UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

LUISA DE CICCO TANGZA

APLICATIVO DE CELULAR *ANDROID* PARA AUXILIAR NO MONITORAMENTO DE REMÉDIOS

Sorocaba

2022

LUISA DE CICCO TANGZA

APLICATIVO DE CELULAR *ANDROID* PARA AUXILIAR NO MONITORAMENTO DE REMÉDIOS

Trabalho de graduação apresentado ao Instituto de Ciência e Tecnologia, da Universidade Estadual Paulista "Júlio de Mesquita Filho" Campus Sorocaba, como parte dos requisitos para obtenção do grau de bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Márcio Alexandre Marques

Coorientadora: Profa. Me. Patrícia Moriguchi

Sorocaba

2022

T164a

Tangza, Luisa de Cicco
Aplicativo de celular Android para auxiliar no monitoramento de remédios / Luisa de Cicco Tangza. -- Sorocaba, 2022
58 p. : fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Márcio Alexandre Marques
Coorientadora: Patrícia Moriguchi

1. Android. 2. Aplicativo. 3. Medicamento. 4. Java. 5. Idoso. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Universidade Estadual Paulista "Júlio de Mesquita Filho"
Instituto de Ciência e Tecnologia de Sorocaba
Engenharia de Controle e Automação

FORMULÁRIO DE AVALIAÇÃO DO TRABALHO DE GRADUAÇÃO

Nome do Aluno:	Luisa de Cicco Tangza
Título do Trabalho:	**APLICATIVO DE CELULAR ANDROID PARA AUXILIAR NO MONITORAMENTO DE REMÉDIOS**
Banca	
Orientador (O)	Prof. Dr. Márcio Alexandre Marques
Avaliador (A1):	Prof. Dr. Everson Martins
Avaliador (A2):	Profa. Me. Patrícia Moriguchi (UNIP/Sorocaba)

AVALIAÇÃO

1. Avaliação do Orientador quanto ao Desenvolvimento do Trabalho (Orientador) **(N1)** - Peso 25% da Nota Final (Notas de 0 a 10).

Item		**Nota**
Avaliação do(s) Relatório(s) Parcial(is)	(60%)	10,0
Gerenciamento do TG	(20%)	10,0
Grau de iniciativa do estudante	(20%)	10,0
	Média (N1)	**10,0**

2. Avaliação da Apresentação do Trabalho de Graduação **(N2)** – PESO 15% da Nota Final (Notas de 0 a 10).

Item		**O**	**A1**	**A2**	**Média**
Apresentação oral	(40%)	10,0	10,0	10,0	10,0
Organização do material	(40%)	9,5	9,5	9,5	9,5
Postura do Aluno	(20%)	10,0	10,0	10,0	10,0
				Média	**9,8**

3. Avaliação do **Relatório Final** e **Artigo (N3)** – PESO 60% da Nota Final (Notas de 0 a 10).

Item	**O**	**A1**	**A2**	**Média**
Relatório Final: estrutura do relatório, pesquisa bibliográfica, metodologia, qualidade técnico-científica, redação, etc. (60%)	10,0	10,0	10,0	10,0
Artigo: definição do problema, pesquisa bibliográfica, originalidade da proposta, redação. (40%)	10,0	10,0	10,0	10,0
			Média	**10,0**

4. Parecer: (X) Aprovado sem alterações ( ) Aprovado com alterações ( ) Reprovado

5. Alterações Exigidas:

6. Nota Final	**10,0**

Sorocaba, 08 de março de 2022

Orientador	Avaliador 1	Avaliador 2

Av. Três de Março, 511 ● CEP 18087-180 ● Sorocaba-SP
Phone: ++55 15 3238-3401 Fax: ++55 15 3228-2842
http://www.sorocaba.unesp.br

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por me guiar em minha jornada. Agradeço ao meu pai, que mesmo não estando mais entre nós, sempre fez o possível e o impossível para me apoiar. Agradeço à minha família que sempre me incentivou e ajudou em vários momentos e decisões importantes da minha vida. Agradeço também a todos os amigos e professores que fizeram parte de todo o meu desenvolvimento, que foram determinantes no meu crescimento tanto pessoal quanto profissional. Por fim, agradeço ao Prof. Dr. Márcio Alexandre Marques e a Profa. Me. Patrícia Moriguchi pela orientação e apoio no desenvolvimento deste trabalho.

RESUMO

LUISA, C. T. **Aplicativo de celular *Android* auxiliar no monitoramento de remédios**. Trabalho de Graduação – Instituto de Ciência e Tecnologia do Campus de Sorocaba, Universidade Estadual Paulista "Júlio de Mesquita Filho", Sorocaba, 2022.

Atualmente, o bem estar e o cuidado com a saúde se destacam, principalmente se tratando de medicamentos, pois muitas enfermidades apenas são curadas ou controladas por meio destes, os quais devem ser administrados seguindo a posologia correta e por um período de tempo determinado. Diante disso, este trabalho visa a implementação de um aplicativo para celulares (*Android*) utilizando-se da Metodologia Ágil (*Scrum*), voltado para o público idoso, aplicativo este, que seja capaz de gerenciar os medicamentos que o paciente deve tomar, além de alertá-lo sobre o horário de sua administração através de um alarme no *smartphone*. Quando o alarme é disparado, é apresentado na tela do celular uma fotografia da caixa do medicamento, com o seu nome e a sua posologia. O usuário pode postergar o alarme por até 3 vezes, sendo que, na última vez, o aplicativo considerará que o medicamento não foi administrado. Ao final do ciclo de uso do medicamento, um *e-mail* é enviado para o cuidador ou médico cadastrado no aplicativo, com um relatório dos dias e horários de prescrição do medicamento, informando se o paciente o tomou ou não, e se houve algum atraso na sua utilização. O aplicativo foi programado em *Java*, com sua estrutura e *FrontEnd* desenvolvidos no Android Studio. Alguns mecanismos nativos do próprio *Android* como: banco de dados SQL, *ContentProvider*, *ContentReceivers*, *AlarmManager* e *BroadcastReceivers* foram utilizados. A interface com o usuário foi desenvolvida especialmente voltada para idosos, utilizando-se botões, instruções e fontes com um tamanho maior para ajudar a leitura e a compreensão, bem como a apresentação de uma fotografia da caixa do medicamento para facilitar a sua identificação. Foram realizados testes unitários e com voluntárias para melhorar a implementação do aplicativo com base nas suas necessidades e *feedback* dos testes. Verificou-se que a Metodologia Ágil foi fundamental para o desenvolvimento do aplicativo, uma vez que a cada *sprint,* os problemas foram corrigidos e o aplicativo aprimorado. Sua interface é amigável, intuitiva e de fácil compreensão, o que auxilia os usuários, principalmente idosos, na sua utilização.

Palavras-chave: *Android*, aplicativo, medicamento, *Java*, idoso.

ABSTRACT

LUISA, C. T. **Android smartphone application to help monitor medications**. Undergraduate Work – Sorocaba Campus Institute of Science and Technology, São Paulo State University "Júlio de Mesquita Filho", Sorocaba, 2022.

Nowadays, well-being and health care stand out, especially when it comes to medicines, as many diseases are only cured or controlled through them, which must be administered following the correct dosage and for a determined period of time. In addition, use Methodology, this application for applications, which is practical for the elderly public, an application that is capable of managing the medications that the patient must take, in addition to alerting him about the time of administration through an alarm on the smartphone. When the alarm is triggered, a photo of the medicine box is displayed on the cell phone screen, with its name and dosage. The user can postpone the alarm for up to 3 times, with the application considering that the drug was not administered for the last time. At the end of the use cycle, an email is sent to the caregiver or physician registered in the day report and the application has not programmed the medication medication, informing if the patient has taken it or has taken it, and if there has been any delay in its use. The application was programmed in Java, with its structure and FrontEnd developed in Android Studio. Some native Android mechanisms such as: SQL database, ContentProvider, Content-tReceivers, AlarmManager and BroadcastReceivers were used. The user interface was large as a reading for identification, using instructions and fonts with a larger size to help with comprehension and comprehension, as well as presenting a photograph of the medicine box for its identification. Unit and feature tests were performed to improve application implementation based on your testing needs and feedback. It was found that the Agile Methodology was fundamental for the development of the application, since each sprint, the problems were fixed and the application improved. Its interface is friendly, intuitive and easy to understand, or that helps users, especially the elderly, in its use.

Keywords: Android, application, medicine, Java, elderly.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

O bem estar e a saúde de idosos é sempre uma preocupação para algumas pessoas. Geralmente, eles precisam de maiores cuidados conforme vão envelhecendo, mesmo vivendo um estilo de vida saudável. Por isso, é muito importante uma atenção constante e monitoramento no uso de medicamentos essenciais para que continuem com uma boa qualidade de vida. É primordial que os idosos tomem seus remédios nos horários e dias corretos, respeitando os intervalos entre eles, pois alguns possuem princípios ativos que podem interagir e causar desconfortos ou efeitos colaterais maléficos para o paciente [1].

Com isso, a má administração de medicamentos, como por exemplo, erros na posologia e interações de princípios ativos, não são os únicos problemas que os idosos enfrentam quando seguem algum tipo de tratamento, há também o esquecimento. Neste caso, eles possuem, na maioria das vezes, uma grande dificuldade de se lembrar dos dias e dos horários que um determinado medicamento deve ser tomado, causando atrasos e pouca eficiência da terapia que está sendo seguida [2].

Para resolver este problema, muitos idosos utilizam métodos simples como guardar todos medicamentos em uma caixa e ficar sempre com ela ao seu lado, ou anotar em papéis todas as informações do tratamento para depois colá-los no guarda-roupas ou geladeira. Dessa forma, eles são constantemente lembrados onde quer que estejam em suas casas, e assim serem capazes de seguir o tratamento corretamente [2].

Diferentemente, há outros métodos que dependem da tecnologia e que podem se mostrar muito mais efetivos do que um simples pedaço de papel, isso porque os avanços tecnológicos são constantes e buscam tornar mais fáceis e práticas as tarefas humanas.

Um exemplo disso são os aplicativos de celular, os quais possuem a capacidade de deixar o processo mais prático para o usuário no momento de se organizar em seu cotidiano. Assim, há aplicações que auxiliam mulheres a não esquecerem de tomarem pílula anticoncepcional, que é o caso do aplicativo “Hora da Pílula”, o qual avisa por meio de um toque de alarme que chegou na hora pré-determinada pelo usuário [3].

Neste mesmo sentido, pode-se encontrar no mercado outros que funcionam como alarme para remédios em geral e que utilizam o mesmo princípio, como é o caso do “Hora do Remédio” [4]. Porém, eles não possuem uma forma de informar ao profissional de saúde o andamento do tratamento do paciente e nem de enviar um relatório ao final do tratamento. Além disso, também não há a possibilidade de selecionar qual a posologia desejada e o aplicativo é somente para dispositivo Apple.

Continuando nesta linha de pensamento, criar e aperfeiçoar aplicativos de alarme capazes de auxiliar ainda mais os idosos no gerenciamento dos remédios que precisam ser administrados, mostra-se de grande importância. Isso porque, adicionando-se mais *features*, muitos detalhes poderiam ser averiguados para que o paciente não se confunda ou se perca com tantos tratamentos e remédios [2].

Com isso, percebe-se que para o desenvolvimento de *software* otimizado, é necessária uma metodologia compatível com as necessidades tanto do usuário quanto do próprio desenvolvedor e, no caso, a Metodologia Ágil (Scrum) se mostra muito eficaz para a criação de um aplicativo que satisfaça as duas partes. Essa metodologia pode, então, ser utilizada com o intuito de organizar melhor o desenvolvimento e tornar a aplicação mais amigável e intuitiva para o usuário. Por meio do *framework* pode-se dividir o desenvolvimento do aplicativo por *sprints*. Ao final dos *sprints*, que possuem os desenvolvimentos das *user interfaces,* é possível realizar testes com usuários a fim de coletar um *feedback* da aplicação [5].

A partir dessas informações, o aplicativo pode ser ajustado para melhor atender as necessidades de seus usuários e com isso, tornar a uma ferramenta mais acessível, já que o público alvo são, principalmente, idosos. Por meio de tal aplicação, os médicos poderão ter um melhor acompanhamento de seus pacientes, indicando com mais segurança os tratamentos, visto que podem monitorar se um remédio está fazendo o efeito desejado ou não para aquele paciente.

