

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO E TELECOMUNICAÇÕES

CARLOS ALBERTO SOARES DE ANDRADE

**APLICATIVO PARA O AUXÍLIO À ROTINAS MEDICAMENTOSAS: UMA IMPLEMENTAÇÃO PARA PESSOAS IDOSAS E PESSOAS COM DEFICIÊNCIA.**

BELÉM - PA

2023

CARLOS ALBERTO SOARES DE ANDRADE

**APLICATIVO PARA O AUXÍLIO À ROTINAS MEDICAMENTOSAS: UMA IMPLEMENTAÇÃO PARA PESSOAS IDOSAS E PESSOAS COM DEFICIÊNCIA.**

Trabalho de Conclusão de Curso

para obtenção do grau de

Bacharel em Engenharia da

Computação pela Universidade

Federal do Pará. Orientador:

Prof. Agostinho Luiz da Silva

Castro.

BELÉM - PA

2023

CARLOS ALBERTO SOARES DE ANDRADE

**APLICATIVO PARA O AUXÍLIO À ROTINAS MEDICAMENTOSAS: UMA IMPLEMENTAÇÃO PARA PESSOAS IDOSAS E PESSOAS COM DEFICIÊNCIA.**

**Trabalho de Conclusão de Curso para obtenção do grau de Bacharel em Engenharia da Computação pela Universidade Federal do Pará.**

Aprovado em: \_\_\_/\_\_\_/\_\_\_

Conceito: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Banca examinadora:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Agostinho Luiz da Silva Castro

(Orientador)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Doutor Ronaldo de Freitas Zampolo

Faculdade de Engenharia da Computação e Telecomunicações/UFPA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professora Mestra Kelly Vale Pinheiro

Faculdade de Fisioterapia e Terapia Ocupacional/ UFPA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. Leonardo Lira Ramalho

Diretor da Faculdade de Engenharia da Computação e Telecomunicações

DEDICATÓRIA

"As drogas não funcionam em pacientes que não as tomam" (KOCUREK, 2009).

“Ser sujeito de direitos significa que qualquer um, ou qualquer uma, tem o direito de ter direitos” (UNESCO, 2007).

“Ao abordar este texto apresentado pela UNESCO não se tem a intenção de classificar o idoso como um deficiente, mas sim propor a discussão de que tanto o idoso quanto a pessoa com deficiência, deve requerer e exercer seus direitos, para garantir que sua voz e vontade sejam ouvidas. Sendo assim, é dever do Estado garantir aos jovens, deficientes e ao adulto idoso o acesso à educação. No que diz respeito ao adulto idoso, o Estado deve oferecer recursos não apenas para educação escolar regular, mas também para o ensino das novas tecnologias (inclusão digital). Para que os idosos consigam exercer a sua cidadania, sem o sentimento de exclusão da sociedade por não acompanhar os avanços tecnológicos. ” (TAVARES, 2012).

AGRADECIMENTOS

A DEUS que me deu a vida e a vontade de estudar,

Ao meu pai MANOEL VICENTE DE ANDRADE que foi um exemplo para mim de trabalho e família,

A minha mãe RAIMUNDA SOARES DE ANDRADE que me educou no caminho da verdade e consciência,

A minha amada esposa MARIA DO SOCORRO que sempre me apoio nos estudos,

A todos os meus filhos GILBERTY, LUTIAN e HINEIDA por seu incentivo a nunca desistir,

Ao meu professor do coração DR. AGOSTINHO que foi como um pai me orientando em todo o processo, mesmo de madrugada.

Ao amigo conhecido ou desconhecido que de alguma forma me incentivou a sempre acreditar e nunca desistir.

RESUMO

O censo de 2010, mostrou que quase 46 milhões de brasileiros assumem possuir deficiência física ou intelectual. Dentro desse cenário se torna necessário criar formas de auxiliar esse público na administração de atividades básicas. Para tanto, o uso da tecnologia é uma forma factível para auxiliar e melhorar a qualidade de vida dessas pessoas, tais como: as com deficiências (PcDs) e as idosas.

A disponibilidade de tecnologia aliada com a maior facilidade de acesso a dispositivos móveis por essas pessoas, abre oportunidades para o uso da tecnologia para ajudar, principalmente, os que possuem alguma deficiência. Nesse contexto, o presente trabalho se traduz na concepção de uma arquitetura de software que, implementada como um aplicativo móvel, pode ajudar pessoas com limitações físicas em sua rotina medicamentosa. Devido à grande variabilidade de situações envolvendo a perda ou limitação física de algum sentido de pessoas e embora a arquitetura proposta neste trabalho possa se adaptar a diversas situações de acordo com o perfil de usuários, este trabalho desenvolve um *case* dessa arquitetura bastante voltado para o usuário com baixa acuidade visual. Sendo assim, o aplicativo mobile, chamado de AUM, foi desenvolvido com a intenção de assistir as pessoas com alteração na acuidade visual, sendo capaz de gerenciar a rotina de medicamentos. Em essência, o aplicativo possui características para disponibilizar informações sobre o medicamento, a sua posologia, horários para a administração do medicamento, entre outras facilidades. Testes (LINT, Scanner Acessibilidade e com o site Guia de Referência de Acessibilidade) foram realizados e por esses o aplicativo foi ajustado para atender as recomendações quanto aos quesitos de acessibilidade. Uma versão do AUM está disponível para uso e avaliação na loja de aplicativos Play Store. Esta fase de disponibilização do AUM na loja é importante para as realimentações em termos de opinião dos usuários e desempenho do aplicativo. Em se tratando de um aplicativo concebido de tal forma a incorporar vários perfis de usuários, a meta futura é incorporar outros *cases* que atendam diferentes níveis de acessibilidade em conformidade com o perfil do usuário.

**Palavras-chave**: Tecnologia, aplicativo, pessoas com deficiência, rotinas, medicamentos.

ABSTRACT

The 2010 census showed that almost 46 million Brazilians admit to having a physical or intellectual disability. Within this scenario, it becomes necessary to create ways to assist this public in the administration of basic activities. Therefore, the use of technology is a feasible way to help and improve the quality of life of these people, such as: those with disabilities (PwDs) and the elderly.

The availability of technology combined with greater ease of access to mobile devices by these people opens up opportunities for using technology to help, mainly, those who have a disability. In this context, the present work translates into the design of a software architecture that, implemented as a mobile application, can help people with physical limitations in their medication routine. Due to the great variability of situations involving the loss or physical limitation of some sense of people and although the architecture proposed in this work can adapt to different situations according to the user profile, this work develops a case of this architecture very focused on the user with low visual acuity. Therefore, the mobile application, called AUM, was developed with the intention of assisting people with changes in visual acuity, being able to manage the medication routine. In essence, the application has features to provide information about the drug, its dosage, times for administering the drug, among other facilities. Tests (LINT, Accessibility Scanner and the Accessibility Reference Guide website) were carried out and for these the application was adjusted to meet the recommendations regarding accessibility requirements. A version of AUM is available for use and evaluation on the Play Store app store. This phase of making the AUM available in the store is important for feedback in terms of user opinion and application performance. As it is an application conceived in such a way as to incorporate several user profiles, the future goal is to incorporate other cases that meet different levels of accessibility in accordance with the user's profile.

**Keywords**: Technology, application, people with disabilities, routines, medication.

LISTA DE FIGURAS

Figura 1 - Percentual da População com algum grau de deficiência........................................... 3 Figura 2 - Aqueduto romano............................................................................................................. 12 Figura 3 - Objeto JSON representando dados de uma pessoa física........................................ 13 Figura 4 - Estrutura do Strategy....................................................................................................... 14 Figura 5-Estrutura do Chain of Responsability .............................................................................. 15 Figura 6 - Envolvendo um objeto com o padrão Proxy ................................................................ 16 Figura 7 - Estrutura de acesso a dados do Aplicativo AUM ........................................................ 17 Figura 8 - Estrutura em nó do banco de dados ............................................................................. 19 Figura 9 - Visão geral da concepção do aplicativo AUM. ............................................................ 20 Figura 10 - Entrada de Dados .......................................................................................................... 21 Figura 11 - Listas de Pesquisa......................................................................................................... 21 Figura 12 - Processos Autônomos .................................................................................................. 22 Figura 13 - Representa o barramento de serviços........................................................................ 22 Figura 14 - Componente de Integração de Bulário....................................................................... 23 Figura 15 - Serviço de Bulário Eletrônico da Anvisa na nuvem.................................................. 23 Figura 16 - Componente de Integração de Usuário...................................................................... 24 Figura 17 - Interface de Conexão com os perfis ........................................................................... 24 Figura 18 - Representa qualquer perfil suportado pela interface................................................ 24 Figura 19 - Componente de Acesso ao Banco de Dados............................................................ 25 Figura 20 - Banco de Dados Remoto Firebase ............................................................................. 25 Figura 21 - Janela de Acesso ao Aplicativo AUM ......................................................................... 26 Figura 22 - Cadastro de Usuário...................................................................................................... 27 Figura 23 - Autenticação de Usuário............................................................................................... 28 Figura 24 - Janela Principal .............................................................................................................. 29 Figura 25 - Fragmento Responsável pela criação da Interface de Pesquisa do Bulário

Eletrônico............................................................................................................................................. 31 Figura 26 - Fragmento de um método de usuário sem necessidade de acessibilidade ......... 32 Figura 27 - Fragmento de um método que trata usuário idoso e/ou com visão reduzida/perda ............................................................................................................................................................... 33 Figura 28 - Estrutura de Pacotes da Integração Bulário Eletrônico ........................................... 34 Figura 29 - Janela principal com a Lista de Medicamentos......................................................... 35 Figura 30 - Representa os comandos por voz............................................................................... 36 Figura 31 - Fragmento que executa o comando de Voz de Retorno da Lista de

Medicamentos..................................................................................................................................... 38 Figura 32 - Fragmento da Definição da Classe Principal com implementação do Recurso de Voz........................................................................................................................................................ 39

Figura 33 - Fragmento que executa o método de integração de usuário para lista medicamentos..................................................................................................................................... 40 Figura 34 - Interface de Transações ............................................................................................... 41 Figura 35 - Classe que Implementa o Proxy de Transação ........................................................ 42 Figura 36 - Implementação da execução do Proxy....................................................................... 43 Figura 37 - Detalhes do Medicamento............................................................................................ 44 Figura 38 - Alterar Medicamento...................................................................................................... 44 Figura 39 - Sinalizar Dose Utilizada ................................................................................................ 44 Figura 40 - Horário Ativo de um medicamento .............................................................................. 45 Figura 41 - Lista de Saldo de Estoque............................................................................................ 45 Figura 42 - Lista de Utilizações........................................................................................................ 46

