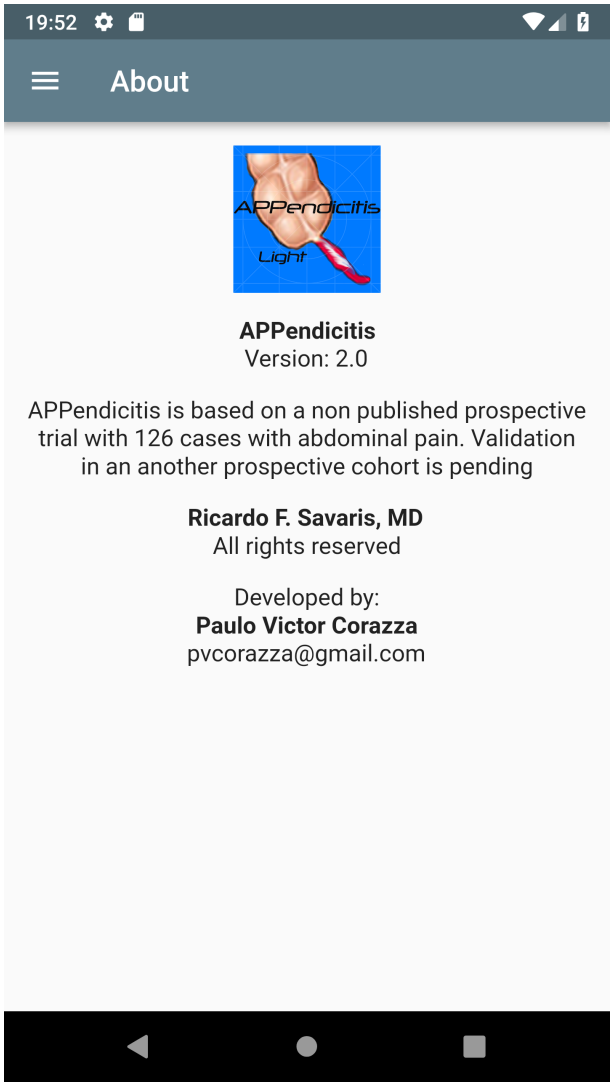


Fonte: Autor

Ao clicar em cada item da lista exibido na Figura 3.13, é exibida uma tela que permite a edição dos dados inseridos anteriormente (Figura 3.14). Na parte superior direita, existe um ícone que permite a remoção desses dados.

3.6.7 Tela de informações

Figura 3.15: Tela com informações sobre o aplicativo

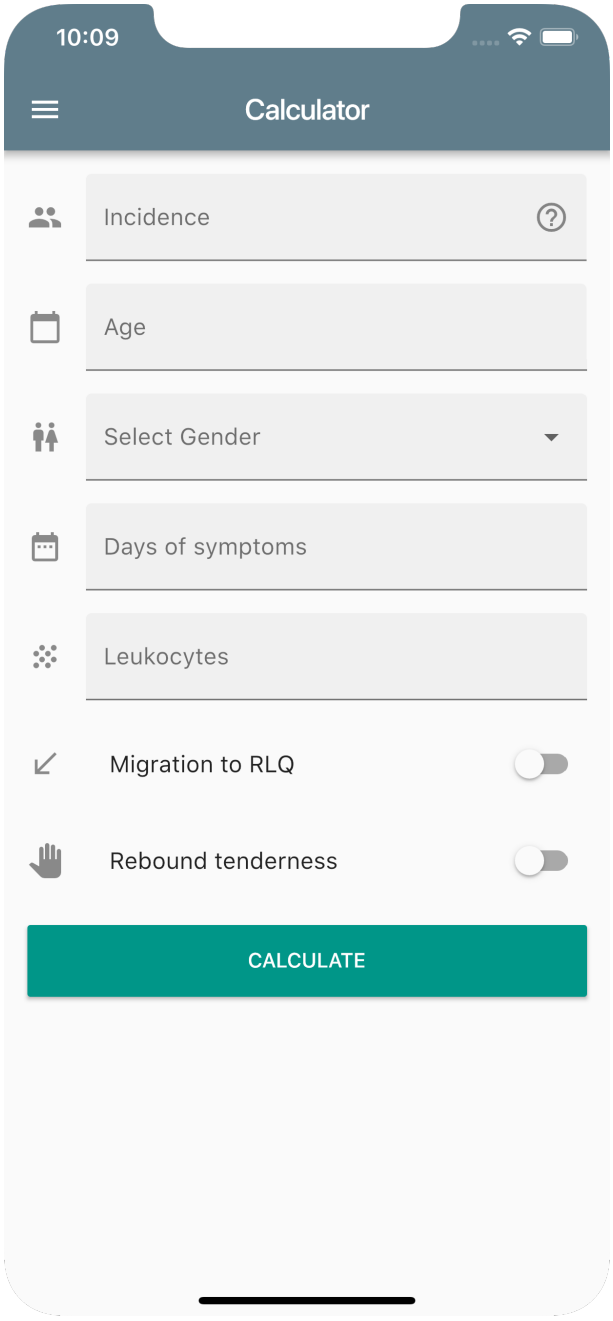


Fonte: Autor

No menu lateral da Figura 3.11, ao clicar em ”About”, é exibida a tela da Figura 3.15, contendo informações sobre a versão do aplicativo, autores e informações sobre o estudo que baseou o desenvolvimento do aplicativo.