1.1 OBJETIVO

O objetivo deste Trabalho de Graduação (TG) é, por meio da metodologia Ágil (*Scrum*), desenvolver um aplicativo de celular específico para o sistema operacional *Android*, que permita a um paciente idoso configurar o alarme do *smartphone* com as informações dos remédios que ele precisa tomar, como horários e posologias. O aplicativo visa trazer mais conforto ao paciente, alertando-o sobre o horário dos medicamentos e auxiliando médicos e/ou cuidadores no acompanhamento da administração dos remédios.

1.2 ESTRUTURA

O presente trabalho está estruturado em capítulos, iniciando-se no Capítulo 2, que apresenta uma revisão da literatura acerca dos tópicos que envolvem o protótipo.

No Capítulo 3, têm-se os materiais e métodos utilizados no desenvolvimento da ferramenta.

No Capítulo 4 encontram-se os resultados. Após a aplicação dos conceitos teóricos na parte prática, observa-se os resultados obtidos, ou seja, se a aplicação conseguiu desempenhar o papel de automatizar um processo que antes era manual, transmitir um *feedback* mais rápido e preciso para o usuário e organizar em uma estrutura de banco de dados, as informações cadastradas.

O Capítulo 5 aborda as conclusões acerca do protótipo. No Capítulo 6 apresenta-se sugestões para trabalhos futuros. E, por fim, no Capítulo 7, estão contidas as referências bibliográficas relevantes utilizadas para o desenvolvimento do projeto.

2. REVISÃO BIBLIOGRÁFICA

Primeiramente, para que o funcionamento do projeto seja compreendido, os conceitos teóricos utilizados serão explicitados.

2.1 Medicamentos para idosos

Atualmente a maior fonte de tratamentos de saúde provem de remédios fabricados pelo ser humano e, com o avanço continuo da medicina e de seus estudos, é possível, cada vez mais, melhorar a qualidade de vida dos indivíduos e, principalmente dos idosos. Isso porque, conforme há o envelhecimento, muitos problemas antes inexistentes podem surgir e causar desconfortos para a vivência da pessoa. No caso, a população idosa possui tendência a adquirir problemas de doenças crônicas, como por exemplo diabetes, artrite ou hipertensão e, por isso, um acompanhamento constante é ideal. Segundo pesquisas realizadas pela empresa MSD (empresa farmacêutica americana), percebe-se que por volta de 90% dos idosos consomem até um medicamento continuamente durante a semana. Do mesmo modo, cerca de 80% precisam tomar pelo menos 2 remédios continuamente e, por fim, 36% dos idosos tomam até cinco remédios continuamente [1].

Estas informações proporcionam um melhor entendimento do porquê de cuidados especiais serem tomados quando pessoas em uma faixa etária mais avançada estão envolvidos. Utilizar medicamentos para a melhora da qualidade de vida é imprescindível, porém, devido a quantidade de problemas de saúde que possuem, deve-se tomar as devidas precauções. Ao mesmo tempo que estes medicamentos ajudam a resolver problemas sérios de saúde, a ingestão destes podem trazer efeitos colaterais perigosos. Isso porque nesta faixa etária o corpo não funciona mais com o mesmo metabolismo de uma pessoa jovem e, a eliminação das substancias químicas do corpo pode demorar mais do que seria considerado o normal. Além disso, conforme a idade aumenta, a proporção de água no corpo do indivíduo decresce. Deste modo, a absorção de remédios desfeitos em água é mais complicada e, por consequência, os remédios que dependem da gordura para serem dissipados acabam ficando mais concentrados. Assim, as dosagens devem ser condizentes com as condições do paciente [1].

Com isso, é de suma importância que as dosagens sejam adequadas para o uso por essas pessoas, com o intuito de que medicamentos não sejam misturados e que não haja interação entre as substâncias, causando problemas a mais de saúde.

É essencial que os idosos respeitem os horários de ingestão dos medicamentos prescritos por seus médicos, para que assim consigam se beneficiar efetivamente dos efeitos que o remédio deve trazer, já que medicamentos possuem um intervalo específico para dar

continuidade ao seu efeito [6]. Porém, é comum eles se confundirem quando a quantidade de medicamentos indicados é grande e isso pode ser uma tarefa complexa para qualquer um, principalmente quando efeitos de interação entre substâncias é um acontecimento provável. Por isso quanto mais informações são adicionadas para o tratamento, mais confuso o idoso irá se tornar.

Algo que também pode agravar esta desorientação, são as doenças crônicas e déficit de memória que comprometem a realização de tarefas simples como tomar um remédio. Muitas vezes, lembrar pelo nome ou somente observar o formato do medicamento não é o suficiente, principalmente porque é comum idosos retirarem os remédios de seus recipientes originais e coloca-los em porta comprimidos, confundindo ainda mais suas mentes. Por isso, guardar elementos visuais mais efetivos como as caixas dos remédios é uma importante saída para que eles consigam lidar com suas dificuldades [7].

O conhecimento dos efeitos da interação entre medicamentos e nas pessoas, além de estudos cada vez mais avançados em seu entendimento, trará uma base mais fundamentada e correta para que possa ser possível trabalhar de forma proveitosa e com menos riscos para os pacientes.

2.2 Tecnologias

Nos dias atuais, é quase impossível viver sem ter o contato com novas tecnologias, pois elas estão cada vez mais presentes no dia a dia das pessoas. Por causa disso, são instrumentos que podem contribuir e muito no que diz respeito à saúde da população, tanto na obtenção de conhecimento quanto na sua propagação e constante desenvolvimento [8].

Um aplicativo de celular, por exemplo, está sempre ao alcance do usuário e pode ajudar em muito de acordo com aquilo que ele está programado a realizar [8]. Se seu objetivo for monitorar se a pessoa tomou ou não um remédio, isso pode trazer benefícios tanto para o usuário quanto para o próprio médico que o acompanha, como melhorar comunicações e o compartilhamento de informações relacionadas ao tratamento [9]. Para o público idoso, aplicações relacionadas à saúde são valiosas, já que na maioria das vezes são simples de serem utilizadas quando criadas especialmente para eles, promovendo o autocuidado e a autonomia no cotidiano [10].

2.3 Metodologia Ágil (*Scrum*)

Atualmente, a economia de tempo é um dos principais fatores, tanto para otimizar tarefas quanto para a criação de algo novo. Por isso, o uso do *Scrum* é muito importante e útil,

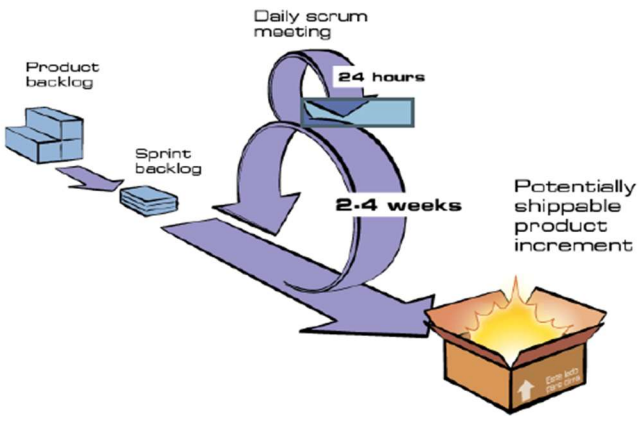
pois esta Metodologia Ágil é uma das mais indicadas para gestão e planejamento de projetos de *software*.

Neste método, para a realização de um projeto de modo mais eficiente, há a divisão em tarefas chamadas de *Sprints*. Elas consistem em um conjunto de atividades nas quais devem ser executadas em um tempo pré-determinado e que, normalmente, variam no tempo de um mês. Essas metodologias de criação de *software* são iterativas, isso quer dizer que há um contato continuo com o interessado pelo produto ou tarefa que estão sendo otimizados. Neste caso, as funcionalidades que devem ser implementadas no projeto, como no caso do aplicativo, as funcionalidades presentes nele são adicionados em uma lista de produto (*Product Backlog*). Antes de se iniciar uma *Sprint*, realiza-se uma reunião (*Sprint Planning Meeting*) para que o dono do produto ou tarefa (*Product Owner*) indique quais os itens que ele deseja que sejam implementados antes, escolhendo dentre as opções existentes dentro do *Product Backlog* [5].

Após a equipe selecionar as atividades que conseguirão executar no determinado prazo, as tarefas alocadas em um *Sprint* são transferidas do *Product Backlog* para o *Sprint Backlog*. Durante toda a duração da *Sprint* especificada, os responsáveis por realiza-la organiza reuniões (*Daily Scrum*) para que as ideias sejam trocadas e os objetivos para cada dia se tornem oficializados, além de analisar o que foi feito anteriormente e identificar se é viável dar continuidade. Toda vez que uma *Sprint* chega ao fim, o grupo responsável por ela apresenta o que foi implementado por meio de uma reunião geral (*Sprint Review Meeting*). Os envolvidos realizam então uma análise da *Sprint* inteira (*Sprint Retrospective*) e, se estiver tudo de acordo com o que é desejado, a equipe pode começar a trabalhar na próxima *Sprint*, iniciando o ciclo novamente [5].

A Figura 1 mostra, por meio de uma ilustração, como este processo funciona.

Figura 1 – Diagrama explicativo sobre *Scrum.*

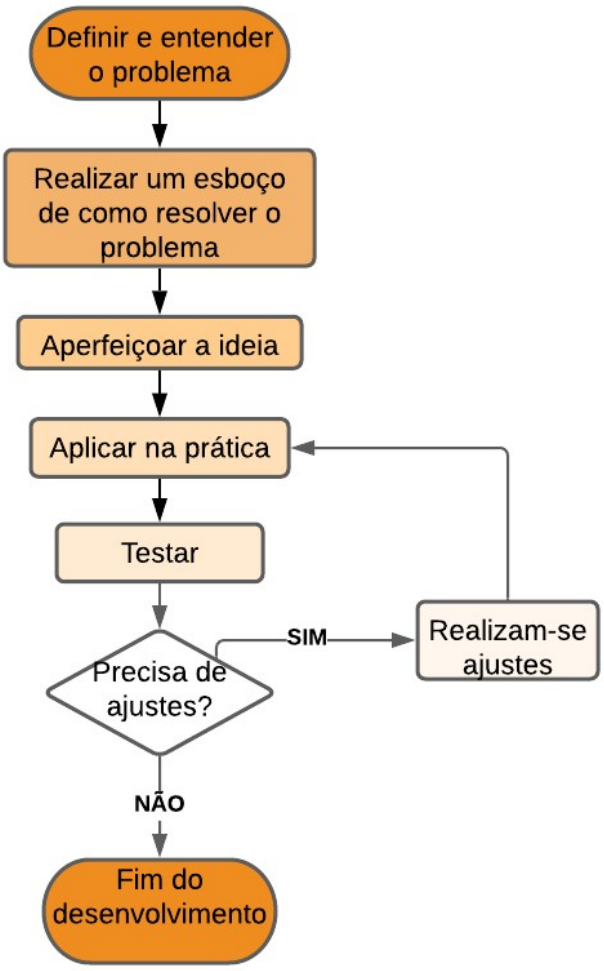


Fonte: [5]

2.4 Desenvolvimento de aplicativos

Para desenvolver um aplicativo é necessário, primeiramente, uma plataforma com a qual seja possível o desenvolvimento do código por trás da aplicação e que permita a visualização do que se está criando. Com isso, é necessário criar a ideia e, após a verificação de que ela é viável, desenhar um esquema de blocos para se ter uma noção de como será a interface e o fluxo de informações do aplicativo e seus recursos. Além disso, um roteiro (*storyboard*) é importante para que as conexões dentro do aplicativo fiquem explicitas. Depois do planejamento ser mapeado, é necessário discutir sobre o *design* e sobre a programação por trás da aplicação (*BackEnd*), ou seja, aplicar as ideias do diagrama de bloco em um aspecto visual e funcional. Assim, como última etapa vem o teste do protótipo do produto final, para se ter a certeza de que ele está realizando aquilo que foi criado para executar [11]. Um exemplo pode ser encontrado na Figura 2.

Figura 2 – Desenvolvimento de um aplicativo.



Fonte: Autoria própria.

2.5 *Android Studio*

Este *software* é uma plataforma que permite o desenvolvimento integrado (IDE) para desenvolver aplicações para o sistema *Android*. Neste ambiente é possível monitorar as alterações visuais realizadas no aplicativo por meio da programação em tempo real, já que este software possui um emulador de *Android*, o qual possibilita testes antes mesmo da aplicação chegar aos aparelhos reais. Há também a possibilidade de teste em diferentes tipos de dispositivos e versões dos sistemas operacionais presentes nos aparelhos [12]. Este emulador pode ser observado na Figura 3.