Figura 43 - Exibir lista de Cores....................................................................................................... 46 Figura 44 - Paleta de Cores.............................................................................................................. 46 Figura 45 - Entrada de Estoque ....................................................................................................... 47 Figura 46 - Saída no estoque........................................................................................................... 47 Figura 47 - Cadastro de Horário ...................................................................................................... 47 Figura 48 - Representa os comandos por voz............................................................................... 48 Figura 49 - Fluxo do Reconhecimento de Comandos Por Voz................................................... 49 Figura 50 - Implementação do Ouvinte no Detalhe do Medicamento........................................ 50 Figura 51 - Criação do Recurso de Voz através da Classe RecursoVozObserver ................. 51 Figura 52 - Controle de reconhecimento de Voz........................................................................... 52 Figura 53 - Assistente Google de Voz............................................................................................. 52 Figura 54 - Tratamento de dados após a captura de voz ............................................................ 53 Figura 55 - Retorno do Controle de Execução ao Ouvinte.......................................................... 54 Figura 56 - Pesquisa da ANVISA..................................................................................................... 55 Figura 57 - Detalhes da Bula ANVISA - Parte 1............................................................................ 56 Figura 58 - Detalhes da Bula ANVISA - Parte 2............................................................................ 57 Figura 59 - Detalhes da Bula ANVISA - Parte 3............................................................................ 58 Figura 60 - Cadastro do Medicamento e horário........................................................................... 59 Figura 61 - Interface de Bulário eletrônico .................................................................................... 60 Figura 62 - Estrutura do Pacote de Integração Bulário Eletrônico ............................................. 61 Figura 63 - Modelo de Dados ........................................................................................................... 62 Figura 64 - Configuração Android Studio ....................................................................................... 64 Figura 65 - Versão do Java............................................................................................................... 65 Figura 66 - Configurações Sistema Operacional e Computador ................................................ 65 Figura 67 - Configuração básica da ferramenta estática de inspeção de código .................... 67 Figura 68 - Recorte do relatório final de inspeção de código feito pela ferramenta LINT....... 68 Figura 69 - Instalador do Scanner de Acessibilidade ................................................................... 70 Figura 70 - Configurações do Tablet utilizado para teste com o Aplicativo Scanner

Acessibilidade ..................................................................................................................................... 71 Figura 71 - Informações da Versão do Android presente no Tablet .......................................... 71 Figura 72 - Ativação do serviço do Scanner Acessibilidade........................................................ 72 Figura 73 - Forma de Execução do Scanner de Acessibilidade ................................................. 72

LISTA DE TABELAS

Tabela 1 - Resultado mais relevantes da inspeção feita pelo LINT........................................... 69 Tabela 2 - Resultado dos testes do Scanner Acessibilidade ...................................................... 73

LISTA DE ABREAVIATURAS E SIGLAS

ANVISA Agência Nacional de Vigilância Sanitária API Application Programming Interface AUM Auxilio no Uso de Medicamento

DAO Data Access Object

GIT sistema de controle de versões distribuído IBGE Instituto Brasileiro de Geografia e Estatística ID Identity

IDAO Interface Data Access Object

IDE Integrated Development Environment IOS iPhone Operating System

JSON JavaScript Object Notation

LINT Ferramenta de análise de código estática NoSQL No Structured Query Language

PCD Pessoa com deficiência

PDF Portable Document Format

QR Code Quick Response

RAM Random-Acess Memory

REST Representational State Transfer

SGBD Data Base Management System

SQL Structured Query Language

TEA Transtorno do Espectro Autista

UBS Unidade Básica de Saúde

UFPB Universidade Federal da Paraíba URL Uniform Resource Locator

SUMÁRIO

**1 INTRODUÇÃO** .................................................................................................................................. 1 1.1. MOTIVAÇÃO............................................................................................................................. 2 1.2. OBJETIVOS............................................................................................................................... 4

1.2.1. Objetivo Geral ....................................................................................................................... 4 1.2.2. Objetivos Específicos............................................................................................................. 4 1.3. ORGANIZAÇÃO DO TRABALHO.......................................................................................... 5 **2 CONTEXTUALIZAÇÃO DO PROBLEMA**.................................................................................... 6 **3 FUNDAMENTAÇÃO TEÓRICA** ................................................................................................... 10 3.1. *APPLICATION PROGRAMMING INTERFACE*................................................................. 11 3.2. FORMATO DE DADOS *JAVA SCRIPT OBJECT NOTATION (JSON)* ......................... 12 3.3. PADRÕES DE PROJETO..................................................................................................... 14 3.3.1. *Strategy*............................................................................................................................. 14 3.3.2. *Chain of Responsibility* ................................................................................................... 15 3.3.3. *Proxy*.................................................................................................................................. 15 3.3.4. *Data Access Object*......................................................................................................... 16 3.4. BANCO DE DADOS *FIREBASE*.......................................................................................... 18 3.5. ARQUITETURA UTILIZADA................................................................................................. 20 3.5.1. Cliente ............................................................................................................................... 21 3.5.2. Barramento de Serviços................................................................................................. 22 3.5.3. Interface de Integração Bulário Eletrônico .................................................................. 22 3.5.4. Interface de Integração de Usuário .............................................................................. 23 3.5.5. Interface de Acesso a Dados......................................................................................... 24 **4 DESENVOLVIMENTO** ................................................................................................................... 26 4.1. O APLICATIVO ....................................................................................................................... 26 4.1.1. Primeiro Acesso............................................................................................................... 26 4.1.2. Janela Principal................................................................................................................ 34 4.1.3 – Para o perfil visão reduzida/perda.............................................................................. 35 4.1.4. Comandos Disponíveis................................................................................................... 36 4.1.5. Alarmes ............................................................................................................................. 40 4.1.6. Detalhes do Medicamento ............................................................................................. 43 4.1.7. Comandos Disponíveis................................................................................................... 48 4.2. ESTRUTURA DE BANCO DE DADOS AUM..................................................................... 62 4.3. DICIONÁRIO DE DADOS ..................................................................................................... 63 **5 RESULTADO E TESTES** .............................................................................................................. 64 **5.1 Informações gerais**................................................................................................................ 64

**5.2 Testes** ....................................................................................................................................... 66 5.1. LINT ...................................................................................................................................... 66 5.2. Scanner Acessibilidade...................................................................................................... 69 5.3. Guia para o Desenvolvimento de Aplicações Móveis Acessíveis............................... 75

**6 CONCLUSÃO**.................................................................................................................................. 78 REFERÊNCIAS BIBLIOGRÁFICAS................................................................................................ 80 **APÊNDICE A - DICIONÁRIO DE DADOS**..................................................................................... 84

Tabela de Usuários ........................................................................................................................ 84 Tabela Medicamentos.................................................................................................................... 84 Tabela de Estoque ......................................................................................................................... 86 Tabela de Alarmes ......................................................................................................................... 86 Tabela de Horários......................................................................................................................... 86 Tabela de Utilização....................................................................................................................... 87 **APÊNDICE B - LINKS IMPORTANTES**......................................................................................... 88

**1 INTRODUÇÃO**

Baseado no Censo de 2010, conforme Instituto Brasileiro de Geografia e Estatística (IBGE) (IBGE, 2010), quase 46 milhões de brasileiros, isto é, cerca de 24% da população, assumiram ter algum grau de dificuldade em pelo menos uma das habilidades investigadas (enxergar, ouvir, caminhar ou subir degraus), ou possuir deficiência mental ou intelectual. Na mesma pesquisa, o IBGE procurou captar a percepção da dificuldade em ouvir, enxergar e caminhar ou subir escadas, mesmo contando com facilitadores como aparelhos auditivos, lentes de contato e bengalas.

Seguindo orientações internacionais, considerou-se “pessoa com deficiência” os indivíduos que responderam ter pelo menos muita dificuldade em uma ou mais questões. Considerando somente os que possuem grande ou total dificuldade para enxergar, ouvir, caminhar ou subir degraus (ou seja, pessoas com deficiência nessas habilidades), além dos que declararam ter deficiência mental ou intelectual, temos mais de 12,5 milhões de brasileiros nessa condição, o que corresponde a 6,7% da população (IBGE, 2010).

Para além do cenário descrito, esta parcela da população ainda pode ter necessidades de saúde especiais em virtude de sua deficiência. Associadas às causas ou às deficiências provocadas por elas, existem processos mórbidos que se instalam mais facilmente ou são parte integrante do processo incapacitante, conforme (CASTRO et al., 2010). Essas morbidades exigem maior atenção em saúde por terem o potencial de agravar a incapacidade ou de prejudicar a qualidade de vida e a saúde da pessoa com deficiência, sendo o medicamento um importante elemento de controle (BRASIL, 2010).

O medicamento, segundo o Ministério da Saúde, é o produto farmacêutico com finalidade profilática, curativa, paliativa ou para fins de diagnóstico. No processo saúde-doença, o medicamento assume importante papel, tanto no que se refere às políticas de gerenciamento do sistema de saúde quanto para os profissionais de saúde envolvidos (BRASIL, 2007).

Para a maioria das pessoas a tecnologia serve para facilitar as tarefas. Porém, no caso das pessoas com deficiência, ela pode representar uma ajuda ainda mais

1

importante, tornando possíveis as atividades como administrar um medicamento. Aplicativos e soluções podem fazer grande diferença na conexão dessas pessoas com o mundo ao seu redor e, para isso, desenvolvedores são capazes de aproveitar tecnologias que fazem parte do dia a dia das pessoas em prol de uma missão bastante nobre, fazer chegar a acessibilidade, por exemplo, no uso de medicamentos por esses públicos (BERSCH, 2017).

Observando-se esse cenário associado às pessoas com deficiência (PcDs), nota-se uma grande semelhança com os problemas enfrentados por pessoas idosas (perda de uma ou mais habilidades, dificuldade na realização de tarefas do cotidiano e uso de medicamentos). Este fato permite que uma mesma solução seja utilizada por ambos os públicos.

Assim, o presente trabalho aborda o desenvolvimento do aplicativo para auxiliar pessoas com deficiências e idosas (AUM) e que tem por finalidade ajudar no processo de administração de medicamentos, tendo como principal alvo pessoas com perda de visão e/ou as que são acompanhadas por cuidadores ou familiares.

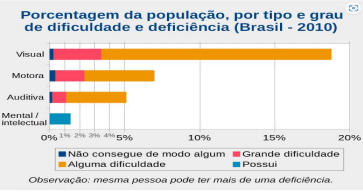
1.1. MOTIVAÇÃO

Como pode ser visto na Figura 1, em 2010, a deficiência visual estava presente em 3,4% da população brasileira; a deficiência motora em 2,3%; a deficiência auditiva em 1,1% e a deficiência mental/intelectual em 1,4%.