3.6.8 Tela principal no iOS

Figura 3.16: Tela principal para inserção dos dados do paciente em um dispositivo iOS



Fonte: Autor

Na Figura 3.16 é apresentada a interface do aplicativo executando em um iPhone XS Max com iOS. Percebe-se a similaridade com a Figura 3.7, onde a mesma tela é exibida em um dispositivo Android. Todas as demais telas do aplicativo executando no iOS mantém essa similaridade com as já apresentadas no Android. Por isso, optou-se por

não inseri-las no trabalho.

3.7 Requisitos de sistema

Para o correto funcionamento do aplicativo, o usuário deverá dispor, minimamente, dos seguintes requisitos:

- **Smartphone ou Tablet**: Dispositivos Android na versão Jelly Bean (4.1.x) ou mais recente, ou dispositivos iOS (iPhone 4S ou mais recente), na versão 8 ou superior.
- **Acesso à Internet**: Para o funcionamento básico de cálculo da probabilidade, não é necessário acesso a Internet. Porém, para login e registro dos resultados, o acesso à rede (móvel ou Wi-Fi) é imprescindível.

4 AVALIAÇÃO DO APLICATIVO

Este capítulo apresentará uma análise dos resultados obtidos após uma pesquisa com doze (12) participantes, para avaliação da usabilidade do aplicativo desenvolvido neste trabalho.

Como o aplicativo está inserido na área médica, a pesquisa de usabilidade foi realizada com estudantes e profissionais da área, com o objetivo de verificar a sua facilidade de uso, possibilitando que eventuais problemas pudessem ser corrigidos em versões futuras.

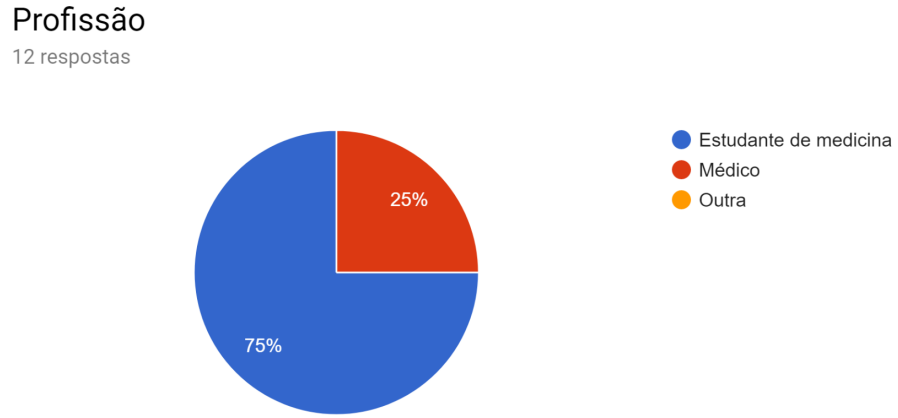
Para a obtenção dos resultados, foi elaborado um questionário no Google Forms (Apêndice A). Nesse questionário estão descritos o experimento, os links para que os participantes pudessem obter o aplicativo e as instruções para que realizassem algumas tarefas.

No questionário são solicitadas informações acerca do perfil do participante e uma avaliação sobre a facilidade de realizar as tarefas solicitadas. Ao final os participantes deveriam responder o questionário padronizado *System Usability Scale* (SUS).

4.1 Perfil dos participantes

Inicialmente foi solicitada a profissão dos participantes, de forma a assegurar que o aplicativo estaria sendo avaliado pelo público ao qual é destinado. Assim, foi identificado que 3 médicos e 9 estudantes de medicina responderam ao questionário (Figura 4.1).

Figura 4.1: Profissão dos participantes

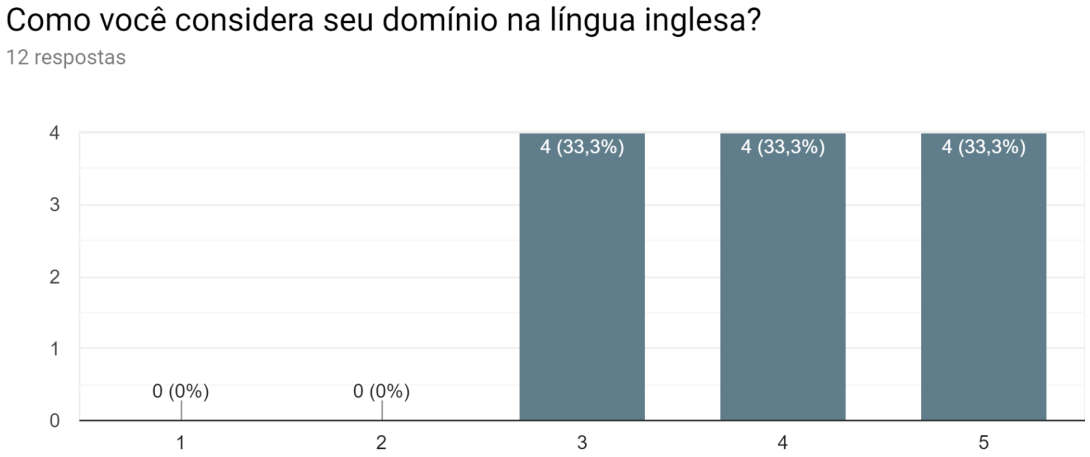


Fonte: Autor

Como o aplicativo foi desenvolvido em inglês, houve a necessidade de verificar

o domínio da linguagem por parte dos participantes, que deveriam informá-lo em uma escala de 1 a 5. O resultados apontam que os participantes possuem um domínio suficiente para compreender quais dados deveriam informar no aplicativo e conseguir realizar as tarefas solicitadas, uma vez que os valores informados se situaram entre 3 e 5.