Figura 3 – Interface do Emulador do *Android Studio*.

Fonte: Autoria Própria

Quando um novo projeto é iniciado sua estrutura terá todos os arquivos dentro do diretório *Software Development Kit* (SDK), o que proporciona uma maior gama de configurações possíveis no desenvolvimento do aplicativo, permitindo que empresas possam focar mais na criação do produto, sem a necessidade de criar uma ferramenta ou função do zero. Com isso, o SDK é um conjunto de recursos que permite a elaboração de novos aplicativos e melhorias dos que já existem [13].

2.6 *Java*

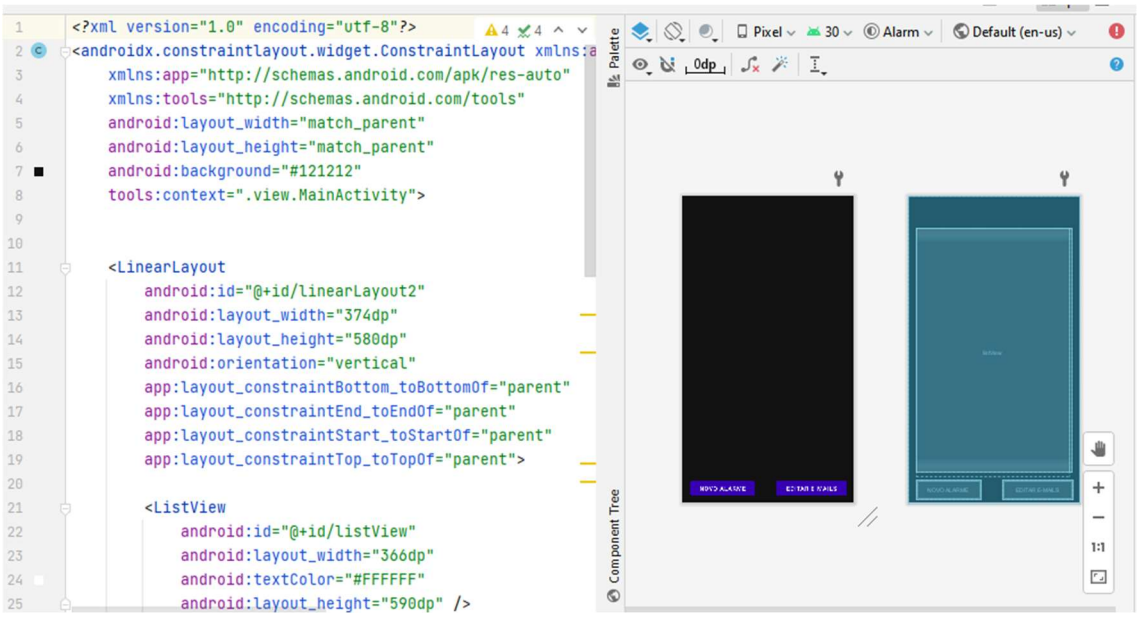
Para o desenvolvimento da aplicação será utilizado a linguagem de programação *Java* e ela foi aplicada no *software Android Studio* com o intuito de se criar a interface do usuário e a programação para o desenvolvimento da base do produto desejado.

A programação em *Java* tem como sua base de criação a Linguagem de Programação em C e é uma das mais utilizadas para criação de programas e aplicativos, já que é muito eficaz e prática. Ela funciona basicamente com um compilador o qual gera *bytecode* (códigos fonte intermediários) e são executados por uma Máquina Virtual, sendo esta, um programa que interpreta os *bytecodes* e logo efetua a compilação [14].

2.7 *FrontEnd*

O *FrontEnd* consiste em arquivos *xml* diretamente sintetizados pelo *Android Studio*, pois como o *software* possibilita a criação facilitada dos arquivos, sua interface permite selecionar *activities* com a estrutura pronta, bastando apenas adicionar os componentes, limitações e *layout* desejados. Pode-se sintetizar os componentes por meio de códigos ou utilizar os componentes já existentes e apenas adicioná-los à composição, conforme ilustra a Figura 4 [15].

Figura 4 - Interface do *Android Studio*, selecionado o arquivo *xml*.

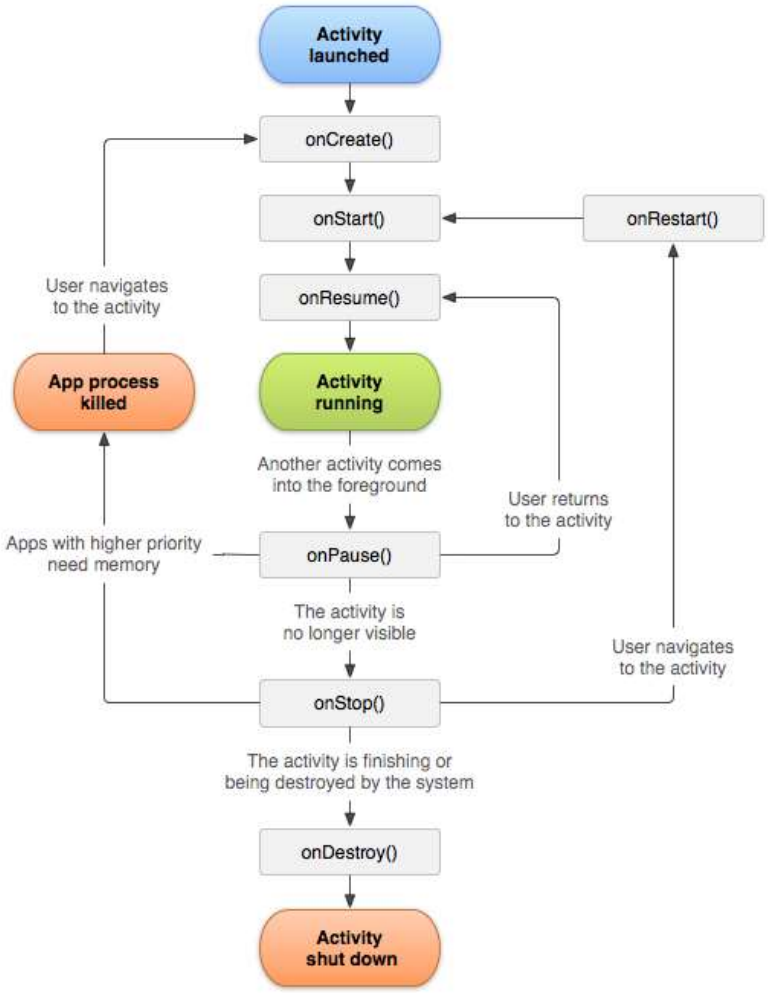


Fonte: Autoria Própria

2.7.1 *Activity*

As partes gráficas do aplicativo *Android* são chamadas de *Activies* e são responsáveis por interagir com o usuário. Cada *activity* tem o seu chamado *lifecycle,* ou seja, as suas fases desde seu início até o seu fim. Quando uma *activity* é iniciada, ela passa pelos métodos *onCreate*, *onStart* e *onResume* até permanecer no estado de ativa e rodando. A partir disso ela pode ser pausada, fechada ou reiniciada, dependendo das operações que são realizadas enquanto a *activity* estiver rodando [16]. O *lifecycle* das *activities* está ilustrado na Figura 5.

Figura 5 - Ciclo de vida de uma *Activity*.



Fonte: [16]

No caso, deve-se programar determinadas funcionalidades de acordo com o *lifecycle* da *activity*, por exemplo, no seu método *onCreate*, quando ela é iniciada, é realizado a inicialização de variáveis, buscas no banco de dados e inicialização de componentes. Para utilizar os métodos do *lifecycle* da *activity,* deve-se utilizar a anotação de "*Override*" em cima dos métodos para que o aplicativo entenda que este será o método escolhido [16].

2.7.2 *Button*

Para criar botões nas *activities* pode-se utilizar o componente *Button*, como mostrado na Figura 6. Para cada botão, deve-se registrar um *listener* associado ao componente para que dessa forma, o aplicativo possa reconhecer quando o usuário aperta o botão [17].

Figura 6 - Componente *Button*.

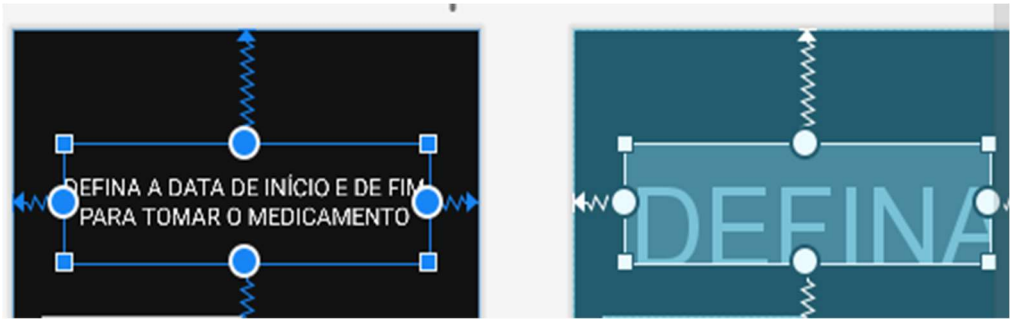
Fonte: Autoria Própria.

O *listener* do botão deve ser configurado na classe que controla a *activity*, assim como o método que será chamado ao se apertar o botão [16].

2.7.3 *TextView*

Pode-se utilizar *TextViews* para informar as instruções ao usuário de como operar a ferramenta e está exemplificado na Figura 7. As *TextViews* podem ser configuradas para adotarem uma fonte grande e clara, assim como adotar cores e efeitos conforme a necessidade para criar uma experiência para o usuário mais amigável [18].

Figura 7 - Componente *TextView*.



Fonte: Autoria Própria.

2.7.4 *TimePicker*

Componentes como os *TimePickers* são utilizados para apresentar ao usuário uma interface gráfica para que eles possam selecionar um horário desejado. É possível configurar o componente para adotar duas formas distintas: o *spinner* ou o em forma de relógio. Para configurar, deve-se definir seus parâmetros durante a sua inicialização e dentro da classe da *activity* que ele irá aparecer. A Figura 8 demonstra o *TimePicker* configurado no modo s*pinner*, pois apresenta uma interface mais amigável [19].

Figura 8 - Componente *TimePicker* em *Spinner mode.*

Fonte: Autoria Própria.

2.7.5 *Calendar*

O componente *Calendar,* observado na Figura 9*,* pode ser utilizado para mostrar ao usuário uma interface gráfica para seleção de uma data específica. Ao se escolher a data e clicar em *OK*, pode-se obter o valor selecionado na classe que o controla [20].

Figura 9 - Componente *Calendar.*

Fonte: [21].

2.8 *BackEnd*

Para cada uma das *activities* é criada uma classe em Java responsável pelo controle de seus componentes e também por implementar a lógica das operações referentes à tela. Pode-se definir qual classe é responsável pelo controle da *Activity* pelo próprio arquivo *xml*, conforme ilustra a Figura 10 [16].

Figura 10 - Atribuição do *Controller* à *Activity*.

Fonte: Autoria Própria.

No caso, observa-se que a classe *MainActivity* foi declarada dentro da *package view* para controlar a *activity_main.xml*. Dessa forma, os componentes presentes na *activity* são declarados na classe controladora, bem como suas lógicas operacionais e propriedades.

2.8.1 *Controller*

Para cada *activity* tem-se uma classe de *controller* com o objetivo de controlar seus componentes, assim como gerenciar as informações inseridas pelo usuário e tratá-las na classe de controle [22]. As ações e funções levam em consideração o *lifecyle* da *activity*, sendo que muitas operações de inicialização são realizadas na sua criação, assim como inserir as devidas permissões para a *activity* [16].

2.8.2 Objeto

Na programação orientada a objetos, para realizar o pré-armazenamento das informações e manipulá-las ao longo da aplicação, pode-se utilizar objetos. Conforme o usuário insere as informações e avança nas telas, as características do objeto são criadas pelos métodos *set*. Em seguida, tais informações são buscadas utilizando os métodos *get* para realizar os devidos tratamentos [23].

A classe do objeto é composta por métodos públicos, no caso os *get* e *set.* Esses métodos são responsáveis por armazenar em uma variável um valor e também por enviá-los para outras variáveis. Pode-se utilizar verificações nos métodos *set* para verificar se o aplicativo está tentando inserir um valor indesejável ao objeto [24].

2.8.3 Banco de dados *SQL*

Utilizando o banco de dados *SQLite* presente nos sistemas operacionais *Android*, pode-se armazenar facilmente dados relevantes para aplicação e acessá-los por meio de *queries.* Estes são comandos na linguagem de *SQL* com o objetivo de buscar informações nos bancos de dados [25].