Ao considerarmos também as pessoas com alguma dificuldade nas habilidades pesquisadas, vemos que 18,8% da população apresentou dificuldade para enxergar; 7,0% tinha dificuldade em se movimentar; e 5,1% possuía dificuldade para ouvir.

2

Figura 1 - Percentual da População com algum grau de deficiência

Fonte: Censo IBGE. Pessoas com deficiência. 2010

Esses dados, referentes ao percentual de pessoas com algum grau de deficiência na população, são informações relevantes, pois mostram que uma parcela significativa da população precisa de atenção, principalmente os deficientes visuais. Tais dados se tornam um indicativo que revela oportunidades para estudos na busca de soluções de problemas que hoje afetam uma parcela da população. De acordo com alguns estudos, pessoas com alterações visuais necessitam de um acompanhamento especial, correlacionando com o cuidado e proteção no que diz respeito às pessoas responsáveis, como por exemplo a família, pois encontram no dia a dia muitas dificuldades de interação e locomoção (NASCIMENTO; MARQUES, 2009).

Esse contexto de busca de soluções para os cidadãos, dentro desse perfil, foi o principal motivador para a proposta indicada neste trabalho, ou seja, a concepção e implementação de um aplicativo que auxilie na rotina medicamentosa de pessoas idosas e com deficiência visual.

3

1.2. OBJETIVOS

Este trabalho é composto de um objetivo geral e alguns objetivos específicos. Os objetivos específicos podem ser entendidos como componentes a serem alcançados na tentativa de contemplar o que é almejado no objetivo geral.

**1.2.1. Objetivo Geral**

Desenvolver um aplicativo mobile, para plataforma Android, que seja capaz de auxiliar pessoas na adesão a rotinas medicamentosas prescritas por profissionais da área da saúde, com foco em pessoas com deficiências e idosas. Também é objetivo para o aplicativo auxiliar os responsáveis pelo paciente (cuidadores ou familiares) na manutenção e gestão dessas rotinas, com a possibilidade de preservar a autonomia e iniciativa do paciente e, ainda, possibilitando a supervisão do responsável.

**1.2.2. Objetivos Específicos**

Desenvolver um aplicativo mobile capaz de:

● armazenar o cadastro de medicamentos em uma base de dados remota; ● permitir pesquisa e captura dos dados da bula registrados na ANVISA; ● permitir o cadastro de medicamentos e controlar os respectivos horários;

● permitir e controlar o saldo em estoque e o número de utilizações dos medicamentos;

● alertar os usuários quanto ao horário das doses e de doses em atraso; ● notificar os usuários de doses em atraso.

4

1.3. ORGANIZAÇÃO DO TRABALHO

Este trabalho foi organizado da seguinte maneira:

Capítulo 1: Introdução - Este capítulo apresenta o contexto para o qual o trabalho foi concebido, motivação, objetivos e estrutura delineada para a descrição do que foi executado.

Capítulo 2: Contextualização do Problema - Neste capítulo é feito um estudo das técnicas já existentes para redução da não-adesão à medicação para pacientes com diferentes diagnósticos e modo de vida. Além disso, também é abordado os aplicativos já existentes no mercado e como que se assemelham ao aplicativo sugerido neste trabalho.

Capítulo 3: Fundamentação teórica - para um bom entendimento do processo de desenvolvimento do trabalho, o capítulo versa sobre os aspectos técnicos que foram usados no trabalho.

Capítulo 4: Desenvolvimento – Este capítulo explana a arquitetura de *software*, bem como fragmentos de código fonte e discussão em torno de tudo que foi utilizado e feito no desenvolvimento do aplicativo.

Capítulo 5: Resultados – Capítulo reservado para apresentar como o aplicativo elaborado é capaz de solucionar o problema abordado neste trabalho. Contando também com resultados de testes operacionais feitos para comprovar a usabilidade do produto.

Capítulo 6: Conclusão e Trabalhos Futuros – Neste capítulo é feita uma discussão do problema geral, a solução elaborada por esse trabalho e suas limitações. Além disso, também é abordado as possíveis aplicações do aplicativo desenvolvido ao decorrer deste trabalho, suas possíveis limitações técnicas e como provavelmente podem ser superadas em trabalhos subsequentes.

5

**2 CONTEXTUALIZAÇÃO DO PROBLEMA**

Encontramos na literatura acadêmica pesquisas que revelam muitas causas e fatores ligados a baixa adesão a uma rotina eficaz no uso de medicamentos por pessoas idosas e pessoas que apresentam deficiências, em especial deficiência visual (MARTINS et al., 2014; ARRUDA et al., 2015; FARIA et al., 2016).

Embora o conceito de adesão medicamentosa possa não ser unânime, no geral consiste no uso de pelos 80% do que foi prescrito. Tomar dose errada, fora do horário, quantidade diferente do prescrito e etc, são formas de não adesão (LEITE; VASCONCELLOS, 2003, p.777).

Dentro desse contexto, os idosos ganham uma importância ainda maior segundo ARRUDA et al. (2015), já que 50% dos usuários que usam medicamentos são constituídos de idosos, em virtude do processo natural da terceira idade, onde há mais carga de doenças e incapacidades. Seguindo esse raciocínio, MARTINS (2014) frisa que pessoas com deficiência, precisam de cuidados e proteção em atividades rotineiras como uso de medicamentos, o que gera um desafio já que o consumo é sempre maior quando comparados em relação a pessoas não deficientes. A causa do aumento do consumo está relacionada à saúde da pessoa e a maior presença de comorbidades (CASTRO et al., 2010). Portanto, estudos buscam encontrar as causas e possíveis caminhos que amenizam ou ajudam a tornar possível uma rotina aderente ao tratamento de saúde para idosos e pessoas com deficiências (ARRUDA et al.,2015).

MARTINS et al. (2014) coloca o profissional farmacêutico como um elemento importante para aumentar a adesão a rotina de medicamento e FARIA et al. (2015) comenta que as estratégias podem começar dentro das Unidades Básicas de Saúde (UBS) e a forma como trabalham e se organizam no sentido de identificar os problemas e traçar metas para resolvê-las.

Interessante notar que embora haja esforço em encontrar as causas e propor soluções, não houve, nas referidas pesquisas acima, proposições que apontem uma estratégia que envolva o uso da tecnologia como equacionador nas diversas propostas. TAVARES (2012) coloca alguns fatores para isso, entre eles, redução da

6

capacidade de memória de curto termo, acuidade visual, audição e dificuldade dos idosos no uso das novas tecnologias. Essa circunstância possui caráter mais complexo quando a pessoa tem alguma deficiência.

Então, embora as tecnologias devessem mitigar as desigualdades e exclusões sociais, há desafios que necessitam ser vencidos, entre eles, o custo de acesso às tecnologias, a falta de habilidades para uso de dispositivos digitais, a carência de disponibilidade das tecnologias e as barreiras de acessibilidade quando se vai a um sítio Web. Essas barreiras podem tornar tecnologias digitais inacessíveis, comprometendo o bem-estar das pessoas e a garantia de seus direitos fundamentais (SÃO PAULO, 2020).

Tem havido progresso em leis, além da Constituição, amparada pelo Estatuto do Idoso, sancionado no ano de 2003. Um dos pontos relevantes, por exemplo no Estatuto do Idoso, é a preocupação com a interação do idoso com a vida moderna, com ênfase na necessidade de interação com os aparatos tecnológicos, para que este público tenha mais autonomia ao executar tarefas cotidianas, como por exemplo: usar um caixa eletrônico sem a necessidade de ser assessorado por um atendente (TAVARES, 2012).

É comum que parte da população idosa e por extensão com deficiência tenha no decorrer de sua vida dificuldades de ordem cognitiva, motora, talvez financeira, que podem ser fatores limitantes ao acesso às novas tecnologias.

A acessibilidade também é uma alternativa a ser usada na interação humano computador. A tecnologia assistiva e sua aplicação para a terceira idade e para deficientes é apresentada como uma forma de superar as limitações (TAVARES 2012).

Não se tem a intenção de classificar o idoso como um deficiente, mas sim propor a discussão de que tanto o idoso quanto a pessoa com deficiência, deve requerer e exercer seus direitos, para garantir que sua voz e vontade sejam ouvidas.

Pesquisas revelam que a barreira dos idosos e os novos recursos tecnológicos está diminuindo a cada dia no Brasil (PAIVA, 2016; COSTA, 2018), e no restante do mundo (ANDERSON; PERRIN, 2017).

7

No artigo de KOCUREK (2009), para cada fator que impede adesão à rotina medicamentosa existem algumas estratégias que podem ser eficazes a fim de superá las. Entre elas está a simplificação das rotinas medicamentosas, a listagem de medicamentos por parte do paciente, a clareza nas instruções de uso dos medicamentos, o uso de lembretes, o uso de porta comprimidos e por fim o acompanhamento da adesão do paciente por parte do agente de saúde.

Outros autores destacam o uso de estratégias que reforcem a memorização, o que pode contribuir para uma maior adesão à terapêutica. Segundo Werlang et al. (2008, p.97)

“Quando uma pessoa relembra informações aprendidas no passado, está executando uma tarefa de memória retrospectiva, e quando relembra a execução de uma ação que será desenvolvida no futuro, refere-se à memória prospectiva”

Então as caixas para acondicionamento diário dos comprimidos, anotações, calendários, seleção prévia dos medicamentos que devem ser tomados diariamente, cartelas de comprimidos com calendários e uso de despertador e esquemas mentais são boas estratégias de memorização reforçando de forma positiva a adesão ao uso correto de medicamentos (WERLANG et al. 2008, p.99).

No mercado nacional existem soluções em projetos e/ou aplicativos que buscam melhorar a relação de pessoas com seus tratamentos medicamentosos, principalmente nos casos onde há uma rotina diária ou semanal de uso de determinados medicamentos.

Uma dessas soluções é o projeto de Grayce Miguel França denominado "Farmácia para Cego ver” onde usa a tecnologia de QR Code. Através de um recurso de relevo nas embalagens de medicamentos e bulas é possível pessoas com deficiência visual, idosos e daltónicos terem a identificação dos medicamentos e bulas, facilitando o uso dos fármacos (CFF, 2022).

Outro projeto desenvolvido por pesquisadores da Universidade Federal da Paraíba (UFPB) é um equipamento chamado ping o qual emite um bipe a cada gota que atravessa o equipamento auxiliando na contagem para pessoas com deficiência

8

visual, sejam elas cegas, com baixa visão, idosos ou qualquer indivíduo que tenha dificuldade visual para administrar medicamentos em gotas. (CONTA-GOTAS, 2020).