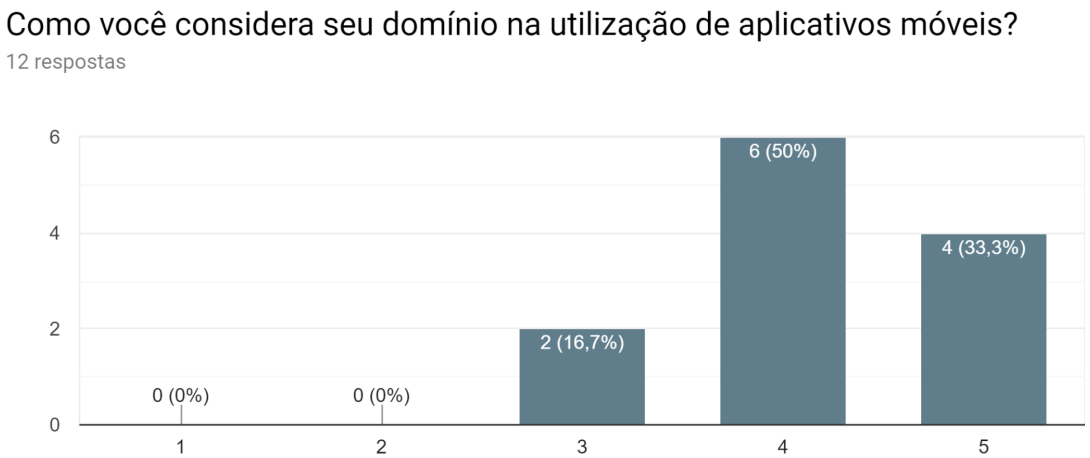
Figura 4.2: Domínio na língua inglesa dos participantes



Fonte: Autor

Também foi questionada a familiaridade dos participantes com o uso de aplicativos móveis. Um usuário que tenha um bom domínio no uso de aplicativos, tem melhores condições de avaliar a usabilidade, em comparação com outros aplicativos que utilizam. Os resultados demonstram que 83,3% dos participantes consideram seu nível de domínio entre 4 e 5, representando boas condições para avaliar o aplicativo.

Figura 4.3: Domínio na utilização de aplicativos móveis por parte dos participantes

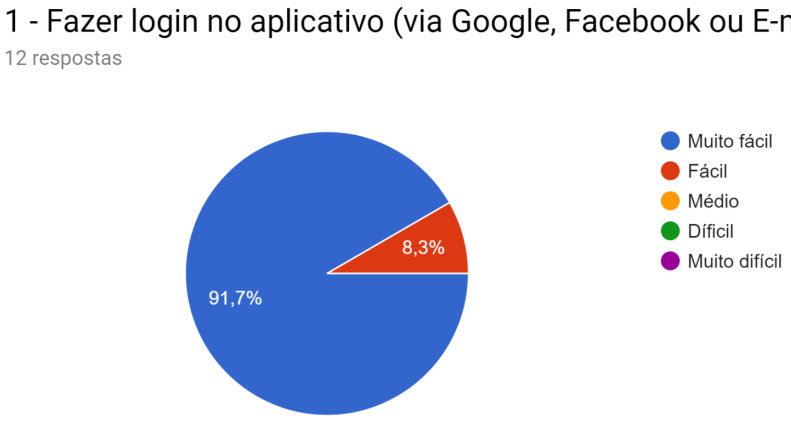


Fonte: Autor

4.2 Facilidade na realização das tarefas

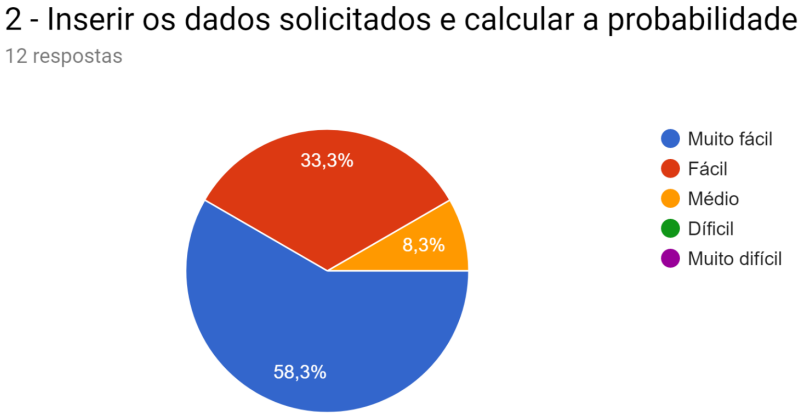
O roteiro de tarefas a ser realizado está demonstrado nas figuras seguintes e contemplou basicamente todas as funcionalidades desenvolvidas, permitindo que o participante pudesse explorar todas as possibilidades de navegação na interface do aplicativo.