Para se utilizar um banco de dados *SQL*, deve-se criar classes para realizar a criação das tabelas e sua manipulação. Cada uma das classes possui a mesma estrutura de métodos, estendendo a classe *SQLiteOpenHelper,* responsável por auxiliar na criação das tabelas. Utilizando tal método, é possível um gerenciamento no banco de dados de forma rápida e fácil, possuindo métodos para a criação de tabelas, a adição e remoção de linhas específicas ou atualização de campos. As operações devem ser feitas por meio de *queries* em *SQL* passadas como *Strings* para os métodos [26].

2.8.4 *Intent*

Intent são objetos de mensagens utilizados para realizar atividades entre ac*tivities*. Com eles é possível iniciar *activities*, transmitir informações entre elas e iniciar serviços. Existem dois tipos de *intent*, os explícitos e os implícitos. No primeiro caso, os *intents* são criados e é configurado o aplicativo ou classe que receberá ou realizará a tarefa a ser desempenhada. No caso dos implícitos, ele é apenas configurado, mas sem nomear uma classe ou aplicativo que deverá executar o processo, deixando livre para outras aplicações ou componentes processarem [27].

Para a funcionalidade de transmitir informações entre as *activities*, basta inicializar e declarar a variável do tipo *Intent,* e então, adicionar o valor que se deseja passar entre as *activies* com o método *putExtra*, passando o valor desejado com uma *tag* associada ao valor. Por exemplo, pode-se colocar uma *tag* como "Nome" junto do valor da variável a ser transmitido. Pode-se então, resgatar o valor utilizando os métodos *getIntent* e *getStringExtra* passando como parâmetro a *tag* associada ao valor desejado. No caso, para obter o valor que foi transmitido, seria realizado o comando "*getIntent().getStringExtra("Nome")*".

Além disso, pode-se utilizar o *Intent*, para alterar as *activities* da aplicação, passando qual a *activity* atual e qual a *activity* que se deseja alterar ao se inicializar o objeto *Intent* com um comando *new*. Após a inicialização, utiliza-se o comando *startActivity* passando como parâmetro o *intent* criado e então, finaliza-se com o comando *finish( )* [16].

2.8.5 *Alarm Manager*

O *AlarmManager* é uma classe que permite acesso ao sistema de alarme do dispositivo, permitindo que seja agendado uma outra classe ou método para ser rodado em um determinado horário. Desse modo, pode-se configurar o *AlarmManager* para realizar tarefas ou ativar determinadas *activities* no horário desejado. Para utilizar o *AlarmManager* é criado um *Intent* para ser enviado, para que tais informações sejam recebidas pela *activity*, informando que o alarme será disparado. Dessa forma, deve-se criar o *intent* e utilizar o comando *putExtra* para transmitir as informações. O *intent* é então passado como parâmetro para o *AlarmManager* e define-se a data e hora que se deseja disparar a *activity*. Ao utilizar o método "*setRepeating( )*", pode-se configurar o parâmetro de repetição, ou seja, o *AlarmManager* acionará a *activity* novamente no período de tempo configurado em milissegundos. Pode-se também utilizar o método "*setExact( )*", para obter valores mais exatos no horário de acionamento da *activity* [28].

Já na classe que foi programado para ser acionada pelo *AlarmManager*, deve-se utilizar os comandos de "*getStringExtra*" com o devido valor da *tag* para obter os valores que foram transmitidos [28].

2.8.6 *Android Manifest*

Um aplicativo *Android* é composto de componentes como A*ctivity*, B*roadcastReceivers*, S*ervices* e *ContentProviders* que são de grande importância para o funcionamento do aplicativo. Dessa forma, todo aplicativo possui um arquivo chamado *AndroidManifest.xml*. Tal arquivo serve para registrar os relacionamentos entre os componentes. Nele pode-se definir qual a primeira *activity* que iniciará a aplicação, assim como configurar quais classes são os *BroadcastReceivers* e também quais permissões do *Android* o aplicativo terá. Pode-se observar na Figura 11, um exemplo de um *AndroidManifest.xml.* No caso, observa-se que o aplicativo de exemplo apresenta apenas uma *activity*, sendo ela a "*Activity1*" [16].