Há aplicativos que têm a proposta de incluir uma pessoa para auxiliar em especial pessoas com perda de visão. Um deles chama-se Be My Eyes. Através de uma chamada de vídeo, voluntários dão auxílio visual para pessoas cegas e com visão limitada, em situações que vão desde combinar cores até checar se as luzes estão acesas ou preparar o jantar. O aplicativo é gratuito e está disponível em para ambos IOS e Android.

Considerando o que foi exposto, identificamos que há a necessidade do desenvolvimento de softwares que auxiliem a demanda da crescente terceira idade e pessoas com deficiência, já que os projetos propostos ainda são incipientes ou não são apontados como solução para problemas relacionados com a não adesão

medicamentosa. É com base nisso que este trabalho propõe o desenvolvimento de um software que preencha essa lacuna.

9

**3 FUNDAMENTAÇÃO TEÓRICA**

*Software* é uma entidade que está em ambientes diversos, provocando mudanças e sendo mudado pelo ambiente a partir dos resultados produzidos; parafraseando a definição da palavra “retroalimentação”:

“Ação ou efeito de retro-alimentar (re-alimentar); realimentação. Eletrônica. Ação de enviar elementos da saída de um sistema em direção a sua entrada, visando o controle e/ou reforço. Todo processo que esteja envolvido com o controle de um sistema, através do conhecimento do resultado de suas respostas. [Por Extensão]. As respostas encontradas nesse processo” (DICIO, 2023).

As mudanças ocorrem por necessidade de corrigir erros existentes no *software* ou de adicionar novas funcionalidades. Essa necessidade evolutiva do sistema de *software* o torna ‘não confiável’ e predisposto a defeitos, atraso na entrega e custos acima do estimado. Ao mesmo tempo, o crescimento em tamanho e complexidade dos sistemas de *software* exige que os profissionais da área raciocinem, projetem, codifiquem e se comuniquem por meio de componentes de *software*. Como resultado, qualquer concepção ou solução de sistema passa então para o nível arquitetural, onde

o foco recai sobre os componentes e relacionamentos entre eles num projeto arquitetural de *software* (SOMMERVILLE, 2011).

O projeto de arquitetura está preocupado com a compreensão de como um sistema deve ser organizado e com a estrutura geral desse sistema. No modelo do processo de desenvolvimento de *software*, o projeto da arquitetura é o primeiro estágio no processo de projeto de *software*. É o elo crítico entre o projeto e a engenharia de requisitos, pois identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles (ASTUDILLO, 1998). O resultado do processo de projeto de arquitetura é um modelo de arquitetura que descreve como o sistema está organizado em um conjunto de componentes de comunicação (GARLAN; SHAW, 1996).

10

Padrões de Projeto ou *design patterns* tiveram sua origem, segundo Busso (2006), nos trabalhos de arquitetura objetivando padrões utilizados nas construções e nas cidades.

Os padrões de projeto permitem analisar uma solução de arquitetura num nível de abstração elevado, sem a necessidade de se detalhar sua implementação. Conforme Busso (2006), as práticas já aceitas pela comunidade de engenharia de software e padrões de projeto já testados no processo de desenvolvimento reduziram a quantidade de trabalho de reescrita e reprojeto nos estágios finais de desenvolvimento.

Desse modo, o termo padrão de projeto pode expressar uma situação reutilizável descrita por três componentes, sendo um contexto, um problema e uma solução. Portanto, o uso de padrões de projeto é essencial para reutilizar soluções e arquiteturas bem-sucedidas a fim de construir *softwares* que tenham flexibilidade e fácil manutenção em face de mudanças. O uso de padrões de projeto pode reduzir a complexidade do processo de projetar *software*. Além disso, o *software* orientado a objetos bem projetado possibilita aos projetistas reutilizar e empregar componentes preexistentes em sistemas futuros. Com o uso dos padrões de projetos maximizam os benefícios existentes na arquitetura proposta (GAMMA et al., 1995).

O Projeto de Arquitetura do *Software* do aplicativo AUM utiliza o JSON para comunicar-se com o repositório de dados e as diversas camadas da arquitetura são expostas por uma *Application Programming Interface* (API) que tem a finalidade de implementar o projeto de *software*.

3.1. *APPLICATION PROGRAMMING INTERFACE*

A enciclopédia nos diz que ‘API’ é um acrônimo para *Application Programming Interface* (API), em uma definição formal, está relacionado a um conjunto de rotinas e padrões estabelecidos por um *software* para a utilização das suas funcionalidades por outros aplicativos.

Uma outra perspectiva interessante é sugerida por KONSTANTINOV (2022) quando diz que “uma ação básica como visualizar uma página da web necessita de centenas ou até milhares de APIs diferentes funcionando corretamente. As

11

tecnologias modernas da Internet”, acrescenta ele, “simplesmente não poderiam existir sem essas toneladas de API funcionando bem”.

Uma API é um contrato formal para conectar diferentes contextos programáveis. Um exemplo interessante dado por KONSTANTINOV (2022) é comparar uma API bem projetada com a foto de um aqueduto romano, visto na Figura 2.

Figura 2 - Aqueduto romano



Fonte: KONSTANTINOV, 2022.

Estendendo o conceito sugerido na Figura 2, API pode interligar e integrar dois sistemas, em que um deles fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade de algum dos sistemas conhecer detalhes de implementação do outro. Em outras palavras, é uma forma bem segura pela qual dois aplicativos trocam dados.

3.2. FORMATO DE DADOS *JAVA SCRIPT OBJECT NOTATION (JSON)*

*JavaScript Object Notation* (JSON) é um modelo para armazenamento e transmissão de informações no formato texto (ECMA-404, 2017). Embora simples, tem sido amplamente utilizado por aplicações devido a sua qualidade de organizar

12

informações de uma forma bem mais reduzida, tornando mais rápido a análise dessas informações.

Os objetos são definidos entre chaves e podem ser compostos por múltiplos pares de nome/valor, por arrays e também por outros objetos (ECMA-404, 2017). Desta forma, um objeto JSON pode representar, virtualmente, qualquer tipo de informação. O exemplo da Figura 3 mostra a representação dos dados de uma pessoa física.

Figura 3 - Objeto JSON representando dados de uma pessoa física



Fonte: SAUDATE, 2013.

Sobre a composição do objeto é significativo explicar:

● Chave: corresponde ao identificador do conteúdo. Por isso, deve ser uma *string* delimitada por aspas;

● Valor: representa o conteúdo correspondente e pode conter os seguintes tipos de dados: *string, array, object, number, boolean* ou *null*.

Consequentemente é uma sintaxe de texto que facilita o intercâmbio de dados estruturados entre a maioria das linguagens de programação (CORDEIRO, 2014).

13

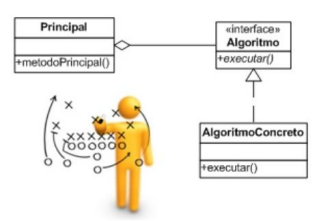
3.3. PADRÕES DE PROJETO

Resumidamente pode-se entender como padrão de projeto, como a solução recorrente para um problema em um contexto, mesmo que em projetos e áreas distintas (BEDER 2001). A seguir há um pequeno glossário dos padrões de projetos utilizados durante o desenvolvimento do aplicativo AUM para satisfazer a arquitetura de *software* proposta.

**3.3.1. *Strategy***

O *Strategy* é um padrão que deve ser utilizado quando uma classe possuir diversos algoritmos que possam ser utilizados de forma intercambiável. A solução proposta pelo padrão consiste em delegar a execução do algoritmo para uma instância que compõe a classe principal. Dessa forma, quando a funcionalidade for invocada, no momento de execução do algoritmo, será invocado um método da instância que a compõe (GAMMA et al. ,1994). A Figura 4 apresenta um diagrama que mostra a estrutura básica do padrão.

Figura 4 - Estrutura do Strategy



Fonte: GUERRA, 2012

A utilização desse padrão é realizada nos construtores de classe com o uso de fábrica de objetos. As fábricas de objetos recebem um parâmetro especificando que objeto deve ser criado para atender aquela especificidade. Um exemplo prático é uma

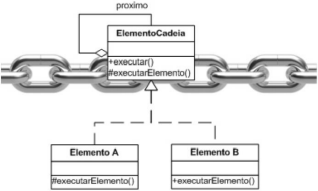
14

fábrica de objetos recebendo o parâmetro de entrada para criar um objeto que atenderá usuários de acordo com o perfil do usuário.

**3.3.2. *Chain of Responsibility***

O *Chain of Responsibility* é um padrão de projeto que cria uma cadeia de execução na qual cada elemento processa as informações e em seguida delega a execução ao próximo da sequência. Em sua implementação tradicional, os elementos são percorridos até que um deles faça o tratamento da requisição, encerrando a execução depois disso. Como alternativa, também é possível criar uma cadeia de execução onde cada um executa sua funcionalidade até que a cadeia termine ou ela seja explicitamente finalizada por um dos elementos (GAMMA et al. ,1994). A Figura 5 apresenta um diagrama que mostra a estrutura básica do padrão.

Figura 5-Estrutura do Chain of Responsability



Fonte: GUERRA, 2012

**3.3.3. *Proxy***

O *Proxy* é um padrão de projeto estrutural que permite que programador forneça um substituto ou um espaço reservado para outro objeto. Um *proxy* controla o acesso ao objeto original, permitindo seja feito algo ou antes ou depois do pedido chegar ao objeto original (GAMMA et al. ,1994).

A ideia básica desse padrão é criar uma classe que envolve uma outra do mesmo tipo. Dessa forma, ela pode ser passada de forma transparente como se fosse

15

a classe original para quem a irá utilizar. A Figura 6 apresenta um diagrama que mostra a estrutura básica do padrão (PINHEIRO, 2015).

Figura 6 - Envolvendo um objeto com o padrão Proxy



Fonte: GUERRA, 2012

**3.3.4. *Data Access Object***

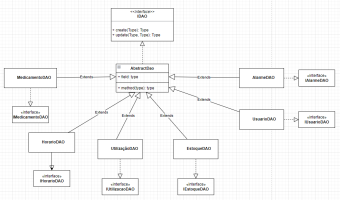
O *Data Access Object* (DAO) consiste em separar e abstrair o mecanismo de persistência utilizado na aplicação. O padrão de projeto DAO é descrito em detalhes por ALUR et al (2001, p. 290), embora seja somente um caso particular do padrão de projeto *Strategy.*

A adoção deste padrão abstrai e encapsula os mecanismos de acesso a banco de dados, o que contribuirá para facilidade na manutenção do aplicativo e aumentando a vida útil desse.