Figura 4.4: Tarefa 1



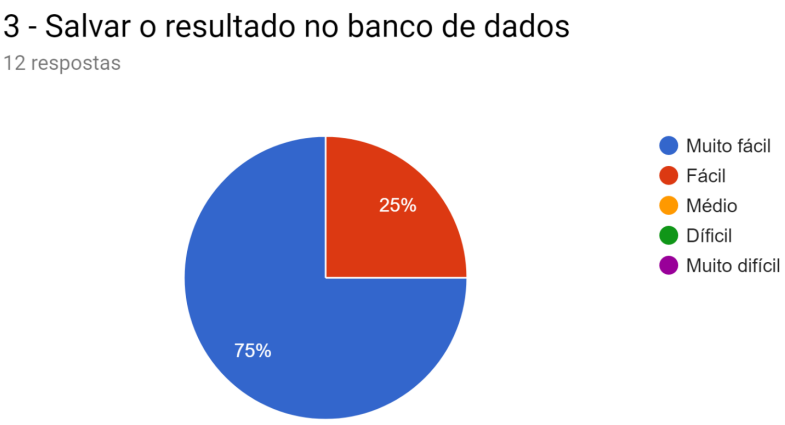
Fonte: Autor

Figura 4.5: Tarefa 2



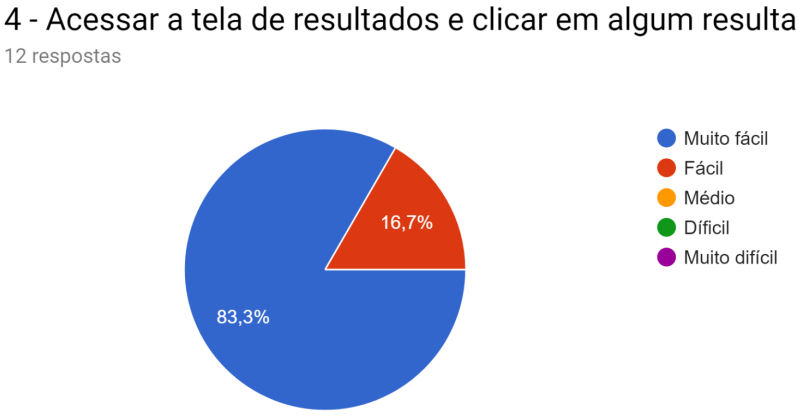
Fonte: Autor

Figura 4.6: Tarefa 3



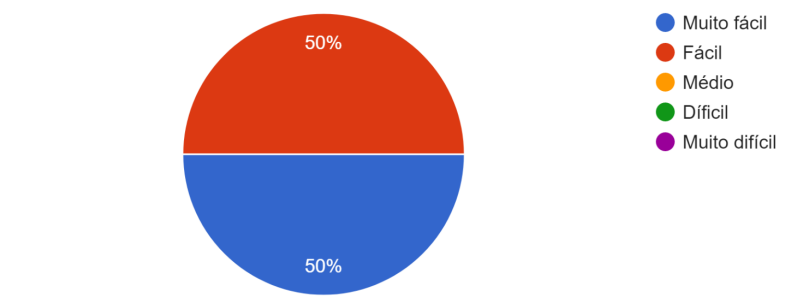
Fonte: Autor

Figura 4.7: Tarefa 4



Fonte: Autor

Figura 4.8: Tarefa 5



Fonte: Autor

Figura 4.9: Tarefa 6

Fonte: Autor

Os resultados apontam que, de um modo geral, os participantes não tiveram dificuldades para realizar as tarefas solicitadas. Mesmo a Tarefa 2 que exige dos participantes uma leitura e compreensão sobre quais dados devem ser inseridos, teve os resultados positivos, com a grande maioria dos usuários considerando-a como sendo fácil ou muito fácil.

Quanto a Tarefa 6, observa-se que um participante avaliou a realização da tarefa com um nível ”Médio”. Além disso, houve uma redução no número de participantes que consideraram a tarefa ”Muito fácil” em relação as outras tarefas. Nesta tarefa, foi solicitado que o participante salvasse o resultado após a edição realizada na tarefa anterior. Antes de salvar o resultado, é exibida a tela com a probabilidade de apendicite recalculada para o caso de algum outro dado do paciente ter sido editado. Porém, ao somente adicionar o diagnóstico e observações, não haveria a necessidade de recalcular a probabilidade, visto que os dados do paciente permanecem os mesmos. O usuário pode ter considerado estranho esse passo a mais, o que provocou um pequeno aumento no nível de dificuldade da tarefa em relação as outras.

4.3 System Usability Scale

A última parte da avaliação consistia na realização do questionário SUS, de forma que ao final do experimento pudesse ser gerada uma pontuação para a usabilidade do aplicativo.

O SUS é uma escala numérica de usabilidade descrito por Brooke (1996) para avaliar a usabilidade de uma variedade de produtos ou serviços. É um questionário simples,

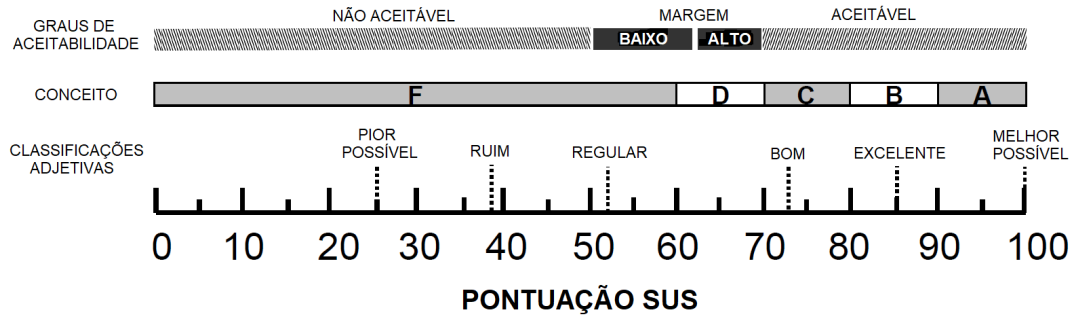
composto por 10 itens, onde o participante deve indicar o seu grau de concordância ou discordância com cada afirmação em uma escala de 0 (discordo totalmente) a 5 (concordo totalmente).