Figura 11 - Exemplo do arquivo *AndroidManifest.xml.*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.msi.manning.unlockingandroid">
    <application android:icon="@drawable/icon">
       <activity android:name=".Activity1" android:label="@string/app_name">
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
    </application>
</manifest>
```

Fonte: [16]

2.8.7 *Broadcast Receiver*

O *AlarmManager* quando disparado, ou seja, quando chegar a data e o horário que foram configurados irá alertar um receptor do aplicativo, no caso um *BroadcastReceiver*. Os aplicativos *Android* podem receber transmissões do sistema por meio do *BroadcastReceiver* que são declarados a partir do arquivo de manifesto do aplicativo, o *AndroidManfest.xml* [29].

Para utilizar os *BroadcastReceivers*, deve-se criar uma classe que estenda a classe do *BroadcastReceiver.* Tal classe será responsável por receber as informações passadas pelo *AlarmManager*, que anteriormente foram enviadas via *intent.* Utiliza-se o método *onReceive*() do *BroadcastReceiver* para obter as informações passadas por *intent*, vindas do *AlarmManager.* A partir da recepção das informações pelo método, pode-se utilizá-los conforme as requisições do projeto, passando-os para outras classes ou realizando operações no próprio método [16].

2.8.8 Envio de *e-mail*

Para realizar a atividade de enviar um *e-mail* utilizando o próprio aplicativo, pode-se utilizar um *intent* com a configuração pré preparada para enviar. Deve-se primeiro criar um *intent* com a *action* de "*Intent.ACTION_SEND*", sendo este configurado ao criar o objeto. Em seguida, basta inserir no objeto os *e-mail*s a serem enviados, o corpo e o assunto, passando por cada parâmetro. Tal operação fará com que uma tela apareça para o usuário para selecionar qual aplicativo ele deseja utilizar para enviá-lo e, assim, o *intent* criará o "corpo" do *e-mail*, assim como preencherá os endereços e assunto automaticamente. Esta ação de escolher a aplicação de envio fica a critério do usuário, uma vez que é mais seguro enviar o *e-mail* por meio do próprio aplicativo do dispositivo que gerencia e protege as contas [30].

2.8.9 Permissões

Para que o aplicativo funcione de forma correta, pode-se configurar determinadas permissões para que determinadas funções sejam facilmente acessadas. Por exemplo, pode-se declarar *flags* em uma *activity* para que a tela possa ligar o celular, caso ele esteja em *stand by* e também aparecer sobre a tela de bloqueio do aparelho. Já outras permissões devem ser primeiramente solicitadas ao usuário. Por exemplo, se o aplicativo precisar acessar a câmera para tirar uma foto, deve-se solicitar acesso à câmera. Assim que a permissão seja dada pelo usuário, ela ficará salva no aplicativo e a mesma não será requisitada novamente [31].

Além disso, pode-se declarar determinadas permissões do aplicativo pelo próprio manifesto (arquivo *AndroidManifest.xml*). Tais permissões já são concedidas ao aplicativo caso

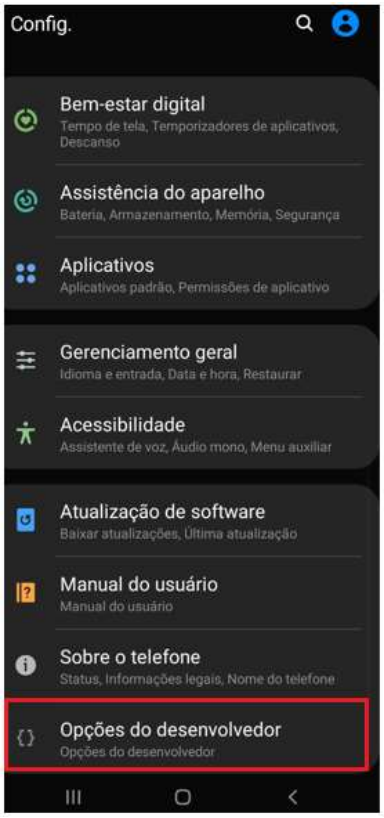
o dispositivo esteja na versão correta. As permissões devem estar escritas no arquivo dentro da tag "*user-permission*" com seu respectivo nome, como por exemplo, "*Android:name=”Android.permision.READ_EXTERNAL_STORAGE”*", que garante ao aplicativo a permissão de ler arquivos no armazenamento externo do dispositivo [31].

2.9 *Developer Mode*

Para que se possa instalar aplicativos utilizando o *Android Studio*, deve-se configurar o aparelho *Android* no modo "*Developer*". Este modo permite que aplicações sejam instaladas no *smartphone* por meio de um cabo USB a partir do *Android Studio* [32].

Na Figura 12, pode-se observar a opção desbloqueada no menu de configurações do aparelho.

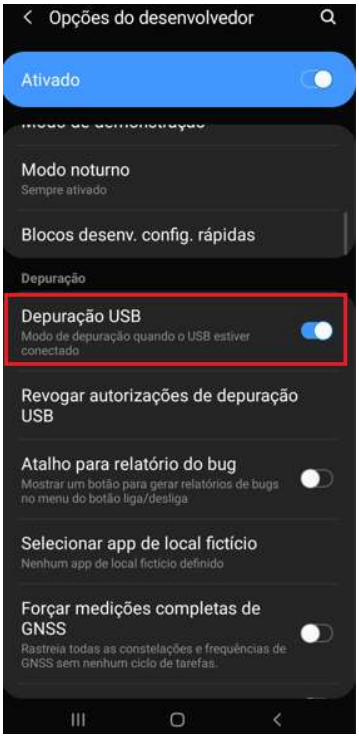
Figura 12 – Modo Desenvolvedor.



Fonte: Autoria Própria.

Com a opção habilitada, permite-se que aplicativos sejam instalados no aparelho por meio de um cabo USB. Isso é possível ao selecionar a opção "Depuração USB" no menu de "Opções do desenvolvedor" desbloqueada ao deixar o dispositivo no modo desenvolvedor [32]. A Figura 13 ilustra a opção de depuração USB.

Figura 13 – Opção de depuração USB.



Fonte: Autoria Própria.

3. MATERIAIS E MÉTODOS

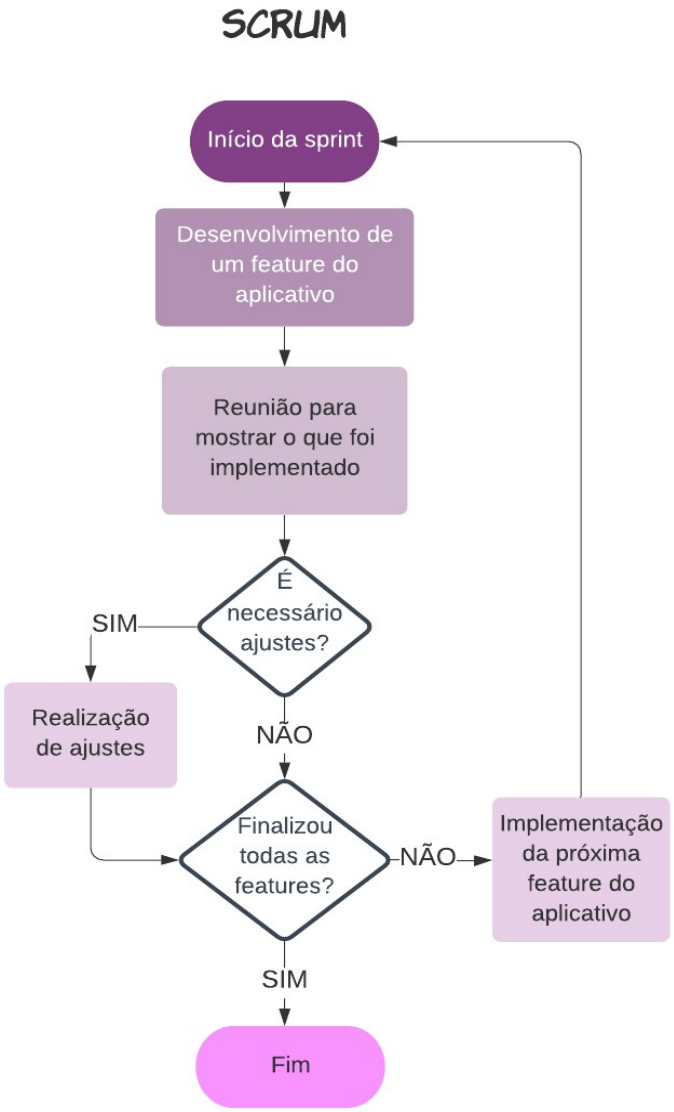
Ao longo deste trabalho foram evidenciados as etapas e os conceitos que permearam o desenvolvimento do projeto. Dessa forma, para o desenvolvimento da aplicação foram utilizados o software *Android Studio*, responsável por compilar a programação em *Java*, assim como gerar o *apk* (aplicativo) executável. Além disso, utilizou-se dispositivos *Android* no modo *developer* para realizar os testes do aplicativo. Durante o desenvolvimento foram empregados mecanismos nativos do sistema operacional *Android*, os quais são gerenciados pelo mesmo. Dentre estes mecanismos estão o banco de dados *SQL* nativo para aplicativos *Android*, com a finalidade de armazenar informações e dados, assim como os *providers* e *receivers* do sistema operacional.

As informações cadastradas nos bancos de dados são referentes às informações inseridas pelo usuário nos alarmes, o que no caso seriam o nome do medicamento a ser administrado, os horários, o período, o endereço que está salvo a imagem do medicamento no dispositivo, sua posologia e o nome do paciente que está tomando. Ademais, há tabelas específicas para armazenar informações referentes ao horário dos alarmes e qual resposta o usuário deu quando foram acionados, como por exemplo, se determinado medicamento foi tomado ou não, e, se sim, se houve algum atraso quando ele foi tomado. Tais informações são separadas pelos dias, horários e código do medicamento, possibilitando a consulta dessas informações quando o aplicativo cria o relatório que será enviado por *e-mail*.

3.1 Utilização da Metodologia Ágil (*Scrum)*

Para promover um melhor desempenho, corrigir *bugs* e melhorar a interface e experiência do usuário, utilizou-se a Metodologia Ágil *Scrum* para desenvolver o aplicativo. O objetivo foi primeiramente empregar as funcionalidades básicas da aplicação. Nas *sprints* seguintes, buscou-se solucionar os eventuais problemas que seriam apontados nos testes e ajustar a aplicação conforme as orientações dos usuários que testaram e dos professores orientadores. Seguindo a ideia desta metodologia, é possível observa-la no fluxograma da Figura 14.

Figura 14 – Fluxograma de como o *Scrum* será aplicado no projeto.



Fonte: Autoria Própria.

Com base no fluxograma da Figura 14, observa-se que as necessidades de ajustes e criação de novas *features* vêm do *feedback* dos professores orientadores e dos testes. Sendo assim, ao término das reuniões com os professores, foram listados quais ajustes deveriam ser realizados na aplicação e, assim que estes eram finalizados, outra reunião era marcada para uma nova discussão sobre quais seriam os próximos passos.

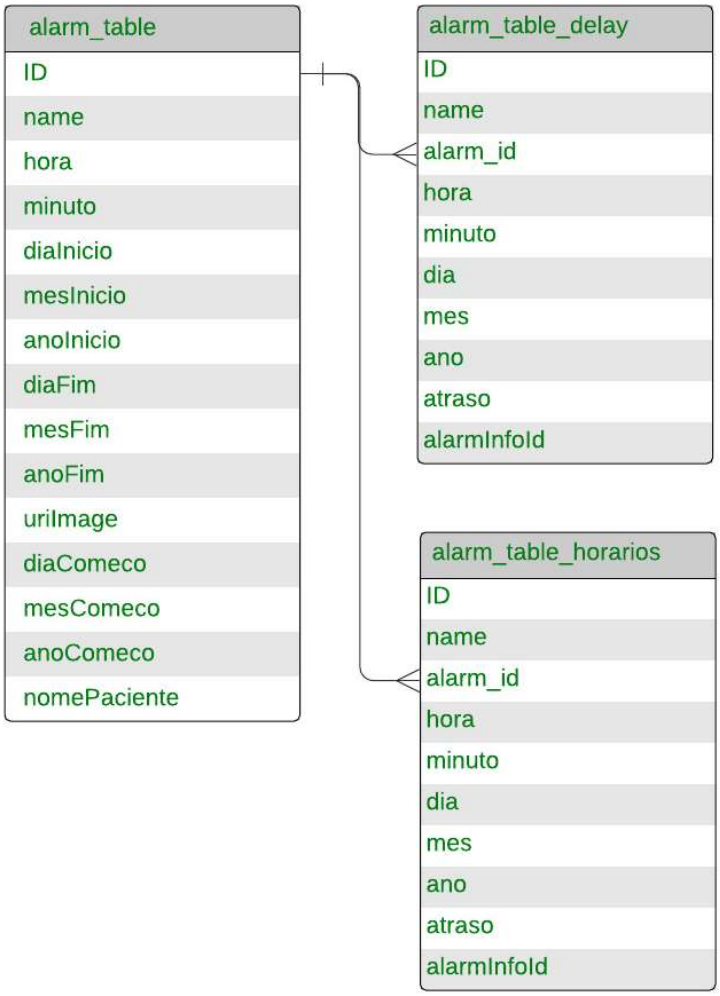
Para os testes, o mesmo esquema foi aplicado, porém, ao invés das reuniões, eles eram cessados caso ocorresse o encontro de algum erro ou *bug* na aplicação. Após o emprego dos ajustes e correções, os testes eram continuados com a versão corrigida do aplicativo. Foram realizados testes unitários conforme as *features* eram implementadas, e depois testes práticos com duas voluntárias idosas. Uma delas utiliza medicamentos diariamente e a outra administrou

um remédio por um período curto de tempo. Ao todo foram realizadas seis *sprints*, aperfeiçoando-se o aplicativo em cada etapa e cada vez mais.

3.2 Banco de Dados *SQL* do Aplicativo

A estrutura do banco de dados foi criada de forma a poder relacionar as tabelas de informações dos alarmes com as de atraso e horários, possibilitando uma melhor organização e confiabilidade das informações a serem armazenadas e buscadas. Conforme ilustra a Figura 15, tem-se a estrutura das tabelas referente aos medicamentos.

Figura 15 - Tabelas dos alarmes.



Fonte: Autoria Própria.

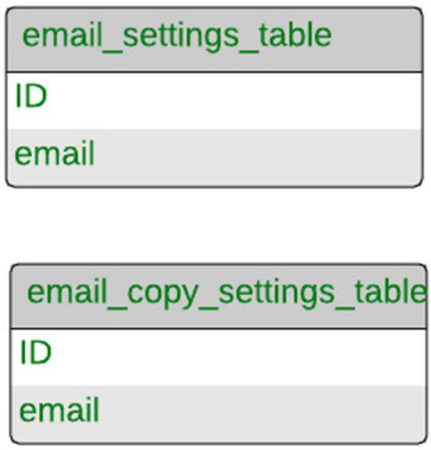
Na Figura 15, observa-se que a tabela principal é a *alarm_table*, onde as informações principais dos medicamentos e alarme são inseridas. Tem-se que o campo *alarm_id* das demais tabelas corresponde ao *ID* do medicamento da tabela de alarmes principal, sendo assim possível

obter as informações referentes à um medicamento em específico por meio de seu *ID* da tabela *alarm_table.*

A tabela *alarm_table_horarios* é referente aos horários que um único medicamento pode ter, ou seja, para os casos em que um medicamento deve ser tomado várias vezes ao longo do dia. Já a tabela *alarm_table_delay* é referente ao atraso que o usuário teve para confirmar a tomada do medicamento, se foi no horário correto, se houve algum atraso ou se ele não tomou depois das tentativas máximas, sendo tais informações registradas nessa tabela.

A Figura 16 ilustra as tabelas responsáveis por armazenar os endereços de *e-mail* que devem ser enviados contendo os relatórios dos medicamentos tomados. Elas são preenchidas quando o usuário salva os *e-mails* na tela de configuração de *e-mails*.

Figura 16 - Tabelas para armazenar os *e-mails*.



Fonte: Autoria Própria.