Segue, na Figura 7, como foi desenhado a estrutura de acesso a dados do Aplicativo AUM.

16

Figura 7 - Estrutura de acesso a dados do Aplicativo AUM



Fonte: Autor

Na estrutura da Figura 7, vemos alguns componentes que compõem a camada de acesso. Alguns detalhes foram suprimidos para evitar excesso de conteúdo. Esse modelo é bem comum no uso do Padrão *DAO*. Alguns elementos utilizados:

● Interface IDAO

○ Abstrai todos os métodos gerais do modelo

● AbstractDao

○ Classe que dá suporte para um elemento que chamamos de entidade, o qual representa um mapeamento de uma tabela de banco de dados. ● "Nome Entidade" + sufixo DAO

○ Classe de Implementação para acesso aos dados da tabela

● I+"Nome Entidade" + Sufixo DAO

○ Interface que deve ser utilizada nas chamadas da camada de acesso de dados ao executar comandos como *CREATE* e *UPDATE*.

17

3.4. BANCO DE DADOS *FIREBASE*

Segundo Korth e Silberschatz (1994), um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, ou seja, sempre que for possível agrupar informações que se relacionam e tratam de um mesmo assunto, podemos dizer que temos um banco de dados.

Pode ser exemplificado situações clássicas como uma lista telefônica, um catálogo de CDs ou um sistema de controle de RH de uma empresa. Já um sistema de gerenciamento de banco de dados (SGBD) é um *software* que possui recursos capazes de manipular as informações do banco de dados e interagir com o usuário. Exemplos de SGBDs são: Oracle, SQL Server, DB2, PostgreSQL, MySQL, entre outros.

Existem dois tipos de banco de dados que são utilizados atualmente.

● Banco de dados relacional – Os dados são armazenados em tabelas com linhas e colunas específicas que definem os dados.

● Banco de dados não relacional – Segue uma linguagem NoSQL. Dessa forma, os campos não são limitados igual ao SQL. Os dados são alocados em pastas, fator que possibilita a definição de um esquema personalizado, além de permitir a adição de novas propriedades sem que impacte as outras informações armazenadas.

Com o advento e crescimento da web e de novas soluções desenvolvidas através do avanço tecnológico baseado em sistemas distribuídos, um enorme volume de informações passou a ser gerado por pessoas e entidades em todo o mundo. Todavia, constatou-se que os modelos de bancos de dados relacionais popularmente utilizados, apresentam limitações ao trabalhar com grandes volumes de dados. Consequentemente, surgiu a necessidade de criar um modelo de banco de dados dotado de escalabilidade, capaz de manipular uma crescente quantidade de dados de maneira uniforme. A partir de então, de acordo com estudos realizados sobre bancos de dados distribuídos e possíveis melhorias para alcançar maior nível de escalabilidade, e alta disponibilidade, novas aplicações não-relacionais foram desenvolvidas criando uma nova tendência chamada de NoSQL. Embora o aplicativo AUM apresente uma estrutura de poucas pastas ou *“nodes”*, esse pode se beneficiar

18

da disponibilidade e escalabilidade oferecida por um banco NoSQL. Além do que não tem custo nenhum no uso de até 5GB de Armazenamento. (CASA NOVA; MOURA, 2014).

O AUM utiliza como repositório de dados o *Firebase*. O *Firebase Realtime Database* é um banco de dados NoSQL hospedado na nuvem. Os dados são armazenados em formato JSON e sincronizados em tempo real para cada cliente conectado. Todos os usuários compartilham uma instância do *Realtime Database* e recebem automaticamente atualizações com os dados mais recentes.

Quando integrado ao *Firebase Authentication*, os desenvolvedores podem definir quem tem acesso a quais dados e como eles podem acessá-los. O *Realtime Database* é um banco de dados NoSQL e, como tal, possui otimizações e funcionalidades diferentes em comparação com um banco de dados relacional. A API do *Realtime Database* foi projetada para permitir apenas operações que podem ser executadas rapidamente. Isso permite que seja criado uma ótima experiência em tempo real que pode atender a milhões de usuários sem comprometer a capacidade de resposta.

Outro ponto a considerar é que não há o encadeamento de tabelas (Figura 8). Essa decisão foi tomada para melhorar a performance conforme recomendado na página do Firebase. Por isso o banco do Aplicativo AUM é plano.

Figura 8 - Estrutura em nó do banco de dados



Fonte: Autor

O *Firebase Realtime Database* permite aninhar dados em até 32 níveis de profundidade, não sendo a estrutura padrão. No entanto, quando se busca dados em um local de banco de dados, há também recuperação de todos os seus nós filhos.

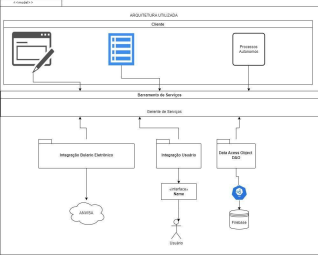
19

Além disso, quando existe concessão do acesso de leitura ou gravação em um nó em um banco de dados, existe também concessão de acesso a todos os dados desse nó. Portanto, na prática, é melhor manter sua estrutura de dados o mais plana possível.

3.5. ARQUITETURA UTILIZADA

O aplicativo AUM, produto final do presente trabalho, destina-se a auxiliar a utilização de medicamentos por grupos de pessoas, tais como: idosos e pessoas com deficiência. Para atender a esse público, a concepção do AUM deverá incorporar soluções que interajam fácil e adequadamente com esse grupo de pessoas. Para tanto, o projeto e construção do aplicativo AUM envolveu a utilização de conceitos e estratégias de engenharia de *software*, as quais serão detalhadas a partir da Figura 9.

Figura 9 - Visão geral da concepção do aplicativo AUM*.*

**

Fonte: Autor

Através da Figura 9 pode-se observar os principais elementos do AUM, ou seja:

20

● Cliente

● Barramento de Serviços

● Interface de Bulário Eletrônico

● Interface de Integração de Usuário

● Interface de Acesso ao Banco de Dados

Esses elementos são descritos da forma como segue:

**3.5.1. Cliente**

Objetivamente falando é qualquer interface ou recurso utilizado pelo usuário para consumir ou solicitar um serviço do aplicativo. Pode ser um formulário ou uma lista, bem como um serviço autônomo monitorando horários. Os tipos de clientes relacionados com o AUM são indicados na Figura 9 e detalhados nas Figura 10 à Figura 12.

A Figura 10 representa janelas de entrada de dados no AUM, como: cadastro de usuário/cadastro de medicamento ou cadastro de perfil.

Figura 10 - Entrada de Dados



Fonte: Autor

As janelas com as listas de dados no AUM, como: a lista de medicamentos, lista de horários, lista de pesquisa da ANVISA, são representadas pela Figura 11.

Figura 11 - Listas de Pesquisa



Fonte: Autor

Na Figura 12 estão representadas as rotinas no AUM onde não existem interações do usuário, como por exemplo: a rotina que verifica um alarme.

21

Figura 12 - Processos Autônomos



Fonte: Autor

**3.5.2. Barramento de Serviços**

A indicação do barramento de serviços indicada na Figura 9, objetiva criar uma interface de acesso a todos os recursos do aplicativo. Para essa tarefa, utilizamos um padrão de projeto conhecido como “Facade”. O padrão “Facade” (pronuncia-se "façade” por ser uma palavra de origem francesa) propõe a criação de uma classe intermediária que serve como uma fachada para que o cliente possa acessar as funcionalidades desejadas. Essa classe encapsula a complexidade da interação entre os diversos componentes e desacopla o cliente das implementações (GUERRA, 2012). A Figura 13 mostra um barramento de serviços padrão com a representação da interface de barramento de serviços padrão entre as requisições vinda dos clientes e os serviços disponíveis. As setas indicam o fluxo das mensagens de envio / resposta (IBM CLOUD EDUCATION, 2019).

Figura 13 - Representa o barramento de serviços



Fonte: Autor

**3.5.3. Interface de Integração Bulário Eletrônico**

Sempre se referenciando à Figura 9, a interface de integração bulário eletrônico é o componente que possibilita criar uma ponte entre o serviço e as consultas às fontes de dados de medicamentos. No presente trabalho, utilizamos um padrão de projeto chamado *Strategy* com a finalidade de abstrair a fonte de dados que utilizamos. A ideia básica é impedir que o aplicativo conheça os detalhes da consulta de

22

medicamento ao ponto de, em uma situação de troca de fonte de dados, tal mudança não gerar nenhum tipo de problema para o restante do projeto (GUERRA, 2012). A Figura 14 representa graficamente a interface bulário eletrônico.

Figura 14 - Componente de Integração de Bulário



Fonte: Autor

A Figura 14 exemplifica graficamente o componente *container* de integração entre qualquer fonte de dados de bula e o barramento de serviços. Adicionalmente, esse componente mostra o isolamento do acesso e permite a reutilização dos diversos recursos de busca da lista de medicamentos, tais como: download de arquivo e conversão de pdf para o formato texto. Já o armazenamento de uma fonte de dados de bulas presente na Figura 9 é mostrado na Figura 15 onde, a nuvem que circunda a palavra ANVISA, indica que está acessível pela internet.

Figura 15 - Serviço de Bulário Eletrônico da Anvisa na nuvem



Fonte: Autor

**3.5.4. Interface de Integração de Usuário**

Seguindo o mesmo padrão *Strategy*, abordado no tópico “Integração de Bulário Eletrônico", é necessário criar recursos específicos para os tipos de usuários. Por exemplo, no caso de termos uma interface para pessoas com visão reduzida ou mesmo sem visão. Dessa forma, formatamos uma lista das interfaces da seguinte forma:

1. Usuário comum

a. Qualquer usuário que não fizer uso de recurso de acessibilidade como comandos por voz.

2. Usuário PcD

a. Usuário com perda de visão ou reduzida

b. Usuário com perda auditiva

23

c. Usuário com TEA

3. Usuário Idoso

Utilizando a interface de integração de usuário podemos, por exemplo, avisar que está na hora de um medicamento abstraindo a forma que será a saída do aviso. Para alguém com perda de visão o aviso será sonoro. No entanto, para alguém com perda auditiva utilizamos a vibração do celular e notificações. Essa ação é representada pela Figura 16.

Figura 16 - Componente de Integração de Usuário



Fonte: Autor

A Figura 16 exibida acima representa o componente *container* de integração entre qualquer perfil de usuário e o barramento de serviços. O acesso ao container de usuário deve ser feito através de uma interface (Figura 17), onde a implementação para cada perfil abre possibilidade da aplicação atender aos mais diversos tipos de perfis de usuários.