A pontuação do SUS é calculada a partir da pontuação individual de cada item, que varia de 0 a 4. Para os itens ímpares a pontuação é a resposta do participante menos 1. Para os itens pares, a pontuação é 5 menos a resposta do participante. A pontuação final é obtida com a soma as pontuações individuais de todos os itens multiplicada por 2,5. Assim, é obtida uma pontuação que varia em um intervalo de 0 a 100.

Para Bangor, Kortum and Miller (2009), o SUS provou ser uma ferramenta robusta, tendo sido constantemente utilizado para avaliar uma ampla gama de interfaces. O seu uso é atrativo por ser gratuito e por ser composto de apenas dez afirmações, sendo relativamente rápido e fácil para os participantes concluírem.

Os mesmos autores constataram a partir de seus 273 estudos que a média de pontuação SUS tem se mantido em torno de 70 nos mais diversos tipos de interface e estabeleceram classificações para interpretar a pontuação individual do resultado da aplicação de um questionário SUS. A Figura 4.10 apresenta essas classificações.

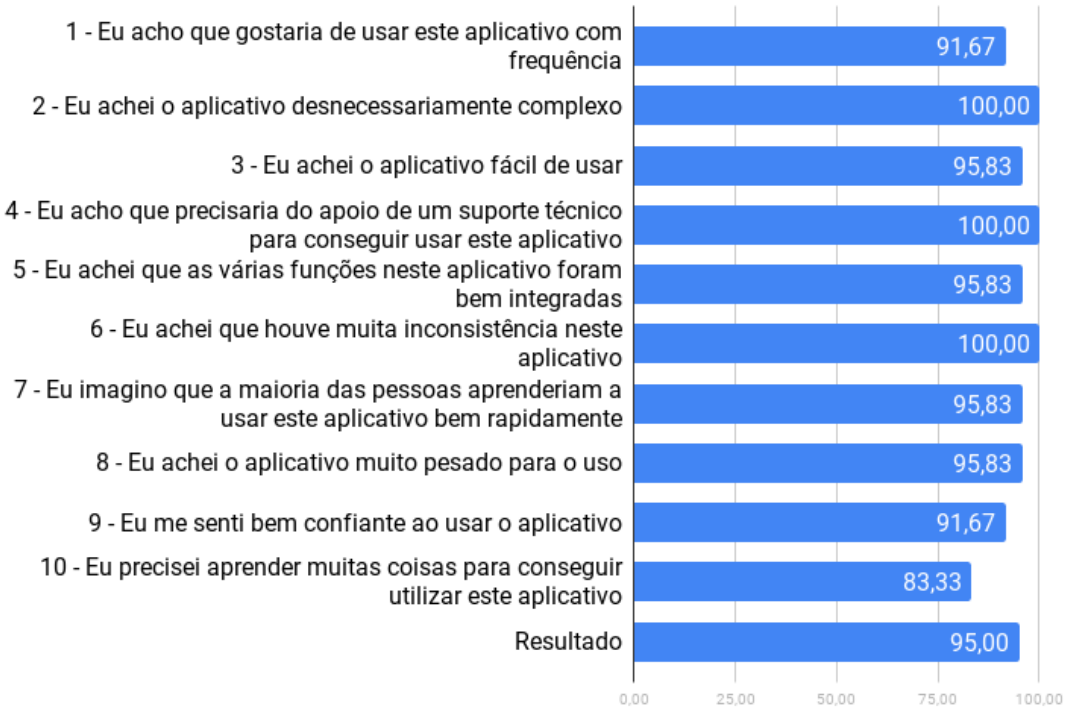
Figura 4.10: Classificações em relação à pontuação SUS



Fonte: Traduzido e adaptado de (BANGOR; KORTUM; MILLER, 2009)

A Figura 4.11 demonstra a pontuação SUS obtida a partir da avaliação do aplicativo. Observa-se pontuações acima de 90 em todos os itens, com exceção do item 10, com pontuação de 83,33 que se refere aos conhecimentos prévios exigidos para utilização do aplicativo. A pontuação final ficou em 95.

Figura 4.11: Questionário SUS - Resultados



Fonte: Autor

Tomando como referência A Figura 4.10, a pontuação obtida pelo aplicativo possui um grau aceitável, conceito A e classificação adjetiva excelente, demonstrando que o aplicativo possui boa usabilidade, ou seja, os usuários tiveram facilidade para interagir com a interface e de uma forma eficiente.

5 CONCLUSÃO

A elaboração deste trabalho resultou em um aplicativo móvel para auxiliar médicos na tomada de decisões clínicas acerca do diagnóstico da apendicite, superando as limitações existentes no aplicativo até então utilizado.

Foram exploradas tecnologias modernas de desenvolvimento multiplataforma como o Flutter, que permitiu tornar o aplicativo disponível para as duas plataformas móveis predominantes no mercado. Com essa limitação superada, espera-se uma maior utilização do aplicativo por parte dos profissionais de medicina que atuam na área.

Com o emprego do Firebase Cloud Firestore, foi possível implementar o armazenamento dos dados na nuvem e diretamente a partir do aplicativo, eliminando a necessidade de registrar essas informações em alguma ferramenta externa.