3.3 Interface e Funcionalidades

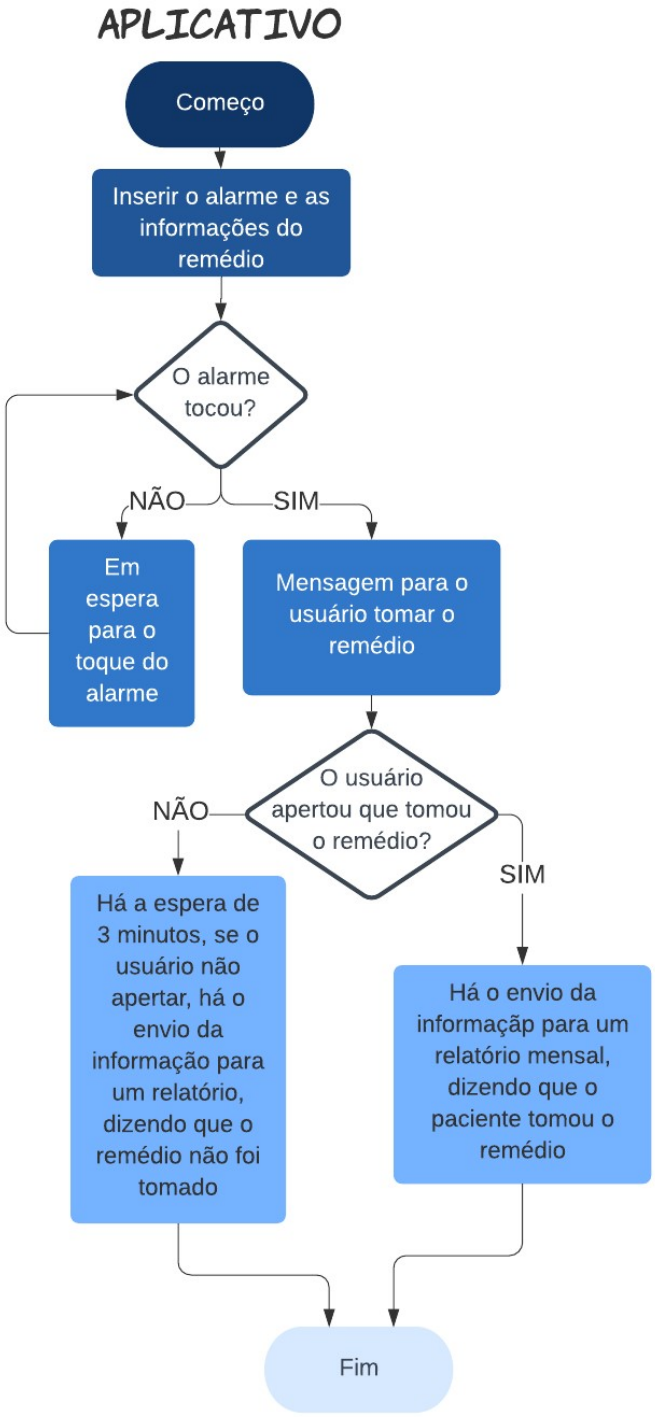
O aplicativo criado tem como ideia inicial ajudar idosos a não esquecerem de tomar seus medicamentos no horário determinado pelo médico. Com isso, a aplicação consiste em definir uma data e horário de quando o alarme irá tocar, além de permitir que uma foto da caixa do medicamento seja tirada para que o usuário não confunda qual remédio deve administrar. Após o toque do alarme no momento definido pelo paciente, esta precisa informar ao aplicativo que a tarefa foi cumprida, selecionando o botão de que o medicamento foi administrado. Se o paciente clicar em “Não tomei”, então o alarme será reagendado para depois de 3 minutos. Caso o usuário selecione que não tomou o medicamento após três tentativas, o aplicativo irá computar que não houve administração.

O aplicativo conta com um banco de dados onde são armazenadas as informações de atrasos para tomar o medicamento. Se o paciente selecionar que tomou o medicamento no

horário certo, sem atrasos, a informação é registrada no banco de dados, para em seguida ser enviada no relatório para o profissional da saúde responsável. Esse relatório é encaminhado todo fim de mês ou no final do tratamento. O processo foi feito para que o médico possa analisar como anda o tratamento e se os remédios estão sendo efetivos.

O fluxograma ilustrado na Figura 17, demonstra o funcionamento do aplicativo.

Figura 17 - Fluxograma de como será a estrutura do aplicativo.



Fonte: Autoria Própria.

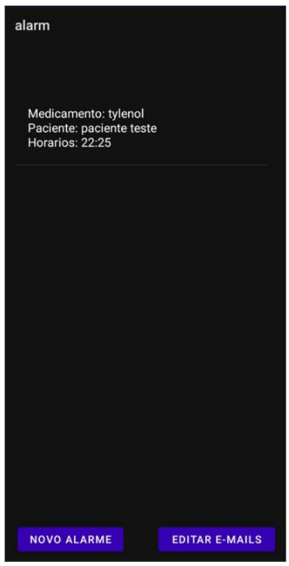
A Figura 18 ilustra a primeira tela do aplicativo onde são realizadas operações no *BackEnd* do aplicativo para buscar informações nas tabelas do mesmo e preparar a próxima tela.

Figura 18 – Tela *splash* ao iniciar o aplicativo.

Fonte: Autoria Própria.

A próxima tela, ilustrada na Figura 19, corresponde à tela principal onde estão localizados os botões que permitem a navegação no aplicativo. Nela, são listados todos os alarmes cadastrados havendo a possibilidade de editá-los ou apaga-los quando selecionados. Há o botão para criar um novo alarme e o botão para editar os *e-mails* que estão cadastrados.

Figura 19 – Tela principal do aplicativo.



Fonte: Autoria Própria.

Ao clicar no botão “Editar *e-mails*” o aplicativo direciona o usuário até a tela de editar os endereços de *e-mails*. Essa tela também é acessada na primeira vez que o usuário entra no aplicativo e ainda não estão cadastrados *e-mails* para serem enviados. A Figura 20 ilustra a tela de cadastro dos *e-mails*. Eles serão armazenados nas respectivas tabelas quando o botão de salvar é pressionado, e em seguida, a “tela inicial” com todos os alarmes já criados é mostrada novamente.

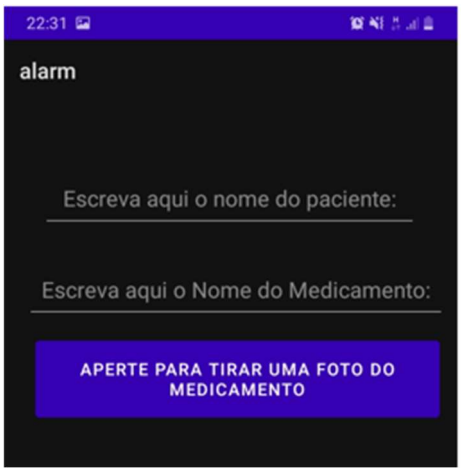
Figura 20 – Tela de cadastro de *e-mail*.

Fonte: Autoria Própria.

Ao clicar no botão “Novo Alarme”, o usuário será redirecionado à tela de cadastro de um novo alarme. A Figura 21 ilustra a tela de cadastro do nome do paciente e o nome do

medicamento a ser administrado. Pode-se clicar no botão para tirar uma foto que será salva no aplicativo e utilizada na hora que o alarme for disparado.

Figura 21 - Cadastro de nome do paciente, medicamento e foto.



Fonte: Autoria Própria

Ao selecionar a foto, a mesma será mostrada para o usuário. É possível alterá-la se outra foto for tirada, conforme ilustra a Figura 22.

Figura 22 - Cadastro do nome do paciente, medicamento e foto preenchidos.

Fonte: Autoria Própria.

O botão “Continuar” direciona o usuário para a próxima tela e o de “Voltar” para a tela anterior. Caso seja selecionado o botão “Continuar”, o próximo passo será realizar o cadastrado da dosagem do medicamento, para que o aplicativo possa informar ao usuário no momento do toque do alarme. A Figura 23 ilustra a tela desse evento.

Figura 23 – Cadastro da dosagem do medicamento.

Fonte: Autoria Própria

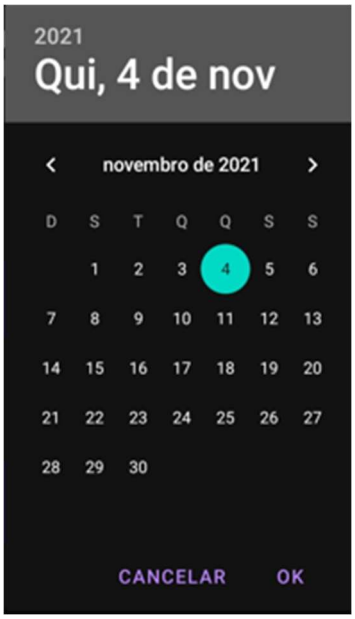
É possível cadastrar comprimidos (“comprimido(s)”), gotas (“mL”), cápsulas (“cápsulas”), drágeas (“Drágea(s)”) e gotas (“Gota(s)”) e um campo de texto que aceita apenas números. A próxima tela, ilustrada na Figura 24, corresponde ao cadastro da data de início e fim que o medicamento deve ser administrado. Os alarmes são gravados a cada dia e a verificação da data final é feita toda vez que o alarme for disparado. Caso a data final não tenha chegado, então o alarme é marcado para o mesmo horário no dia seguinte. Na situação da data final ter chegado, o alarme não é mais reagendado.

Figura 24 – Cadastro do dia de início e fim do alarme.

Fonte: Autoria Própria.

Ao clicar nos botões para selecionar a data é aberto um componente do tipo calendário e nele deve ser selecionado qual a data desejada (Figura 25).

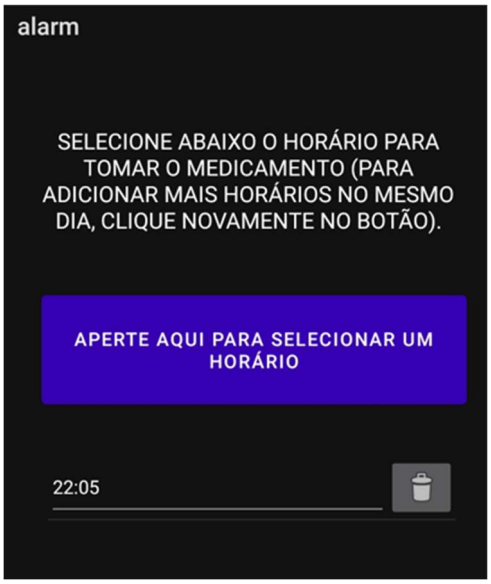
Figura 25 – Calendário para seleção da data.



Fonte: Autoria Própria.

Por fim, deve-se cadastrar os horários do dia que o alarme será disparado. Pode-se selecioná-los ao clicar no botão e selecionar a hora no componente, conforme ilustra a Figura 26.

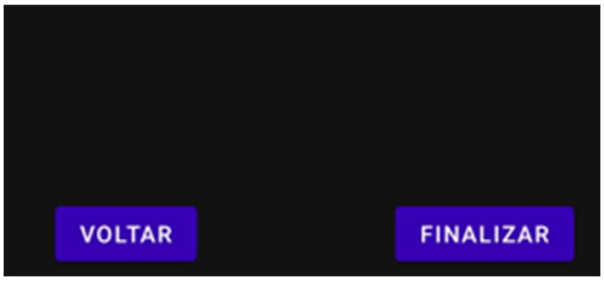
Figura 26 – Seleção dos horários.



Fonte: Autoria Própria.

Assim, finaliza-se o cadastro de um novo alarme. Pode-se, então, clicar no botão "Finalizar" para armazenar as informações no banco de dados e agendar os alarmes no *Alarm Manager*. A Figura 27 ilustra os botões para finalizar o cadastro.

Figura 27 – Botões para finalizar o cadastro.



Fonte: Autoria Própria.

O aplicativo apresenta ao usuário a tela principal com o novo alarme agendado. Assim, quando a hora definida chegar, o aplicativo acionará a tela de alarme junto de um toque sonoro para alertar o usuário que chegou o momento de administrar o medicamento.

Por conta das permissões adicionadas no aplicativo e as configurações da *activity* de alarme, a tela de alarme aparecerá mesmo com o celular em *stand by* ou bloqueado. Quando a tela de alarme é acionada pelo *Alarm Manager* um alarme sonoro do próprio celular é disparado.

A foto tirada da caixa do medicamento no momento do seu cadastro aparece e duas opções são apresentadas para o usuário: “Tomei” e “Não Tomei” (o medicamento). A opção selecionada pelo usuário será salva no banco de dados.

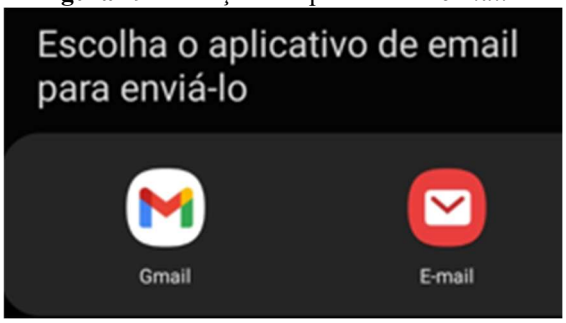
Caso o usuário clique em “Não Tomei”, depois de 3 minutos o alarme será acionado novamente e as mesmas opções serão apresentadas. Caso o usuário clique três vezes em “Não Tomei”, então o aplicativo interpretará que o medicamento não foi tomado. A Figura 28 ilustra a tela do alarme.

Figura 28 – Tela de alarme disparado.

Fonte: Autoria Própria.

Caso o usuário clique em “Tomei” o aplicativo vai checar se chegou ao fim do período de tomar o medicamento. Caso seja o último dia e hora do aplicativo disparar, então será dado a opção para o usuário escolher qual aplicativo será utilizado para enviar o *e-mail*. Na Figura 29, a imagem ilustra a seleção do aplicativo de *e-mail*.

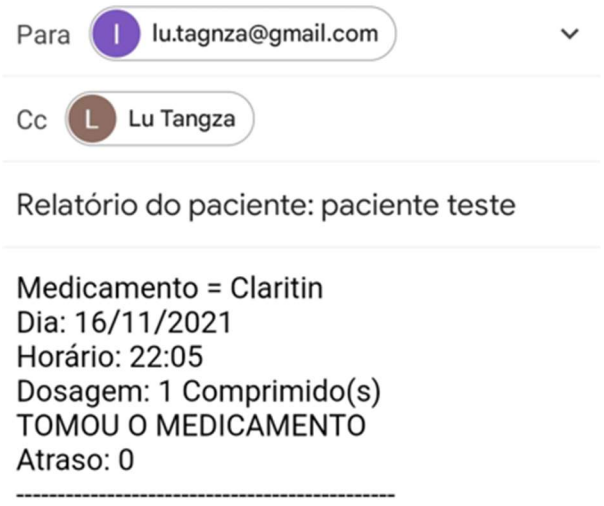
Figura 29 – Seleção do aplicativo de *e-mail*.



Fonte: Autoria Própria.

Ao selecionar o aplicativo de *e-mail* desejado, será criado automaticamente um *e-mail* com o relatório em seu corpo, com o nome do usuário no assunto e já selecionados os endereços de *e-mails* a serem enviados. O relatório contido na mensagem, possui as informações de todos os dias e horários dos alarmes com as informações se o usuário tomou ou não o medicamento e se houve atraso na hora de sua administração. O usuário precisa apenas enviar o *e-mail*, conforme ilustra a Figura 30.

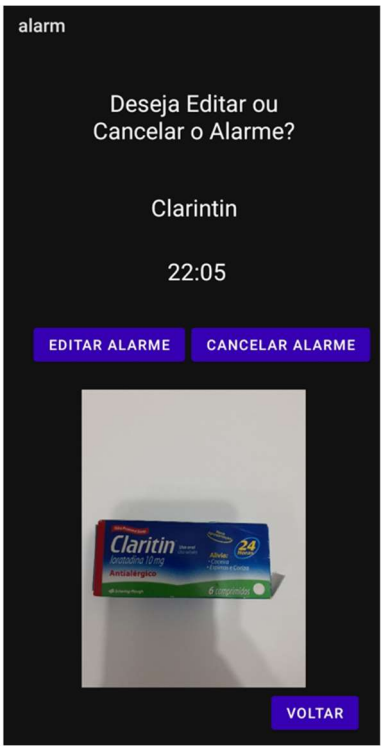
Figura 30 – *E-mail* montado automaticamente pelo aplicativo.



Fonte: Autoria Própria.

Quando um alarme está configurado, há a opção de editá-lo ou cancelá-lo por meio da tela principal. Basta clicar no alarme desejado que o aplicativo apresentará a tela de decisão para o usuário. Pode-se selecionar a opção editar o alarme, o que permitirá ao usuário configurar horário, data e imagem, como mostrado na Figura 31, mas dessa vez já estarão preenchidas com as informações previamente cadastradas. Se o usuário escolher “Cancelar alarme”, o aplicativo mostra a tela principal do aplicativo e o alarme não estará mais listado.

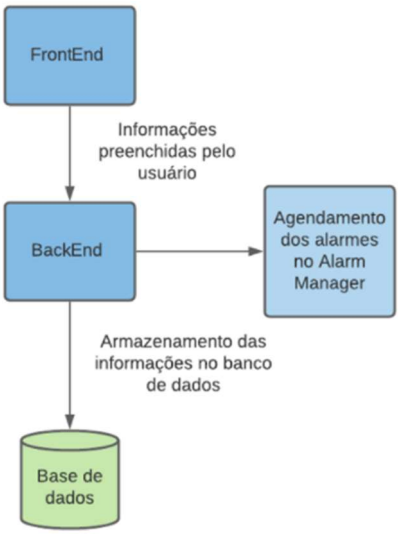
Figura 31 – Tela para editar ou cancelar o alarme.



Fonte: Autoria Própria.

Pode-se exemplificar o funcionamento do aplicativo por meio do fluxograma da Figura 32. Tem-se que o *FrontEnd* recebe as informações que são enviadas para o *BackEnd*, e este as trata e interpreta da forma adequada, conforme o código programado e implementado. As informações são inseridas nas tabelas do banco de dados do *Android* e a aplicação agenda os alarmes.

Figura 32 – Fluxograma do cadastramento do alarme de medicamentos.



Fonte: Autoria Própria.

3.4 *Sprint*s

Seguindo a Metodologia Ágil *Scrum*, desenvolveu-se o aplicativo em *sprints* para que a cada reunião ou teste, o aplicativo fosse aperfeiçoado e os *bugs* fossem corrigidos. Dessa forma, foram realizados um total de seis *sprints* ao longo do desenvolvimento, sendo elas:

- Primeira *sprint*: consistiu no desenvolvimento das funcionalidades principais do aplicativo;