Figura 17 - Interface de Conexão com os perfis



Fonte: Autor

A Figura 18 representa qualquer perfil implementado suportado pela interface, ou seja, representa um ator qualquer relacionado ao perfil de usuário. Por exemplo, podemos ter usuário com visão reduzida ou um usuário comum.

Figura 18 - Representa qualquer perfil suportado pela interface



Fonte: Autor

**3.5.5. Interface de Acesso a Dados**

O componente de acesso de dados (Figura 19) além de abstrair o uso da API do *Firebase*, possibilita que o usuário tenha acesso ao aplicativo em qualquer

24

dispositivo instalado, em virtude do Banco de Dados estar remoto. Utilizamos o padrão de projeto “DAO”. Esse padrão possibilita a troca do banco de dados, com a simples substituição da fábrica de implementação (ALUR et al,2001, p. 290).

Figura 19 - Componente de Acesso ao Banco de Dados



Fonte: Autor

O Banco de dados *Firebase* (Figura 20) foi escolhido em razão da simplicidade, gratuidade, escalabilidade, segurança e fácil integração da API com Android Studio®.

Figura 20 - Banco de Dados Remoto Firebase



Fonte: Autor

25

**4 DESENVOLVIMENTO**

O aplicativo AUM foi concebido de forma a auxiliar a utilização de medicamentos por grupos de pessoas como idosos e portadores de deficiência. Este capítulo destina-se, portanto, a reportar as etapas envolvidas no desenvolvimento do trabalho,

4.1. O APLICATIVO

**4.1.1. Primeiro Acesso**

No primeiro acesso do aplicativo é apresentado ao usuário a opção de cadastrar um novo usuário ou efetuar o acesso informando o e-mail e senha (Figura 21).

Figura 21 - Janela de Acesso ao Aplicativo AUM



Fonte: Autor

Se for primeiro acesso o usuário poderá efetuar o cadastro clicando no botão *CADASTRE-SE*, onde será exibido o formulário de cadastro de usuário (Figura 22).

26

Figura 22 - Cadastro de Usuário



Fonte: Autor

Se o usuário possuir um cadastro feito anteriormente escolherá a opção efetuar acesso (Figura 23).

27

Figura 23 - Autenticação de Usuário



Fonte: Autor

Tanto no primeiro caso como no segundo caso o aplicativo exibirá a janela principal (Figura 24).

28

Figura 24 - Janela Principal



Fonte: Autor

Quando o usuário já tiver feito acesso pelo menos uma vez, o aplicativo utilizará o recurso *Firebase Auth*, do serviço de banco de dados *Firebase*, para autenticar o usuário.

O objetivo do *Firebase Authentication* é facilitar o desenvolvimento de um sistema de autenticação seguro, além de melhorar a experiência de acesso e integração para os usuários. Ele oferece uma solução de identidade completa, compatível com contas de e-mail/senha, autenticação por telefone, acesso do Google, Twitter, Facebook e GitHub. O acesso por e-mail e senha foi escolhido por apresentar simplicidade e os e-mails serem únicos em qualquer servidor de e-mail.

Dito isto, sempre que o usuário acessar o aplicativo fará acesso automático apresentando facilidade de uso desde a tela inicial.

Nesta primeira versão do aplicativo AUM, o usuário sem necessidade de auxílio para acessibilidade também poderá utilizar todas as funcionalidades. O usuário idoso

29

e/ou com perda de visão ou reduzida terá, através do uso da voz, acesso a algumas funcionalidades. Para usuários surdos/mudos e TEA ainda não foram adaptadas funcionalidades especiais para esse público.

O perfil é o elemento chave na comunicação entre as camadas do aplicativo. O aplicativo determina através do perfil a forma como os comandos devem ser respondidos e para sustentar esse recurso, utilizamos padrões de projetos. O principal padrão de projeto adotado foi o *Strategy*.

Quando o aplicativo inicia pelo cadastro inicial, este identifica qual é o perfil do usuário. Uma variável global com o nome "integração usuário" então é inicializada por uma fábrica de objetos. Para exemplificar o uso desse recurso, quando o aplicativo dispara um alarme indicando que um medicamento deve ser tomado, isso é feito em duas etapas: através de notificação e por áudio. Isto é necessário para atender quem não pode ler uma notificação. Então o aplicativo fará uma chamada para "integração usuário" que dispara um alarme em áudio. A mesma chamada existe para o usuário comum, mas como não tem utilidade para ele, este não fará nada. Então cada perfil tem uma série de funcionalidades que serão chamadas de acordo com o perfil de usuário, com isso, teremos Integração Usuário Comum ou Integração Visual Reduzida e etc.

O fragmento de código (Figura 25) mostra o uso de uma fábrica de objetos para ser utilizado conforme o perfil.

30

Figura 25 - Fragmento Responsável pela criação da Interface de Pesquisa do Bulário Eletrônico 

Fonte: Autor

Conforme observado na classe *FactoryIntegracaoUsuario* (Figura 25)*,* quando o método *construtor createIntegracaoUsuario* é chamado, um parâmetro com o tipo de perfil é passado. Após considerar que tipo de perfil foi informado, a classe construirá um objeto que atenderá o perfil conforme suas especificações internas delineadas na classe.

Nesse ponto, o padrão de projeto *Strategy* (Figura 4) será importante conforme observamos nos dois fragmentos de código abaixo. O primeiro fragmento dá suporte ao perfil de usuário sem necessidade de acessibilidade.

31

Figura 26 - Fragmento de um método de usuário sem necessidade de acessibilidade 

Fonte: Autor

Na imagem acima (Figura 26), ao atualizar o saldo de medicamentos uma mensagem é exibida na tela.

O segundo fragmento (Figura 27) dá suporte ao usuário idoso e/ou com visão reduzida/perda.

32

Figura 27 - Fragmento de um método que trata usuário idoso e/ou com visão reduzida/perda 

Fonte: Autor

Na imagem acima (Figura 27), ao atualizar o saldo de medicamentos um áudio é reproduzido. Assim, constatamos o uso adequado do padrão *Strategy*, onde o comportamento do aplicativo é ajustado com base no usuário que acessa.

Para dar suporte a estrutura mencionada acima foi necessário dividir o projeto em uma arquitetura de camadas. Na Figura 28, temos uma imagem que exibe a estrutura da camada de API-integração.

33

Figura 28 - Estrutura de Pacotes da Integração Bulário Eletrônico



Fonte: Autor

Na Figura 28 é apresentada uma estrutura de pacotes organizados de uma forma que expressa claramente uma divisão de papel. O pacote API é a entrada de qualquer implementação tanto para o acesso a bulário eletrônico quanto integração de usuário. Neste pacote estão as especificações de como a API deve responder para cada caso. O restante dos pacotes são os elementos necessários para a construção desse acesso. Temos o pacote *factory* responsável pela criação de objetos que representam os elementos da API. O pacote interfaces estará contido nas interfaces que são os contratos de utilização das APIs.

**4.1.2. Janela Principal**

O objetivo do presente trabalho é ajudar pessoas com alguma dificuldade em fazer uso de medicamentos e nesta primeira versão serão atendidos idosos e/ou pessoas com visão reduzida/perda. Assim, foram desenvolvidos alguns comandos por voz. Partimos da premissa que para o perfil de visão reduzida/perda será necessário

34

um cuidador para fazer algumas etapas preliminares as quais iremos abordar em tópico anexo.

**4.1.3 – Para o perfil visão reduzida/perda**

A partir da tela principal (Figura 29), comandos por voz estão disponíveis se o perfil do usuário tiver suporte para o recurso.

Figura 29 - Janela principal com a Lista de Medicamentos



Fonte: Autor

Após o usuário tocar em qualquer área da tela principal, comandos por voz estarão disponíveis (Figura 30). Com o toque, o aplicativo informa que o usuário poderá falar o comando reproduzindo um pequeno bip. Após escutar o comando falado, o aplicativo informa que a captura foi encerrada e então processa a voz:

35

Figura 30 - Representa os comandos por voz



Fonte: Autor

**4.1.4. Comandos Disponíveis**

O AUM falará os comandos por voz disponíveis

● **Lista de Medicamentos**

O AUM lerá as informações principais de cada medicamento previamente cadastrado

● **Detalhe do Medicamento “apelido”**

O AUM localizará na lista o medicamento que corresponde ao apelido cadastrado. Após localizar o medicamento exibirá o detalhe lendo as principais informações do medicamento e informando que está na janela de detalhe do medicamento.

● **Administrar**

O AUM trocará o perfil do usuário temporariamente possibilitando que o cuidador faça a administração dos dados cadastrados. Entre esses itens estão: cadastrar medicamentos, cadastrar horários, cadastrar compras ou informar uma redução na quantidade de medicamentos disponíveis. Poderá alterar o apelido do medicamento entre outros dados. Após efetuar a mudança basta clicar no botão “Restaurar perfil”.

● **Sair**

O AUM será fechado.

Para termos acesso a todos os recursos de voz, foi necessário utilizar alguns recursos importantes relativos à plataforma Android, bem como fundamentos do

36

desenvolvimento móvel. O recurso de reconhecimento de voz, reprodução de voz e mesmo o uso do *Firebase* são normalmente assíncronos.

Devido a essa característica precisamos “assinar” métodos para receber as respostas ao final da execução. Para isso foi necessário o uso do recurso de *callbacks*.

Trata-se de uma função que é executada quando algum evento acontece ou depois que algum código chega ao estado desejado. Também conhecido como função de retorno, o *callback* cria regras dentro de outras funções para que sejam utilizadas no futuro. Normalmente, ele age de forma assíncrona, ou seja, não é executado imediatamente. A aplicação continuará rodando enquanto espera o momento certo da sua execução.

O *callback* é muito comum na linguagem Java, por exemplo, durante a busca de dados ou reprodução de mídia. Isso porque permite ao programador especificar o que deve ocorrer quando a execução acabar.

As notificações (ou “**callbacks**”) permitem que o aplicativo seja informado quando uma transação tiver seu *status* alterado. Dessa forma, podemos identificar quando um dado solicitado já foi devolvido pelo banco de dados *Firebase* ou quando uma captura de voz já foi obtida.

Para fazer uso deste importante recurso na programação orientada a objeto foi necessário fazer uso massivo de interfaces. Alguns fragmentos de código (Figuras 31 a 33) mostram o uso de um desses esquemas.

Nos fragmentos das Figura 31 a Figura 33, precisamos destacar os elementos chaves: Gerente Serviço *Lester* e Constantes que determinam a entrada do comando e ação resultado.