A utilização de uma versão adaptada da metodologia ágil, envolveu a criação de *user stories* para orientar a implementação das funcionalidades, que foram sendo validadas com o coorientador do trabalho ao final de cada sprint. Após o último sprint, a implementação do aplicativo foi validada com dados reais pelo coorientador, para garantir que o aplicativo estava exibindo os resultados de forma correta. Todo esse processo resultou em um aplicativo que atendeu a necessidade inicialmente identificada, embora seja possível a inserção de outras funcionalidades a partir de novos requisitos.

Identificar se o aplicativo atendia ao atributo da usabilidade era fundamental. Neste sentido, foi aplicado um questionário especificamente para profissionais e futuros profissionais da medicina, buscando a opinião de quem possui conhecimento na área, possibilitando uma avaliação mais criteriosa sobre a qualidade do aplicativo. Os resultados dessa avaliação foram considerados positivos, uma vez que esses usuários não tiveram dificuldades para realizar as tarefas propostas. A pontuação SUS de 95, sugere que o aplicativo possui um grau elevado de usabilidade e poderá contribuir para que os profissionais da área obtenham resultados mais rápidos e confiáveis em relação ao diagnóstico de apendicite.

Trabalhos futuros poderão explorar formas de exportar os resultados armazenados no banco de dados para a obtenção de relatórios, ou mesmo, o desenvolvimento de uma aplicação web, aproveitando o conceito de interface já desenvolvido, para que os usuários possam realizar os cálculos e acessar os resultados também em outros dispositivos.

REFERÊNCIAS

ANDROID. **Android**. 2018. Disponível em: <https://www.android.com/>. Acessado em: 31/10/2018.

ANDROID. **Android Platform Architecture**. 2018. Disponível em: <https://developer.android.com/guide/platform/>. Acessado em: 27/10/2018.

APPLE. **iOS Technology Overview**. 2014. Disponível em: <https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html>. Acessado em: 25/10/2018.

APPLE. **Apple**. 2018. Disponível em: <https://www.apple.com/>. Acessado em: 31/10/2018.

APPSTORE. **APPendicitis**. 2018. Disponível em: <https://itunes.apple.com/us/app/appendicitis/id1165848743>. Acessado em: 08/11/2018.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, Usability Professionals' Association, v. 4, n. 3, p. 114–123, 2009.

BJORN-HANSEN, A.; GROENLI, T.; GHINEA, G. A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. 2018.

BROOKE, J. Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London–, v. 189, n. 194, p. 4–7, 1996.

COHN, M. **User stories applied: For agile software development**. [S.l.]: Addison-Wesley Professional, 2004.

COHN, M. **Advantages of the "As a user, I want" user story template**. 2008. Disponível em: <https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template>. Acessado em: 20/10/2018.

EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. **Ain Shams Engineering Journal**, Elsevier BV, v. 8, n. 2, p. 163–190, jun 2017. Available from Internet: <https://doi.org/10.1016/j.asej.2015.08.004>.

FIREBASE. **FirebaseUI**. 2018. Disponível em: <https://opensource.google.com/projects/firebaseui>. Acessado em: 04/11/2018.

FLUTTER. **Editor**. 2018. Disponível em: <https://flutter.io/docs/get-started/editor>. Acessado em: 30/10/2018.

GARGENTA, M.; NAKAMURA, M. **Learning Android**. Sebastopol, CA: O'Reilly Media, 2014. ISBN 978-1-449-31923-6.

GARTNER. **Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018**. 2018. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2018-05-29-gartner-says-worldwide-sales-of-smartphones-returned-to-growth-in-first-quarter-of-2018>. Acessado em: 31/10/2018.

HARRISON, G. **Next Generation Databases**. [S.l.]: Apress, 2015.

KNIBERG, H.; SKARIN, M. **Kanban and Scrum-making the most of both**. [S.l.]: Lulu, 2010.

LECHETA, R. R. **Google Android - Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 5. ed. [S.l.]: Novatec Editora, 2015.

LEVIN, J. **Mac OS X and IOS Internals: To the Apple's Core**. [S.l.]: John Wiley & Sons, 2012.

SADALAGE, P. J.; FOWLER, M. **NoSQL Distilled**. 1st. ed. [S.l.]: Addison-Wesley Professional, 2012.