- Segunda *sprint*: dividiu-se a tela de configurar o alarme em mais 4, aumento das fontes e melhoramento das descrições no aplicativo para melhor entendimento;
- Terceira *sprint*: foi adicionada a tela de dosagem do medicamento;
- Quarta *sprint*: revisão das funcionalidades com a professora Patrícia e início dos testes;
- Quinta *sprint*: testes com usuário idoso e mudança na lógica para reagendar o alarme no caso de o usuário clicar em “Não tomei”;
- Sexta *sprint*: corrigiu-se o modo como os horários estavam aparecendo: quando os minutos eram da primeira dezena, eles não apareciam com o numeral “0” na frente, podendo confundir o usuário.

4. RESULTADOS E DISCUSSÕES

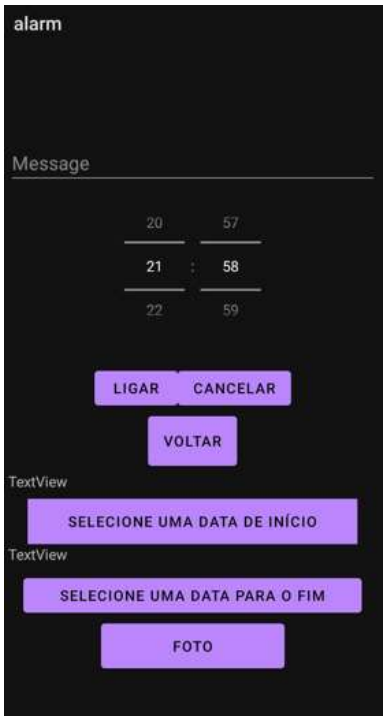
4.1 Primeira *Sprint*

Na primeira *sprint* foi levantada toda a revisão bibliográfica necessária para desenvolver o aplicativo, assim como empregar todas as suas funcionalidades principais: agendar o horário para tomar o medicamento, tocar o despertador e informar ao usuário qual o remédio que ele deve administrar e, por fim, ao término do mês ou do tratamento, enviar um *e-mail* para o médico responsável pelo tratamento do idoso. Foi adicionado também, uma tela para editar os alarmes ou cancelá-los caso necessário.

4.2 Segunda *Sprint*

Após o desenvolvimento das funcionalidades principais, realizou-se uma reunião com o professor orientador para avaliar e obter o *feedback* da aplicação. As funções adicionadas estavam de acordo com a proposta, mas a apresentação estava confusa e pouco explicativa. Consequentemente, foi sugerido pelo professor que a tela de configuração dos medicamentos fosse separada em várias outras, dividindo as informações que o usuário deveria preencher para a criação do alarme. A Figura 33 mostra a primeira tela criada de configuração do alarme. Nota-se que diferentemente das novas, a antiga possuía todos os campos juntos e sem qualquer explicação do que se deveria fazer no aplicativo.

Figura 33 – Primeira tela inicial de configuração do alarme.



Fonte: Autoria Própria.

Com isso, separou-se a única *activity* de cadastro do alarme em várias *activities*, cada uma com seu *controller* e componentes próprios para o paciente preencher. Foi sugerido também que os textos fossem mais explicativos e com uma fonte maior, deixando claro para o usuário o que ele deveria fazer e quais informações ele deveria inserir.

4.3 Terceira *Sprint*

Ao final dos ajustes e separação da tela de configuração do alarme, foi realizado outra reunião com o professor Márcio para demonstrar o novo *layout* do aplicativo. Foi sugerido então, que uma tela de posologia fosse adicionada, para que o usuário pudesse inserir a dosagem do medicamento que deveria ser administrada. Desta forma, criou-se um campo de posologia no banco de dados, bem como uma *activity* para que o paciente pudesse selecionar qual a apresentação do medicamento: em mililitro (ml), em gotas (gts), comprimidos, drágeas ou cápsulas. Na Figura 34 é possível observar esta tela de seleção.

Figura 34 – Tela onde se configura a posologia com todas as opções.

Fonte: Autoria Própria.

4.4. Quarta *Sprint*

Após o desenvolvimento da tela de dosagem, realizou-se uma reunião com a professora coorientadora para revisar as características do aplicativo. Tanto o *layout* quanto as funcionalidades foram aprovados pela professora. Foram iniciados, então, os testes unitários da aplicação, assim como os testes com usuários idosos.

Para os testes unitários, configurou-se um alarme para tocar por três dias seguidos e em dois horários distintos, conforme ilustra a Figura 35. A figura mostra também, o *e-mail* enviado pelo aplicativo após o término dos três dias.

Figura 35 – Relatório do segundo teste.

Fonte: Autoria Própria.

Nesse teste, observa-se que não foi tomado o medicamento no primeiro dia, pois o número de atrasos é igual a três e a mensagem em caixa alta informa que o medicamento não foi administrado. No segundo dia, foi tomado o medicamento em todos os horários, havendo apenas um atraso no segundo horário. No terceiro dia, o medicamento foi administrado em ambos os horários, havendo um atraso de duas tentativas no segundo horário. Para esse caso, testou-se também a funcionalidade de editar os alarmes. Nota-se que no terceiro dia há um horário a mais no relatório, isso porque adicionou-se o horário das "15:08" para esse alarme nesse dia.

Notou-se que em determinados momentos, o alarme demorava cerca de um minuto para tocar, comparando-se com o horário estipulado. Isso acontecia por conta do próprio dispositivo *Android* que gerencia as atividades de segundo plano e o *AlarmManager*. Dessa forma, se houverem muitas atividades prioritárias na frente do *AlarmManager*, o aplicativo de alarme pode sofrer leves atrasos no disparo.

A partir desse teste, foi evidenciado pelo professor Márcio, na sexta *sprint*, que os horários estão aparecendo de forma errada no relatório e no *display* da aplicação. Em casos de horários como "15:08" a aplicação mostrava como se fosse "15:8". Tal *bug* foi corrigido na sexta *sprint*.

Para os testes envolvendo o usuário idoso, instalou-se a aplicação no aparelho de duas voluntárias idosas. A aplicação foi testada por elas e os resultados foram analisados na próxima *sprint*.

4.5 Quinta *Sprint*

A partir do *feedback* dos testes das voluntárias notou-se que o aplicativo estava de fácil compreensão, sendo que sua utilização era bem simples de ler e entender por conta das letras grandes e dos textos explicativos. Porém, foi relatado que em alguns momentos o aplicativo demorava muito para tocar novamente em casos que o botão "Não tomei" foi pressionado. Segundo a usuária, em alguns momentos o alarme demorou cerca de meia hora para tocar novamente.

O motivo de tal comportamento era o método que estava sendo utilizado para agendar os alarmes e configurar o *AlarmManager*. Era utilizado o "*setRepeating*", em que era configurado um intervalo de tempo para que o alarme fosse disparado caso ele não fosse parado pelo botão "Tomei". Caso o dispositivo estivesse sobrecarregado no momento com operações de segundo plano, os disparos do alarme subsequentes poderiam ser mantidos na fila do *Android*

e disparados apenas quando o aparelho estivesse mais livre. Por conta disso, optou-se por alterar o método de "*setRepeating*" para "*setExact*".

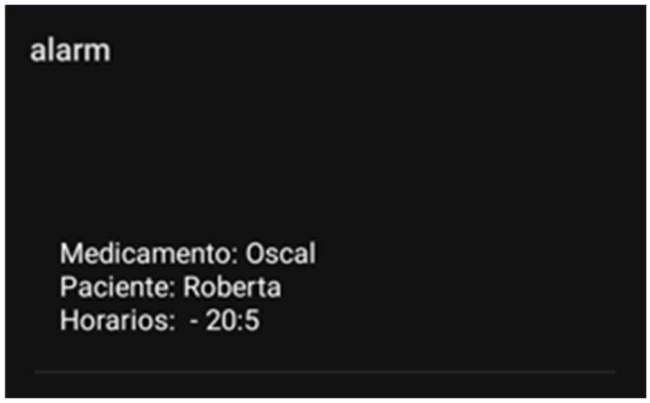
O método "*setExact*" informa ao *Android* que o alarme configurado no *AlarmManager* deve ser tocado com o máximo de precisão possível em relação ao horário. Dessa forma, ao clicar em "Não tomei", o alarme atual era cancelado e reagendado para três minutos no futuro, utilizando o método "*setExact*".

A mudança no método de configuração do *AlarmManager* fez com que o aplicativo não demorasse tanto para tocar novamente nos casos que a usuária clicava em "Não tomei". Foi relatado que o máximo de atraso que ocorreu foi de cerca de um minuto, mas tal comportamento já foi observado anteriormente.

4.6 Sexta *Sprint*

Para a sexta *sprint*, o professor Márcio alertou sobre o horário errado apresentado nos testes unitários. Ao invés de informar "15:08" o relatório apresentava o horário como "15:8". Tal comportamento foi observado nas telas do aplicativo, quando eram marcados horários cujos minutos estavam entre "00" e "09", conforme ilustra a Figura 36 tirada de um dos testes com usuários.

Figura 36 – Tela do alarme criado com o horário errado.

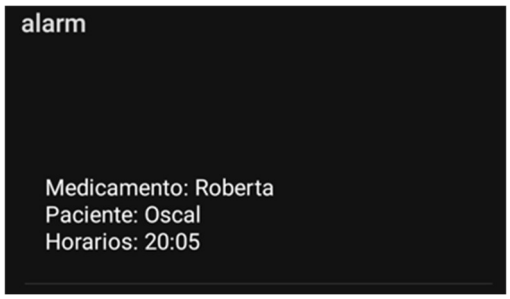


Fonte: Autoria Própria.

Isso ocorria por conta do modo com que o valor dos minutos era armazenado no banco de dados, no caso, como um inteiro. Em casos de "20:05", o valor armazenado dos minutos era apenas "5", assim, quando o valor era mostrado na tela ou no relatório, ele não apresentava o numeral "0" na frente. Para corrigir a forma como os minutos eram demonstrados, utilizou-se o método "*format*" formatando a *String* para que ela assumisse o formato de horário. Como parâmetros para o método, deve-se colocar a forma que se deseja que o horário apareça, no

caso, "*%02d:%02d*" e em seguida, os parâmetros de hora e minutos. Utilizando tal método foi possível corrigir os horários mostrados no *e-mail* e no aplicativo. Dessa forma, pediu-se para que as voluntárias testassem novamente o aplicativo. Conforme a Figura 37 ilustra, tem-se que o horário na tela principal agora está correto.

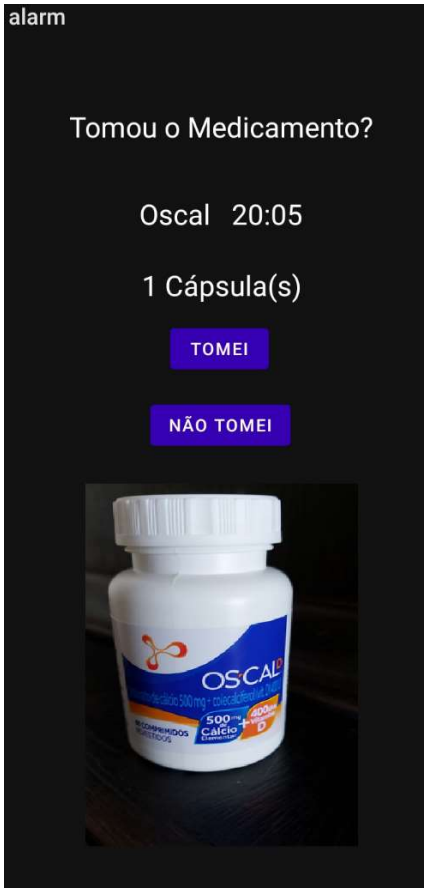
Figura 37 – Tela do alarme criado com o horário apresentado corretamente.



Fonte: Autoria Própria.

A Figura 38 mostra, também, que o horário na tela de alarme agora está correto, com o numeral "0" no campo de minutos.

Figura 38 – Tela do alarme com minutos corrigido.



Fonte: Autoria Própria.

Por fim, o horário foi corrigido também no *e-mail* enviado com o relatório dos alarmes, presente na Figura 39.

Figura 39 – Relatório do medicamento Oscal.

Fonte: Autoria Própria.

A outra voluntária testou novamente o aplicativo e não encontrou problemas com a aplicação. Os testes foram realizados com os dois medicamentos que são administrados de

noite. Conforme ilustra a Figura 40, tem-se a tela principal da aplicação com os dois medicamentos.