37

Figura 31 - Fragmento que executa o comando de Voz de Retorno da Lista de Medicamentos 

Fonte: Autor

Na Figura 31, o trecho ocorre após o usuário tocar na tela e falar um comando. Quando o usuário termina de falar, o aplicativo aguarda 5 segundos e então encerra a captura e executa o comando de voz para obter a lista de medicamentos.

Para que seja possível o aplicativo devolver para a atividade ou tela o resultado da consulta da lista de medicamentos é necessário que a atividade tenha implementado a interface ouvinte *GerenteServicosListener* (Figura 32)*.*

38

Figura 32 - Fragmento da Definição da Classe Principal com implementação do Recurso de Voz 

Fonte: Autor

Na Figura 32 há suporte para o ouvinte quando a definição do formulário Principal Activity implementa a interface *GerenteServicosListener* que fará o papel do *callback*.

Então, com o recurso de *callback* conectando Tela <-> Ouvinte <-> Consulta de dados, a ação executa o comando da Figura 33.

39

Figura 33 - Fragmento que executa o método de integração de usuário para lista medicamentos 

Fonte: Autor

Quando o ouvinte termina a etapa, ele devolve o controle para a tela. A tela executa a ação que é reproduzir em áudio a lista de medicamentos. Então, o recurso de áudio lerá os itens que constam na lista de medicamentos exibida.

**4.1.5. Alarmes**

O Aplicativo fará uma verificação de minuto a minuto sobre horários ativos de medicamentos. Como o aplicativo realiza chamadas assíncronas do banco de dados Firebase, será necessário adotar um o padrão de projeto *Chain of Responsibility* (Figura 25) para criar uma cadeia de execução na qual cada elemento processa as informações e em seguida delega a execução ao próximo da sequência. Será necessário aninhar as chamadas de bancos de dados pois, para geração de alarme, precisaremos de: Horários Ativos, Utilizações realizadas e Registro de Alarmes.

40

Para diminuir o acoplamento, combinamos o padrão *Chain of Responsibility* com o padrão de projeto *Proxy* (Figura 6). A ideia básica é criar uma classe que envolve uma outra do mesmo tipo. Dessa forma, ela pode ser passada de forma transparente como se fosse a classe original para quem a irá utilizar.

No nosso caso, utilizaremos os dois padrões para desacoplar a interface das chamadas de banco de dados. Os fragmentos de código Java abaixo exibem o uso do padrão *Proxy*. O primeiro fragmento (Figura 34) apresenta a interface de transações responsável por representar o componente intermediário.

Figura 34 - Interface de Transações



Fonte: Autor

O segundo fragmento (Figura 36) apresenta a classe *Proxy* que implementa a interface transação. Ela adiciona uma funcionalidade à classe cliente do mesmo tipo.

41

Figura 35 - Classe que Implementa o Proxy de Transação



Fonte: Autor

No fragmento da Figura 36, vemos que quando o método *executar* é invocado, ele cria um *Thread* e executa dentro dela o método da classe concreta que foi passada no construtor do *Proxy* como transação cliente.

Dessa forma a classe *Proxy* abstrai e desacopla a classe cliente. Então o método *executar* é invocado. O terceiro fragmento (Figura 36), a parte final do fragmento, percebemos o encadeamento das classes responsáveis por buscar os dados e disparar o alarme.

42

Figura 36 - Implementação da execução do Proxy



Fonte: Autor

**4.1.6. Detalhes do Medicamento**

Após o usuário selecionar um medicamento da lista, será exibida a tela de detalhes dos medicamentos (Figura 37). Nela encontram-se as principais funcionalidades que atendem o uso do medicamento: Registro de Utilização do Medicamento, compra de medicamento e horário de medicamentos.

43

Figura 37 - Detalhes do Medicamento



Fonte: Autor

Primeiramente iremos fragmentar a tela em diversos pedaços e abordar cada funcionalidade e após isso discorrer como a camada de serviço interage com as funcionalidades. A Figura 38 é uma funcionalidade disponível somente se o apelido ou a cor ou a quantidade de medicamentos por embalagem for alterada.

Figura 38 - Alterar Medicamento



Fonte: Autor

Então, quando o usuário clicar no botão alterar, os dados alterados serão atualizados no banco de dados.

A funcionalidade da Figura 39 sinaliza ao aplicativo que uma dose foi tomada do medicamento atual.

Figura 39 - Sinalizar Dose Utilizada



Fonte: Autor

São feitas algumas críticas para tornar o processo válido:

44

● O saldo do medicamento precisa ser maior ou igual a quantidade de medicamento por dose.

● O horário precisa estar cadastrado e ativo. Veja a Figura 40 um exemplo de horário ativo de um medicamento.

Figura 40 - Horário Ativo de um medicamento



Fonte: Autor

Após a validação, o aplicativo obterá o último saldo e criará um novo registro de saldo com o abatimento da quantidade utilizada. Também fará um registro com o horário da utilização. Os dois fragmentos da tela (Figura 41 e Figura 42) mostram esse contexto.

O primeiro fragmento de tela (Figura 41) exibe o controle de saldo de estoque de um medicamento.

Figura 41 - Lista de Saldo de Estoque



Fonte: Autor

45

O segundo fragmento (Figura 42) de tela exibe o registro de uma utilização do medicamento.

Figura 42 - Lista de Utilizações



Fonte: Autor

Uma funcionalidade importante para usuários idosos é a cor de indicação do medicamento, que pode ajudar a selecionar facilmente qual é o medicamento. Ao clicar no botão da Figura 43, uma paleta simples de cores será exibida (Figura 44). A lista de cores é reduzida para tornar mais simples o processo.

Figura 43 - Exibir lista de Cores



Fonte: Autor

O usuário poderá apenas clicar em uma das cores e assim mudar a forma como é apresentada a linha na lista de medicamentos (Figura 29).

Figura 44 - Paleta de Cores



Fonte: Autor

46

A funcionalidade ao clicar no botão da Figura 45 tem o objetivo de dar entrada no estoque de medicamentos. Com base no que foi informado na caixa de texto “estoque” será adicionado no saldo atual.

Figura 45 - Entrada de Estoque



Fonte: Autor

Já a funcionalidade selecionada ao se clicar no botão da Figura 46, tem o objetivo de dar saída no estoque de medicamentos. Com base no que foi informado na caixa de texto “estoque” será subtraído do saldo atual.

Figura 46 - Saída no estoque



Fonte: Autor

A funcionalidade relacionada com o botão da Figura 47, tem a ação de abrir o cadastro de horário.

Figura 47 - Cadastro de Horário



Fonte: Autor

As funcionalidades relacionadas na Figura 37, em sua maioria, tem um correspondente para dar suporte através dos comandos por voz. Os comandos por voz estão disponíveis, representado pela Figura 48, após o usuário tocar em qualquer área da tela detalhe do medicamento. O aplicativo informa que o usuário poderá falar

47

o comando reproduzindo um pequeno bip. Após escutar o comando falado o aplicativo informa que a captura foi encerrada e então processa a voz:

Figura 48 - Representa os comandos por voz



Fonte: Autor

**4.1.7. Comandos Disponíveis**

O AUM emitirá os seguintes comandos por voz:

● **Horário do Medicamento**

O AUM fará a leitura dos dados principais do último horário cadastrado, se houver. Caso não exista, o usuário receberá a informação também por voz.

● **Remédio Utilizado**

O AUM executará a mesma funcionalidade que o botão **Utilização**. Porém, ela informará o resultado da ação por voz para o usuário ter certeza que utilizou o medicamento certo.

● **Estoque Atual**

O AUM informará a quantidade do medicamento que ainda existe na embalagem ou no estoque.

● **Entrada “quantidade”**

O AUM executará a mesma funcionalidade do botão +. Porém informará o resultado da ação por voz para o usuário ter certeza que deu entrada no remédio certo. A palavra quantidade é a máscara para representar o número informado de entrada.

● **Saída “quantidade”**

O AUM executará a mesma funcionalidade do botão +. Porém informará o resultado da ação por voz para o usuário ter certeza que deu saída no remédio certo. A palavra quantidade é a máscara para representar o número informado de saída.

● **Voltar**

O AUM voltará para a tela principal.

48

Similarmente ao comando por voz “lista de medicamentos”, o AUM precisará recorrer ao uso de algumas API para executar a ação “utilizar medicamento”. Vamos descrever agora esse processo através da Figura 49.

Figura 49 - Fluxo do Reconhecimento de Comandos Por Voz



Fonte: Autor

A classe Recurso Voz *Life CyCle Observer* é fundamental no processamento de voz. Para facilitar o desenvolvimento unificamos todas as chamadas por voz para ela. Novamente para conseguir esse objetivo utilizamos o *callback*.

Além de utilizar o processo de *callback* foi necessário fazer uso do reconhecimento de voz do googleTM identificado através da intenção *RecognizerIntent*.

O uso da intenção tem por base a *Android Speech API* que fornece controle de reconhecimento, serviços em segundo plano, intenções e suporte para vários idiomas. A simples adição à entrada do usuário para o aplicativo é um recurso muito poderoso.

Tal recurso é útil para pessoas com deficiência que usam um teclado ou simplesmente para quem está tentando encontrar uma maneira de aumentar a produtividade e melhorar seu fluxo de trabalho.

49

A Figura 49 descreve o processamento de qualquer comando por voz realizado pelo usuário. Vamos descrever agora o comando por voz “remédio utilizado” e cada uma das etapas na classe. Conforme o fragmento (Figura 50) de código Java implementamos a interface *GerenteServicosListener* que terá o papel de dar suporte ao recurso de *callback*.

Figura 50 - Implementação do Ouvinte no Detalhe do Medicamento



Fonte: Autor

No segundo fragmento (Figura 51) já assinalamos para o gerenciador de voz qualquer interação por voz do usuário com a tela com o simples toque na tela.

50

Figura 51 - Criação do Recurso de Voz através da Classe RecursoVozObserver 

Fonte: Autor

No terceiro fragmento (Figura 52) vemos que qualquer clique na tela (e se o perfil for para o reconhecimento de voz) acionará a interface de voz.

51

Figura 52 - Controle de reconhecimento de Voz



Fonte: Autor

Logo após invocar o serviço de reconhecimento de voz, o assistente do GoogleTM aparece com o símbolo característico na Figura 53.

Figura 53 - Assistente Google de Voz



Fonte: Autor

O serviço do Google então percebe que o usuário parou de falar e devolve para o AUM o controle passando o conteúdo da voz em formato de texto. O fragmento de código Java da Figura 54 apresenta esse tratamento.