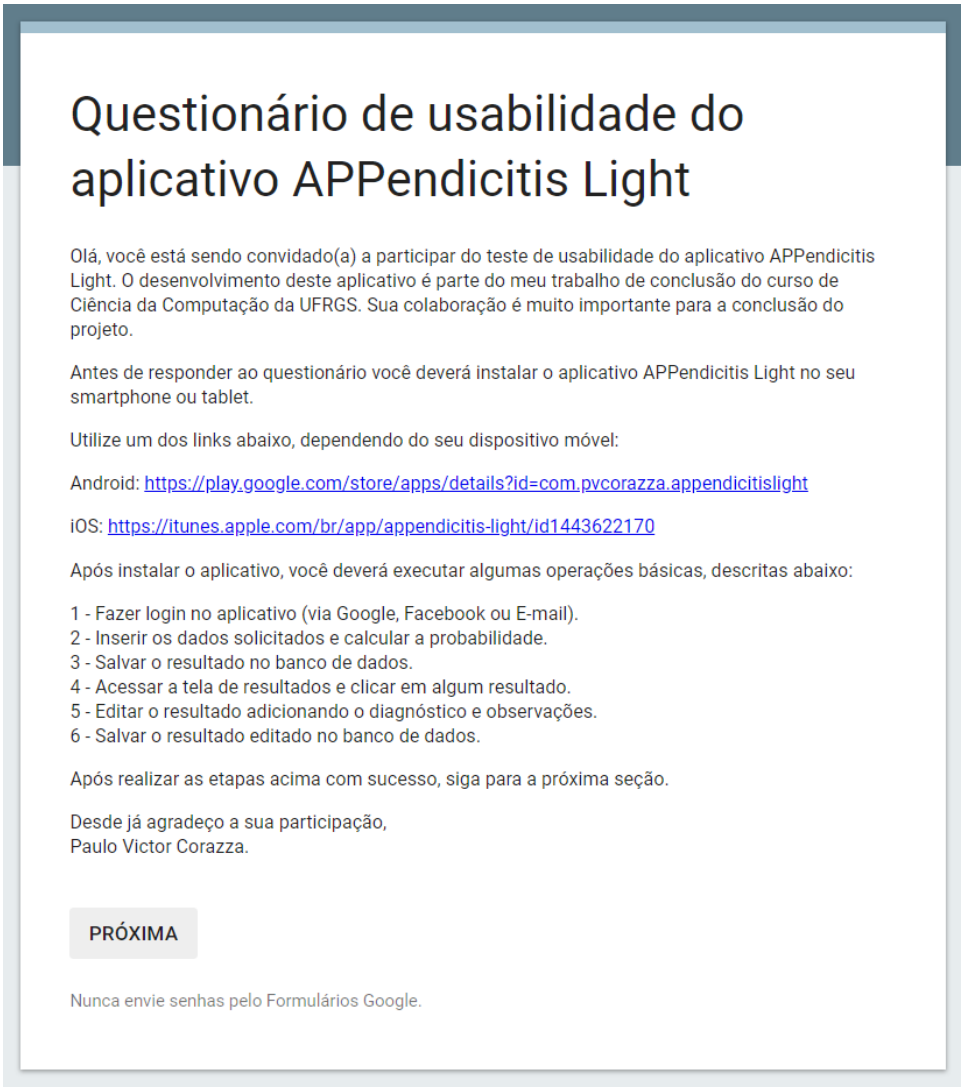
SCHWABER, K. **Agile project management with Scrum**. [S.l.]: Microsoft press, 2004.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. [S.l.]: Pearson, 2015.

VENTOLA, C. L. Mobile devices and apps for health care professionals: uses and benefits. **P T**, v. 39, n. 5, p. 356–364, May 2014.

APÊNDICE A — QUESTIONÁRIO DE AVALIAÇÃO

A.1 Instruções para realização do questionário



Questionário de usabilidade do aplicativo APPendicitis Light

Olá, você está sendo convidado(a) a participar do teste de usabilidade do aplicativo APPendicitis Light. O desenvolvimento deste aplicativo é parte do meu trabalho de conclusão do curso de Ciência da Computação da UFRGS. Sua colaboração é muito importante para a conclusão do projeto.

Antes de responder ao questionário você deverá instalar o aplicativo APPendicitis Light no seu smartphone ou tablet.

Utilize um dos links abaixo, dependendo do seu dispositivo móvel:

Android: https://play.google.com/store/apps/details?id=com.pvcorazza.appendicitislight

iOS: https://itunes.apple.com/br/app/appendicitis-light/id1443622170

Após instalar o aplicativo, você deverá executar algumas operações básicas, descritas abaixo:

1 - Fazer login no aplicativo (via Google, Facebook ou E-mail).
2 - Inserir os dados solicitados e calcular a probabilidade.
3 - Salvar o resultado no banco de dados.
4 - Acessar a tela de resultados e clicar em algum resultado.
5 - Editar o resultado adicionando o diagnóstico e observações.
6 - Salvar o resultado editado no banco de dados.