Figura 40 – Tela com os dois remédios programados.

Fonte: Autoria Própria.

Para o medicamento Olmesartana, obteve-se a tela de alarme conforme o esperado, como ilustra a Figura 41.

Figura 41 – Tela do alarme quando ele toca.

Fonte: Autoria Própria.

O *e-mail* com o relatório foi enviado ao término dos seis dias referentes ao teste, como ilustra a figura 42.

Figura 42 – Relatório do medicamento Olmesartana.

Fonte: Autoria Própria.

Para o segundo medicamento Rosuvastatina o alarme funcionou conforme o esperado segundo a usuária. Obteve-se a seguinte tela de alarme, conforme ilustra a Figura 43.

Figura 43 – Relatório do medicamento Rosuvastatina.

Fonte: Autoria Própria.

A tela de alarme também apresentou as informações necessárias para a usuária saber qual medicamento tomar, assim como quantos comprimidos ela deveria administrar. A Figura 44 ilustra o *e-mail* enviado pelo usuário, referente ao medicamento.

Figura 44 – Fonte: Autoria Própria.

Fonte: Autoria Própria.

5. CONCLUSÃO

Foi desenvolvido um aplicativo para dispositivos *Android* capaz de auxiliar o usuário a lembrar da hora certa para tomar seus medicamentos, assim como possibilitar ao médico ou cuidador o acompanhamento do tratamento do paciente por meio de um relatório.

Para a implementação do aplicativo, foi utilizada a Metodologia Ágil (*Scrum*), o que facilitou muito o processo de desenvolvimento, já que a cada *sprint* o aplicativo era melhorado com base nos testes executados pelas voluntárias e das opiniões e sugestões dadas pelo orientador e pela coorientadora, como por exemplo, o tamanho da fonte dos textos.

Durante os *sprints*, algumas falhas foram corrigidas, como os numerais dos minutos que estavam sendo mostrados de forma errada, bem como o alarme que não tocava de novo em algumas ocasiões após selecionar o botão “Não tomei”.

Para validar o aplicativo, o mesmo também foi utilizado por duas voluntárias idosas ao longo da quarta *sprint*, sendo corrigido o *bug* encontrado por uma delas na quinta *sprint*. Além disso, ao término dos ajustes da sexta *sprint*, as voluntárias testaram o aplicativo por 6 dias e não relataram nenhum problema.

Conclui-se que o aplicativo implementado é capaz de alertar o usuário para tomar os seus medicamentos, bem como enviar um *e-mail* com o relatório contendo todas as informações que ocorreram ao longo do período de administração dos medicamentos. Sua interface é amigável, intuitiva, de fácil compreensão, com fontes grandes e foto dos medicamentos, o que auxilia os usuários, principalmente idosos, na sua identificação.

6. SUGESTÕES PARA TRABALHOS FUTUROS

Para trabalhos futuros, propõe-se a possibilidade de customização da aplicação, podendo selecionar diferentes toques e músicas para serem tocadas no alarme. Realizar um estudo sobre a utilização de cores, fontes, ícones, entre outras configurações do aplicativo para torná-lo ainda mais intuitivo e atraente. Verificar as normas e procedimentos para disponibiliza-lo na *Play Store*. Para isso, deve-se implementar um fluxo de *logs* capazes de enviar informações para o desenvolvedor quando *bugs* ou *crashes* acontecerem na aplicação.

7 REFERÊNCIAS

[1] LINEBUR, J. M. e RUSCIN, S. A. **Medicamentos e envelhecimento**. Manual MSD. 2018. Disponível em: <https://www.msdmanuals.com/pt-br/casa/quest%C3%B5es-sobre-a-sa%C3%BAde-de-pessoas-idosas/medicamentos-e-envelhecimento/medicamentos-e-envelhecimento>. Acesso em: 10 de jan. de 2021.

[2] SANTO REMÉDIO. **Saúde do idoso: 5 cuidados essenciais ao tomar medicamentos.** Drogaria Santo Remédio. 2018. Disponível em: <https://drogariasantoremedio.com.br/saude-do-idoso/>. Acesso em: 10 de dez. de 2021.

[3] BAYER. **Aplicativo Hora da Pílula**. Bayer Para Você. Disponível em: <https://paravoce.bayer.com.br/hora-da-pilula>. Acesso em: 11 de dez. de 2021.

[4] HALFEN, E. **Hora do remédio.** Apple Play Store. Disponível em: <https://apps.apple.com/br/app/hora-do-rem%C3%A9dio/id526927576>. Acesso em: 11 de dez. de 2021.

[5] HE:LABS. ***Scrum***. Desenvolvimento Ágil. Disponível em: <https://www.desenvolvimentoagil.com.br/scrum/ >. Acesso em: 10 de jan. de 2021.

[6] COLUNISTA PORTAL. **Cuidados no Manuseio dos Medicamentos dos Idosos**. Portal Educação. Disponível em: <https://www.portaleducacao.com.br/conteudo/artigos/enfermagem/cuidados-no -manuseio-dos-medicamentos-dos-idosos/19885>. Acesso em: 10 de jan. de 2021.

[7] SILVA, C. H. e SPINILLO, C. G. Dificuldades e estratégias no uso de múltiplos medicamentos por idosos no contexto do design da informação. **Estudos em Design, Revista (Online).** v. 24, n. 3., p. 130 – 144, 2016.

[8] FERREIRA, D. P. e JUNIOR, S. C. S. Aplicativos móveis desenvolvidos para crianças e adolescentes que vivem com doenças crônicas: uma revisão integrativa. **Interface – Comunicação, Saúde, Educação.** v. 25, p. 1 – 17, 2021.

[9] BARRA, D.C.C., PAIM S.M.S., SASSO G.T.M.D. e COLLA G.W. Métodos para desenvolvimento de aplicativos móveis em saúde: revisão integrativa da literatura. **Texto Contexto Enfermagem**. v. 26, 2017.

[10] SOUZA, C. M. e SILVA, A. N. Aplicativos para smartphones e sua colaboração na capacidade funcional de idosos. **Rede Saúde Digital e Tecnologias Educacionais**. v. 1, n. 1, p. 06-19, jan./jul.2016.

[11] DÂMASO, L. **O que é *app*? Quatro perguntas e respostas sobre aplicativos para celular.** Techtudo. 2019. Disponível em: <https://www.techtudo.com.br/noticias/2019/12/o-

que-e-app-quatro-perguntas-e-respostas-sobre-aplicativos-para-celular.ghtml>. Acesso em: 17 de jan. de 2021.

[12] GOOGLE LLC. ***Android Studio***. *Android* Studio. 2020. Disponível em: <https://*Android*-studio.br.uptodown.com>. Acesso em: 17 de jan. de 2021.

[13] MENEZES, K. **O que é SDK e quais as suas vantagens no desenvolvimento e uso de aplicações mobile**. Idblog. 2018. Disponível em: <https://blog.idwall.co/o-que-e-sdk-e-vantagens-para-mobile/>. Acesso em: 14 jan. de 2021

[14] ARRUDA, R. **Programando em *Java Android***. GeekHunter. 2020. Disponível em: <https://blog.geekhunter.com.br/*Java*-*Android*/>. Acesso em: 14 jan. de 2021.

[15] ABLESON, F. et al. *Android* em ação. 3ª Edição. Editora Campus, 2012

[16] DEVELOPERS**. Ciclo de vida de uma *activity.*** Developer *Android*. 2020. Disponível em:< https://developer.*Android*.com/reference/*Android*/app/Activity>. Acesso em: 15 jan. de 2021.

[17] DEVELOPERS**. *Android Button.*** Developer *Android*. 2020. Disponível em: < https://developer.*Android*.com/reference/*Android*/widget/Button>. Acesso em: 16 jan. de 2021.

[18] DEVELOPERS**. *Android TextView.*** Developer *Android*. 2020. Disponível em: < https://developer.*Android*.com/reference/*Android*/widget/TextView>. Acesso em: 18 jan. de 2021.

[19] DEVELOPERS**. *Android Time Picker.*** Developer *Android*. 2020. Disponível em: < https://developer.*Android*.com/reference/*Android*/widget/TimePicker>. Acesso em: 18 jan. de 2021.

[20] DEVELOPERS**. *Android Calendar.*** Developer *Android*. 2020. Disponível em: < https://developer.*Android*.com/reference/*Java*/util/Calendar>. Acesso em: 17 jan. de 2021.

[21] Y.S. ***Android custom calendar / date picker***. Stack Overflow. 2021. Disponível em: <https://stackoverflow.com/questions/34672358/*Android*-custom-calendar-date-picker>. Acesso em: 20 abr. de 2021.

[22] DEVMEDIA. ***Android* MVC: Criando um *Framework Model-View-Controller* para *Android*.** Devmedia. 2014. Disponível em: < https://www.devmedia.com.br/*Android*-mvc-criando-um-framework-model-view-controller-para-*Android*/29924>. Acesso em: 20 abr. de 2021.

[23] SCHILDT, H. ***Java*:** para iniciantes. 6. ed. Porto Alegre: Bookman, 2015.

[24] DEITEL, H. M.; DEITEL, P. J. ***Java:*** como programar. Porto Alegre: Bookman, 2003.

[25] DEVELOPERS**. Banco de dados *SQLite***. Developer *Android*. 2020. Disponível em: <https://developer.*Android*.com/training/data-storage/sqlite?hl=pt-br>. Acesso em: 14 jan. de 2021.

[26] DEVELOPERS. ***SQLiteOpenHelper***. Developer *Android*. 2020. Disponível em: < https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper >. Acesso em: 14 de fev. de 2021.

[27] DEVELOPERS. ***Android Intent***. Developer *Android*. 2020. Disponível em: <https://developer.*Android*.com/guide/components/intents-filters?hl=pt-br>. Acesso em: 14 fev. de 2021.

[28] DEVELOPERS. ***AlarmManager***. Developer *Android*. 2020. Disponível em: < https://developer.*Android*.com/reference/*Android*/app/AlarmManager>. Acesso em: 14 fev. de 2021.

[29] DEVELOPERS. **Transmissores e receptores**. Developer *Android*. 2020. Disponível em: <https://developer.*Android*.com/guide/components/broadcasts?hl=pt-br>. Acesso em: 14 fev. de 2021.

[30] QASTACK. **Enviar *e-mail* via *intent***. Qastack. Disponível em: < https://qastack.com.br/programming/8701634/send-*e-mail*-intent>. Acesso em: 5 mar. de 2021.

[31] DEVELOPERS. **Permissões do *Android***. Developer *Android*. 2020. Disponível em: <https://developer.*Android*.com/training/permissions/requesting>. Acesso em: 5 mar. de 2021.

[32] DEVELOPERS. **Modo desenvolvedor**. Developer *Android*. 2020. Disponível em: <https://developer.*Android*.com/studio/debug/dev-options?hl=pt-br>. Acesso em: 2 jun. de 2021.