52

Figura 54 - Tratamento de dados após a captura de voz



Fonte: Autor

Embora extenso, utilizamos uma estrutura *switch*. Claro que poderíamos estender o serviço do padrão de projeto *Strategy* para eliminar a estrutura. Porém, isso demandaria mais tempo e tornando o projeto ainda mais complexo. Para diminuir o tempo de desenvolvimento optamos por não utilizar.

Novamente vemos no trecho de código da Figura 54 o uso da interface *GerenteServicosListener*. Assim que o comando de voz é processado e reconhecido, a ação é realizada através do tipo de ação “ACAO\_VOZ\_DOSE\_REALIZADA”.

Aqui é importante destacar que a tela do Android é chamada para resolver essa funcionalidade. O fragmento da Figura 55 exibe como é feito esse processo. Um método para executar ação da interface do *GerenteServicosListener* implementada pela tela do Android é acionado. O fragmento exibido na Figura 55 indica a chamada

53

seguinte onde irá percorrer pelas camadas de serviços e no final registrar a utilização do medicamento.

Figura 55 - Retorno do Controle de Execução ao Ouvinte



Fonte: Autor

O tópico final que será abordado será a pesquisa de medicamentos na base de dados da ANVISA. Nesta versão essa parte não terá suporte ao recurso de voz, porém, utilizaremos da mesma forma algumas chamadas de APIs entre elas a API Retrofit, assim como o recurso de *callback,* já que a busca na ANVISA tem como característica ser um serviço assíncrono por escolha de implementação. Naturalmente vamos precisar abordar nessa seção a conversão de arquivo pdf para arquivo texto.

Abaixo vemos a tela na Figura 56 que deverá ser exibida após o usuário clicar no + da tela principal.

54

Figura 56 - Pesquisa da ANVISA



Fonte: Autor

Aqui vemos uma lista preenchida com medicamento chamado “Doril”. Ao clicar em um dos itens da lista pesquisada na ANVISA através da API retrofit podemos visualizar os detalhes obtidos na Figura 57.

55

Figura 57 - Detalhes da Bula ANVISA - Parte 1

Fonte: Autor

A segunda parte da tela continua exibindo os detalhes na Figura 58.

56

Figura 58 - Detalhes da Bula ANVISA - Parte 2

Fonte: Autor

A terceira parte da tela continua exibindo os detalhes na Figura 59.

57

Figura 59 - Detalhes da Bula ANVISA - Parte 3



Fonte: Autor

O primeiro fragmento da tela possui um botão para capturar os dados. Ao clicar nesse botão o usuário entrará no cadastro de medicamento onde poderá complementar os dados como apelido, cor e quantidade de medicamento por embalagem. Além de já iniciar o estoque do medicamento através da quantidade por embalagem, o usuário poderá cadastrar o horário. A Figura 60 ilustra esse recurso.

58

Figura 60 - Cadastro do Medicamento e horário



Fonte: Autor

Como já explicamos, a tela do medicamento anterior não será necessária para o detalhamento. O usuário deverá apenas complementar as informações de apelido, cor, quantidade por embalagem e horário. Após isso, clica-se no botão confirmar.

Uma informação importante para destacar nesta funcionalidade é que a fonte de pesquisa não precisa necessariamente ser a ANVISA. Isso é possível devido ao uso do Padrão *“Strategy”.* Então podemos ter fontes de dados diferentes para a busca dos dados de medicamentos. Para dar suporte ao recurso utilizamos a interface Bulário Eletrônico Cliente. A interface é sustentada com o uso da tecnologia REST. Esta tecnologia tem como base o uso de arquivos do tipo JSON. A grande vantagem dessa tecnologia é a facilidade em conectar aplicações que rodam em ambientes heterogêneos.

59

REST é uma coleção de regras que os desenvolvedores seguem ao criar APIs; um conjunto de princípios que regem como diferentes programas se comunicam. Portanto, uma API REST é simplesmente uma API que aplica esses princípios.

Quando um cliente, um programa que solicita a conexão com uma API, solicita um recurso (informações que podem ser comunicadas e compartilhadas usando uma API), o estado existente do recurso é transferido de volta pelo servidor em uma representação padronizada.

Uma das regras do REST é que o usuário deve obter um dado (um recurso) ao vincular a um determinado URL. As APIs REST permitem que uma solicitação de um recurso vá do cliente para o servidor e, em seguida, que as informações relevantes sejam enviadas de volta como resposta.

Uma solicitação consiste em um *endpoint* (a URL que foi solicitada), um método, que define o tipo de solicitação enviada ao servidor; cabeçalhos, que representam os metadados e dados.

A Figura 61 apresenta esse processo com as respectivas definições.

Figura 61 - Interface de Bulário eletrônico



Fonte: Autor

Para dar suporte a estrutura mencionada na Figura 61, foi necessário dividir o projeto em camadas. Abaixo temos uma imagem (Figura 62) que exibe a estrutura da camada de API-integração.

60

Figura 62 - Estrutura do Pacote de Integração Bulário Eletrônico



Fonte: Autor

Na Figura 62 é apresentada uma estrutura de pacotes organizados de uma forma que expressa claramente uma divisão de papéis. O pacote API é a entrada de qualquer implementação, tanto para o acesso ao bulário eletrônico quanto à integração de usuário. Neste pacote estão as especificações de como a API deve responder para cada caso. O restante dos pacotes são os elementos necessários para a construção desse acesso. Temos o pacote *factory* responsável pela criação de objetos que representam os elementos da API. O pacote interfaces estará contido nas interfaces que são os contratos de utilização das APIs.

61

4.2. ESTRUTURA DE BANCO DE DADOS AUM

Modelo de dados é um diagrama estático. Quando um software é projetado um diagrama de como os dados vão fluir para dentro e para fora das diversas tabelas requer um diagrama que represente.

Este fluxograma é usado para definir as características dos formatos de dados, estruturas e funções de processamento de banco de dados para atender com eficiência os requisitos do fluxo de dados.

Depois da criação e implementação do banco de dados, o modelo permanece ativo e se torna a documentação e a justificativa da existência do banco de dados e de como o fluxo de dados foi projetado. (SAP, 2023)

A arquitetura proposta e o aplicativo AUM desenvolvido como *case* se desdobram sobre o modelo de dados da Figura 63.

Figura 63 - Modelo de Dados



Fonte: Autor

62

4.3. DICIONÁRIO DE DADOS

Todos os dados do *Firebase Realtime Database* são armazenados como objetos JSON. O usuário pode pensar no banco de dados como uma árvore JSON hospedada na nuvem. Ao contrário de um banco de dados SQL, não há tabelas ou registros. Quando são adicionados dados à árvore JSON, eles se tornam um nó na estrutura JSON existente com uma chave associada, podendo fornecer suas próprias chaves, como IDs de usuário ou nomes semânticos, ou elas podem ser fornecidas para o usuário.

Utilizo o nome “tabela” apenas para simplificação, embora conforme explicado acima, existem nós para representar os dados. O Apêndice A apresenta os nós, explicando o objetivo de cada nó e descrição de cada campo.

63

**5 RESULTADO E TESTES**

**5.1 Informações gerais**

O Aplicativo AUM foi desenvolvido utilizando o Android Studio com as especificações da Figura 64.

Figura 64 - Configuração Android Studio

Fonte: Autor

A escolha da IDE Android se deu em virtude da integração nativa com a linguagem Java, muito difundida no mercado. Além disso, devido aos inúmeros recursos no editor de código e a integração rápida com o git. Ademais, a IDE Android apresenta vários emuladores o que possibilita rapidamente uma simulação com vários tipos de dispositivos.

A linguagem utilizada foi a versão do Java 11 conforme Figura 65.

64

Figura 65 - Versão do Java

Fonte: Autor

A linguagem Java foi escolhida tanto pelo amplo uso no mercado de desenvolvimento de software, com um rico acervo mundial de exemplos para diversos tipos de situações que ocorrem durante o desenvolvimento de um software. Não podemos deixar de destacar a integração da linguagem Java com os padrões de projeto modernos utilizando o paradigma da programação orientada a objeto.

O Computador e o Sistema operacional são descritos na Figura 66. Figura 66 - Configurações Sistema Operacional e Computador



Fonte: Autor

65

**5.2 Testes**

A proposta do trabalho era de desenvolver um aplicativo baseado em uma arquitetura de software que possibilitasse ajudar pessoas idosas e com deficiência, em especial com baixa acuidade visual, a terem mais facilidade no seu dia a dia de rotinas terapêuticas.

Abordou-se que existem estratégias de memorização que podem ajudar a aumentar a adesão medicamentosa.

Discorremos que os cenários atuais em termos de soluções de software são ainda incipientes, além disso, mesmo quando a tecnologia assistiva está disponível, muitas vezes essas soluções de mercado ou fornecidas por profissionais da saúde deixam a desejar porque nem sempre os profissionais estão disponíveis para oferecer suporte e orientações no dia a dia da rotina terapêutica.

Diante do exposto, foi desenvolvido o aplicativo AUM para contribuir com o aumento da adesão das rotinas medicamentosas. Serão apresentados três testes: Inspeção de código através da ferramenta LINT, testes realizados pelo aplicativo SCANNER e por fim análise com base no site guia de acessibilidade. O resultado de cada teste ajudará a definir quais os limites do aplicativo e como ele poderá ser melhorado em trabalhos futuros.

5.1. LINT

O LINT é uma ferramenta de análise estática de inspeção de código embarcada no Android Studio IDE utilizada para o desenvolvimento de aplicativos da plataforma Android. Através dela o desenvolvedor pode analisar o projeto em busca de problemas estruturais em seu código, como acessibilidade, correções de layout, usabilidade entre outras características que podem ser inspecionadas.

A figura 67 apresenta uma configuração básica que pode ser feita acessando no Android Studio o menu file/settings.

66

Figura 67 - Configuração básica da ferramenta estática de inspeção de código 

Fonte: Autor

No contexto do trabalho proposto a análise foi focada nos itens destacados:

● Acessibilidade:

○ Verifica se o código intercepta mecanismos de clique que poderiam impedir os leitores de telas de funcionarem corretamente.

○ Verifica se existem imagens sem rótulos (atributo *contentDescription*) que permitam ao leitor de telas e outras ferramentas de acessibilidade descrever corretamente a interface do aplicativo.

○ Verifica se existem campos na tela marcados como clicáveis porém estão desabilitados para receber o foco.

○ Verifica se campos editáveis estão com rótulos (atributo *labelFor*), a fim de que o usuário tenha uma dica do objetivo ou restrições do campo. ● Correções - por se muito extenso será elencado apenas alguns itens ○ Verifica se há janelas não redimensionáveis para usuários aproveitarem telas maiores.

○ Verifica se há janelas duplicadas

○ Verifica se há conversões de widgets que podem gerar erros durante a execução do aplicativo

67