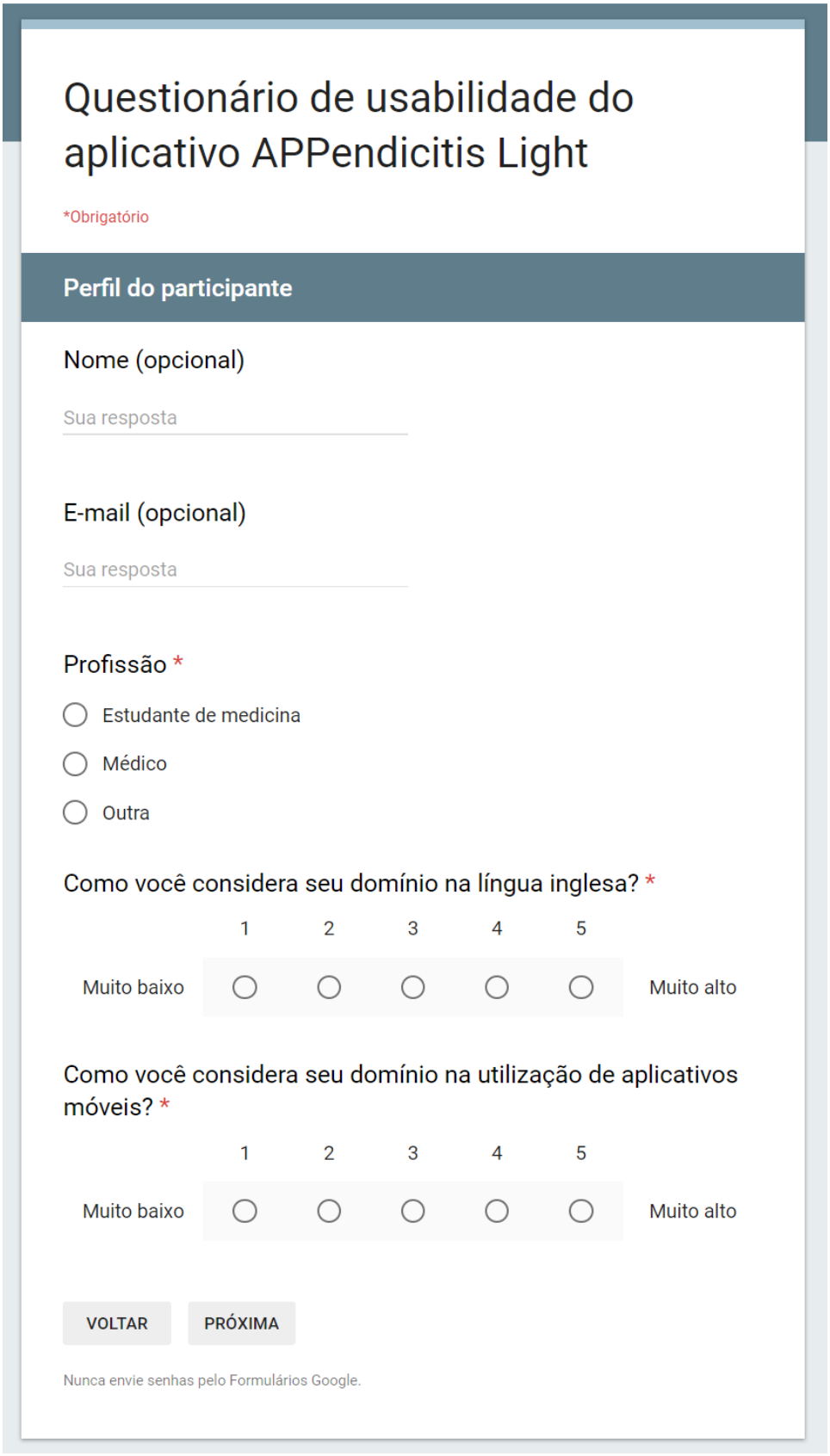
Após realizar as etapas acima com sucesso, siga para a próxima seção.

Desde já agradeço a sua participação,
Paulo Victor Corazza.

PRÓXIMA

Nunca envie senhas pelo Formulários Google.

A.2 Perfil do participante



Questionário de usabilidade do aplicativo APPendicitis Light

*Obrigatório

Perfil do participante

Nome (opcional)

Sua resposta

E-mail (opcional)

Sua resposta

Profissão *

- Estudante de medicina
- Médico
- Outra

Como você considera seu domínio na língua inglesa? *

	1	2	3	4	5	
Muito baixo	◯	◯	◯	◯	◯	Muito alto

Como você considera seu domínio na utilização de aplicativos móveis? *

	1	2	3	4	5	
Muito baixo	◯	◯	◯	◯	◯	Muito alto

VOLTAR PRÓXIMA

Nunca envie senhas pelo Formulários Google.

A.3 Facilidade no uso do aplicativo

Facilidade no uso do aplicativo

Com base nas operações que você realizou no aplicativo, responda de acordo com o nível de dificuldade encontrado em cada operação.

1 - Fazer login no aplicativo (via Google, Facebook ou E-mail) *

- ◯ Muito fácil
- ◯ Fácil
- ◯ Médio
- ◯ Díficil
- ◯ Muito difícil

2 - Inserir os dados solicitados e calcular a probabilidade *

- ◯ Muito fácil
- ◯ Fácil
- ◯ Médio
- ◯ Díficil
- ◯ Muito difícil

3 - Salvar o resultado no banco de dados *

- ◯ Muito fácil
- ◯ Fácil
- ◯ Médio
- ◯ Díficil
- ◯ Muito difícil

4 - Acessar a tela de resultados e clicar em algum resultado *

- ◯ Muito fácil
- ◯ Fácil
- ◯ Médio
- ◯ Díficil
- ◯ Muito difícil

5 - Editar o resultado adicionando o diagnóstico e observações *

- ◯ Muito fácil
- ◯ Fácil
- ◯ Médio
- ◯ Díficil
- ◯ Muito difícil

6 - Salvar o resultado editado no banco de dados *

- ◯ Muito fácil
- ◯ Fácil
- ◯ Médio
- ◯ Díficil
- ◯ Muito difícil

A.4 Questionário System Usability Scale

Questionário System Usability Scale (SUS)

Agora você está convidado a avaliar o aplicativo através de respostas indicando a sua concordância com a afirmação na escala de 1 a 5 através do System Usability Scale (SUS), um questionário de avaliação padronizado que permite gerar uma pontuação para medir a usabilidade do aplicativo.

Eu acho que gostaria de usar este aplicativo com frequência: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei o aplicativo desnecessariamente complexo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei o aplicativo fácil de usar: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu acho que precisaria do apoio de um suporte técnico para conseguir usar este aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei que as várias funções neste aplicativo foram bem integradas: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei que houve muita inconsistência neste aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu imagino que a maioria das pessoas aprenderiam a usar este aplicativo bem rapidamente: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei o aplicativo muito pesado para o uso *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu me senti bem confiante ao usar o aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu precisei aprender muitas coisas para conseguir utilizar este aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente