UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE CIÊNCIAS DA TECNOLOGIA

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

CARLOS ALBERTO SOARES DE ANDRADE

**UM ESTUDO SOBRE A ELABORAÇÃO DE UM APLICATIVO DE SMARTPHONE PARA AUXÍLIO DE PESSOAS COM DEFICIÊNCIA USUÁRIAS DE MEDICAMENTOS**

BELÉM - PA

2023

CARLOS ALBERTO SOARES DE ANDRADE

**UM ESTUDO SOBRE A ELABORAÇÃO DE UM APLICATIVO DE SMARTPHONE PARA AUXÍLIO DE PESSOAS COM DEFICIÊNCIA USUÁRIAS DE MEDICAMENTOS**

Trabalho de Conclusão de Curso para obtenção do grau em Engenharia da Computação pela Universidade Federal do Pará. Orientador: Prof. Agostinho Luiz da Silva Castro.

BELÉM - PA

2023

CARLOS ALBERTO SOARES DE ANDRADE

**UM ESTUDO SOBRE A ELABORAÇÃO DE UM APLICATIVO DE SMARTPHONE PARA AUXÍLIO DE PESSOAS COM DEFICIÊNCIA USUÁRIAS DE MEDICAMENTOS**

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

**Banca examinadora:**

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

Orientador

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

Nome/ Instituição

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

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

DEDICATÓRIA

“Para as pessoas sem deficiência a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis”. (RADABAUGH, 1993)

“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 resili ente e que me educou no caminho da verdade e consciência,

A minha 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,

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

RESUMO

Aqui fazer um breve resumo de introdução, objetivo, resultados, discussão e conclusão

PODE SER 1 PARAGRAFO, EM TORNO DE 8 LINHAS

ABSTRACT (RESUMO EM INGLÊS)

LISTA DE FIGURAS

[Figura 1-Percentual da População com algum grau de deficiência 15](#_Toc126357503)

[Figura 2-Aqueduto romano 21](#_Toc126357504)

[Figura 3-Objeto JSON representando dados de um filme 22](#_Toc126357505)

[Figura 4-Estrutura do Strategy 23](#_Toc126357506)

[Figura 5-Estrutura do Chain of Responsability 24](#_Toc126357507)

[Figura 6-Envolvendo um objeto com o padrão Proxy 25](#_Toc126357508)

[Figura 7-Estrutura de acesso a dados do Aplicativo AMU 26](#_Toc126357509)

[Figura 8-Estrutura em nó do banco de dados 29](#_Toc126357510)

[Figura 9-Visão geral da concepção do aplicativo AMU. 30](#_Toc126357511)

[Figura 10-Entrada de Dados 32](#_Toc126357512)

[Figura 11-Listas de Pesquisa 32](#_Toc126357513)

[Figura 12-Processos Autônomos 32](#_Toc126357514)

[Figura 13-Representa o barramento de serviços 33](#_Toc126357515)

[Figura 14-Componente de Integração de Bulário 33](#_Toc126357516)

[Figura 15-Serviço de Bulário Eletrônico da Anvisa na nuvem 33](#_Toc126357517)

[Figura 16-Componente de Integração de Usuário 34](#_Toc126357518)

[Figura 17-Interface de Conexão com os perfis 35](#_Toc126357519)

[Figura 18-Representa qualquer perfil suportado pela interface 35](#_Toc126357520)

[Figura 19-Componente de Acesso ao Banco de Dados 35](#_Toc126357521)

[Figura 20-Banco de Dados Remoto Firebase 35](#_Toc126357522)

[Figura 21-Janela de Acesso ao Aplicativo AMU 37](#_Toc126357523)

[Figura 22-Cadastro de Usuário 38](#_Toc126357524)

[Figura 23-Autenticação de Usuário 39](#_Toc126357525)

[Figura 24-Janela Principal 40](#_Toc126357526)

[Figura 25-Fragmento Responsável pela criação da Interface de Pesquisa do Bulário Eletrônico 42](#_Toc126357527)

[Figura 26-Fragmento de um método de usuário sem necessidade de acessibilidade 43](#_Toc126357528)

[Figura 27-Fragmento de um método que trata usuário idoso e/ou com visão reduzida/perda. 43](#_Toc126357529)

[Figura 28-Estrutura de Pacotes da Integração Bulário Eletrônico 44](#_Toc126357530)

[Figura 29-Jabela principal com a Lista de Medicamentos 45](#_Toc126357531)

[Figura 30-Representa os comandos por voz 46](#_Toc126357532)

[Figura 31-Fragmento que executa o comando de Voz de Retorno da Lista de Medicamentos 47](#_Toc126357533)

[Figura 32-Fragmento da Definição da Classe Principal com implementação do Recurso de Voz 48](#_Toc126357534)

[Figura 33-Fragmento que executa o método de integração de usuário para lista medicamentos 48](#_Toc126357535)

[Figura 34- Interface de Transações 49](#_Toc126357536)

[Figura 35-Classe que Implementa o Proxy de Transação 49](#_Toc126357537)

[Figura 36-Implementação da execução do Proxy 50](#_Toc126357538)

[Figura 37--Detalhes do Medicamento 52](#_Toc126357539)

[Figura 38-Alterar Medicamento 53](#_Toc126357540)

[Figura 39-Sinalizar Dose Utilizada 53](#_Toc126357541)

[Figura 40-Horário Ativo de um medicamento 53](#_Toc126357542)

[Figura 41-Lista de Saldo de Estoque 54](#_Toc126357543)

[Figura 42-Lista de Utilizações 54](#_Toc126357544)

[Figura 43-Exibir lista de Cores 55](#_Toc126357545)

[Figura 44-Paleta de Cores 55](#_Toc126357546)

[Figura 45-Entrada de Estoque 55](#_Toc126357547)

[Figura 46-Saída no estoque 56](#_Toc126357548)

[Figura 47-Cadastro de Horário 56](#_Toc126357549)

[Figura 48-Representa os comandos por voz 56](#_Toc126357550)

[Figura 49-Fluxo do Reconhecimento de Comandos Por Voz 57](#_Toc126357551)

[Figura 50-Implementação do Ouvinte no Detalhe do Medicamento 59](#_Toc126357552)

[Figura 51-Controle de reconhecimento de Voz 59](#_Toc126357553)

[Figura 52-Assistente Google de Voz 60](#_Toc126357554)

[Figura 53-Tratamento de dados após a captura de voz 60](#_Toc126357555)

[Figura 54-Retorno do Controle de Execução ao Ouvinte 61](#_Toc126357556)

[Figura 55-Pesquisa da ANVISA 62](#_Toc126357557)

[Figura 56-Detalhes da Bula ANVISA - Parte 1 63](#_Toc126357558)

[Figura 57-Detalhes da Bula ANVISA - Parte 2 63](#_Toc126357559)

[Figura 58-Cadastro do Medicamento e horário 64](#_Toc126357560)

[Figura 59-Interface de Bulário eletrônico 65](#_Toc126357561)

[Figura 60-Estrutura do Pacote de Integração Bulário Eletrônico 66](#_Toc126357562)

[Figura 61-Modelo de Dados 67](#_Toc126357563)

SUMÁRIO

1. INTRODUÇÃO —-------------------------------------------------------------------------

1.1. MOTIVAÇÃO —----------------------------------------------------------------------

1.2. OBJETIVOS —-----------------------------------------------------------------------

1.3. ORGANIZAÇÃO DO TRABALHO —-------------------------------------------

1. FUNDAMENTAÇÃO TEÓRICA -------------------------------------------------------

2.1. APPLICATION PROGRAMMING INTERFACE ------------------------------

2.2. FORMATO DE DADOS JSON —------------------------------------------------

2.3. PADRÕES DE PROJETO —------------------------------------------------------

2.3.1. ENTERPRISE SERVICE BUS —-------------------------------------

2.3.2. STRATEGY —-------------------------------------------------------------

2.3.3. CHAIN OF RESPONSIBILITY —-------------------------------------

2.3.4. PROXY —------------------------------------------------------------------

2.4. BANCO DE DADOS FIREBASE —---------------------------------------------

2.5. ARQUITETURA UTILIZADA —--------------------------------------------------

2.5.1. CLIENTES —--------------------------------------------------------------

2.5.2. BARRAMENTO DE SERVIÇOS —----------------------------------

2.5.3. INTERFACE DE INTEGRAÇÃO BULÁRIO ELETRÔNICO —

2.5.4. INTERFACE DE INTEGRAÇÃO DE USUÁRIO —--------------

2.5.5. INTERFACE DE ACESSO A DADOS —---------------------------

1. DESENVOLVIMENTO —----------------------------------------------------------------

3.1. O APLICATIVO —-------------------------------------------------------------------

3.1.1. PRIMEIRO ACESSO ---------------------------------------------------

3.1.2. JANELA PRINCIPAL ---------------------------------------------------

3.1.3. COMANDOS DISPONÍVEIS ------------------------------------------

3.1.4. ALARMES -----------------------------------------------------------------

3.1.5. DETALHES DO MEDICAMENTO -----------------------------------

3.1.6. COMANDOS DISPONÍVEIS ------------------------------------------

3.2. ESTRUTURA DE BANCO DE DADOS DO AMU —------------------------

3.3. DICIONÁRIO DE DADOS —------------------------------------------------------

1. RESULTADOS —---------------------------------------------------------------------------
2. DISCUSSÃO —-----------------------------------------------------------------------------
3. CONCLUSÃO —----------------------------------------------------------------------------
4. REFERÊNCIAS —-------------------------------------------------------------------------

# **1 INTRODUÇÃO**

Baseado no Censo de 2010, conforme 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, considera-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 com deficiência, 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 (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 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 estes públicos (BERSCH, 2017).

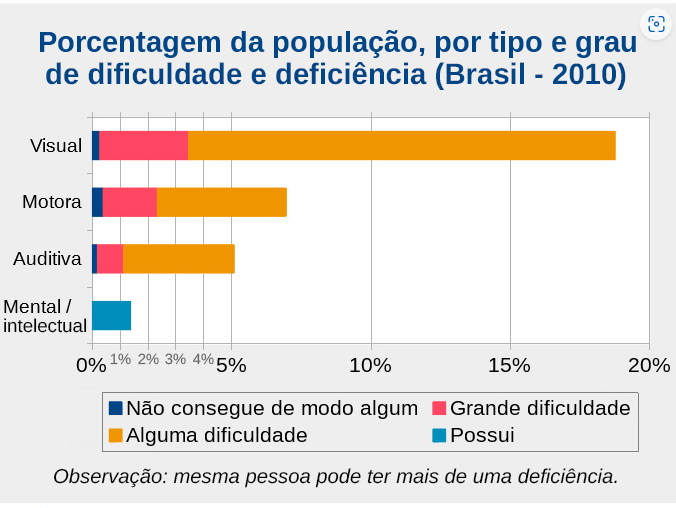
Observando-se esse cenário associado às PcDs, nota-se uma grande semelhança com os problemas enfrentados por pessoas idosas (perda de uma ou mais habilidades, dificuldade na realização de tarefas do cotidiano e uso de medicamentos). Este fato permite que a mesma ação seja utilizada por ambos os públicos.

Assim, o presente trabalho procura abordar sobre o desenvolvimento do aplicativo para auxílio de pessoas com deficiências e idosas usuárias de medicamentos (AMU) que tem por finalidade auxiliar o processo de administração de medicamentos para pessoas que necessitam de cuidados, tendo como principal alvo as 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.

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 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 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.

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 do 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.

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.

**2 CONTEXTUALIZAÇÃO DO PROBLEMA**

**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 retroalimentar (realimentar); 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 de 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 (SHAW & GARLAN, 1996).

Padrões de Projeto ou *design patterns* tiveram sua origem, segundo Gamma et al. (2000), nos trabalhos do arquiteto Christopher Alexander (ALEXANDER, 1977 apud GAMMA et al., 2000) sobre 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. Segundo Ecker, em seu projeto X-Engine (ECKER, 2003, p. 85), “as práticas reconhecidas de engenharia de software e padrões de projeto utilizadas no processo de desenvolvimento reduziram a quantidade de trabalho de recodificação 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 AMU 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 SERGEY 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 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 SERGEY 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.

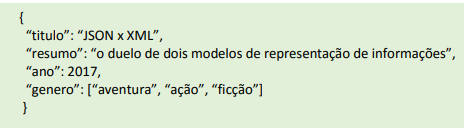
Estendendo o conceito sugerido na Figura 2, API pode interligar e integrar dois sistemas, em que um deles fornece informações e serviços que podem ser utilizados pelo outro, sem a necessidade de algum dos sistemas conhecer detalhes de implementação do outro. Em outras palavras, é uma forma bem segura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

Figura 3-Objeto JSON representando dados de um filme



Fonte: PROCEDE, BAHIA

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).

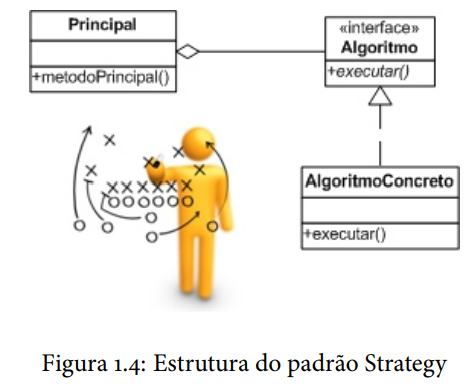
3.3. PADRÕES DE PROJETO

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 AMU 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



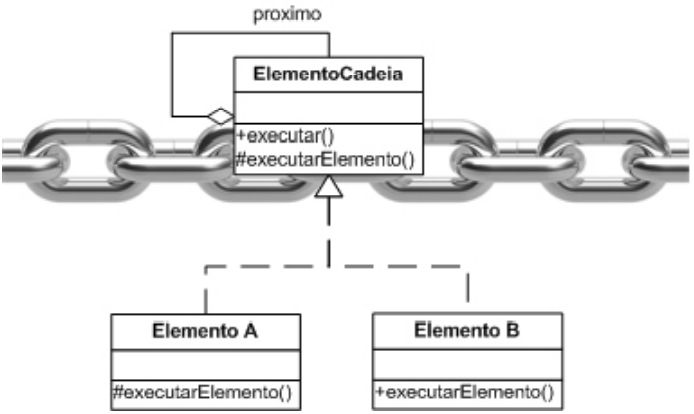
Fonte: Livro Design Patterns com Java

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 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



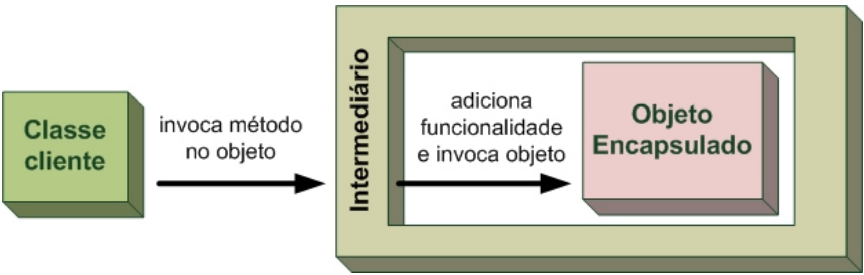
Fonte: Livro Design Patterns com Java

**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 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: Livro Design Patterns com Java

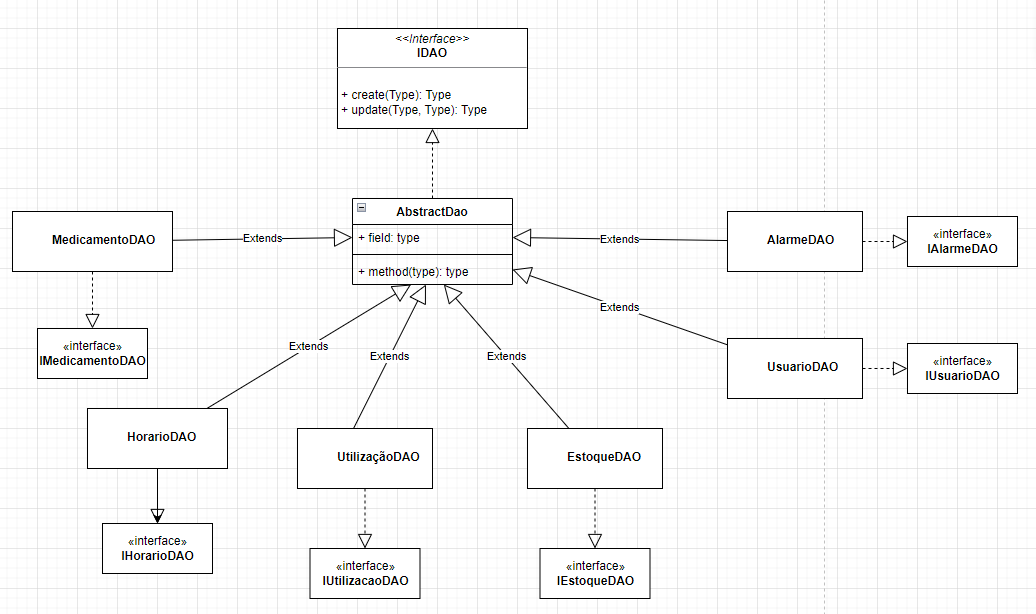
**3.3.3. *Data Access Object***

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 de GAMA et al (1995).

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.

Segue abaixo na Figura 7 como foi desenhado a estrutura de acesso a dados do Aplicativo AMU.

Figura 7-Estrutura de acesso a dados do Aplicativo AMU



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 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*.

3.4. BANCO DE DADOS *FIREBASE*

Segundo KORTH (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 AMU apresente uma estrutura de poucas pastas ou *“nodes”*, essepode se beneficiar 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 AMU 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 AMU é plano.

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. 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.

Figura 8-Estrutura em nó do banco de dados

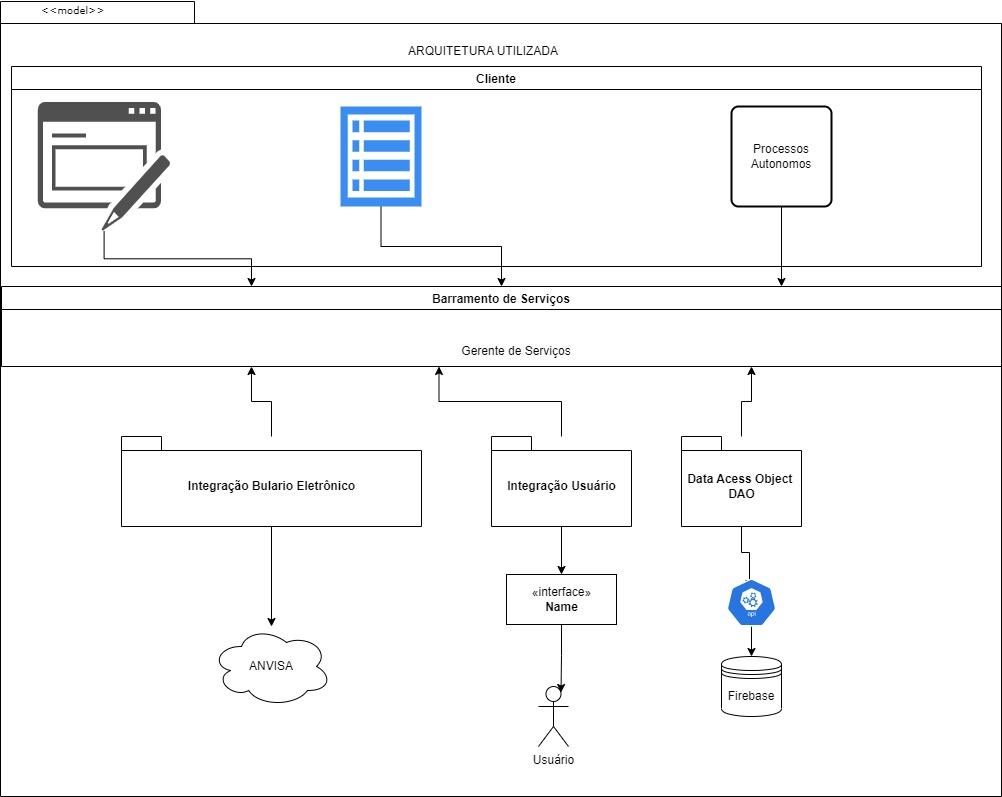


Fonte: *Firebase Realtime Database*

3.5. ARQUITETURA UTILIZADA

O aplicativo AMU, produto final do trabalho corrente, 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 AMU deverá incorporar soluções que interajam fácil e adequadamente com esse grupo de pessoas. Para tanto, o projeto e construção do aplicativo AMU 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 AMU.



Fonte: o autor

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

* 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 AMU são indicados na Figura 9 e detalhados nas Figura 10 a Figura 12.

A Figura 10 representa janelas de entrada de dados no AMU, 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 AMU, 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 AMU onde não existem interações do usuário, como por exemplo: a rotina que verifica um alarme.

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 (EDUARDO 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 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 (EDUARDO 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
   1. Qualquer usuário que não fizer uso de recurso de acessibilidade como comandos por voz.
2. Usuário PCD
   1. Usuário com perda de visão ou reduzida
   2. Usuário com perda auditiva
   3. 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 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

**4 DESENVOLVIMENTO**

O aplicativo AMU, produto final do trabalho corrente, destina-se a auxiliar a utilização de medicamentos por grupos de pessoas, tais como: idosos e portadores de deficiência. Para atender a esse público, a concepção do AMU deverá incorporar soluções que interajam fácil e adequadamente com esse grupo de pessoas. Para tanto, o projeto e construção do aplicativo AMU envolveu a utilização de conceitos e estratégias de engenharia de *software*, as quais foram detalhadas no capítulo 3.

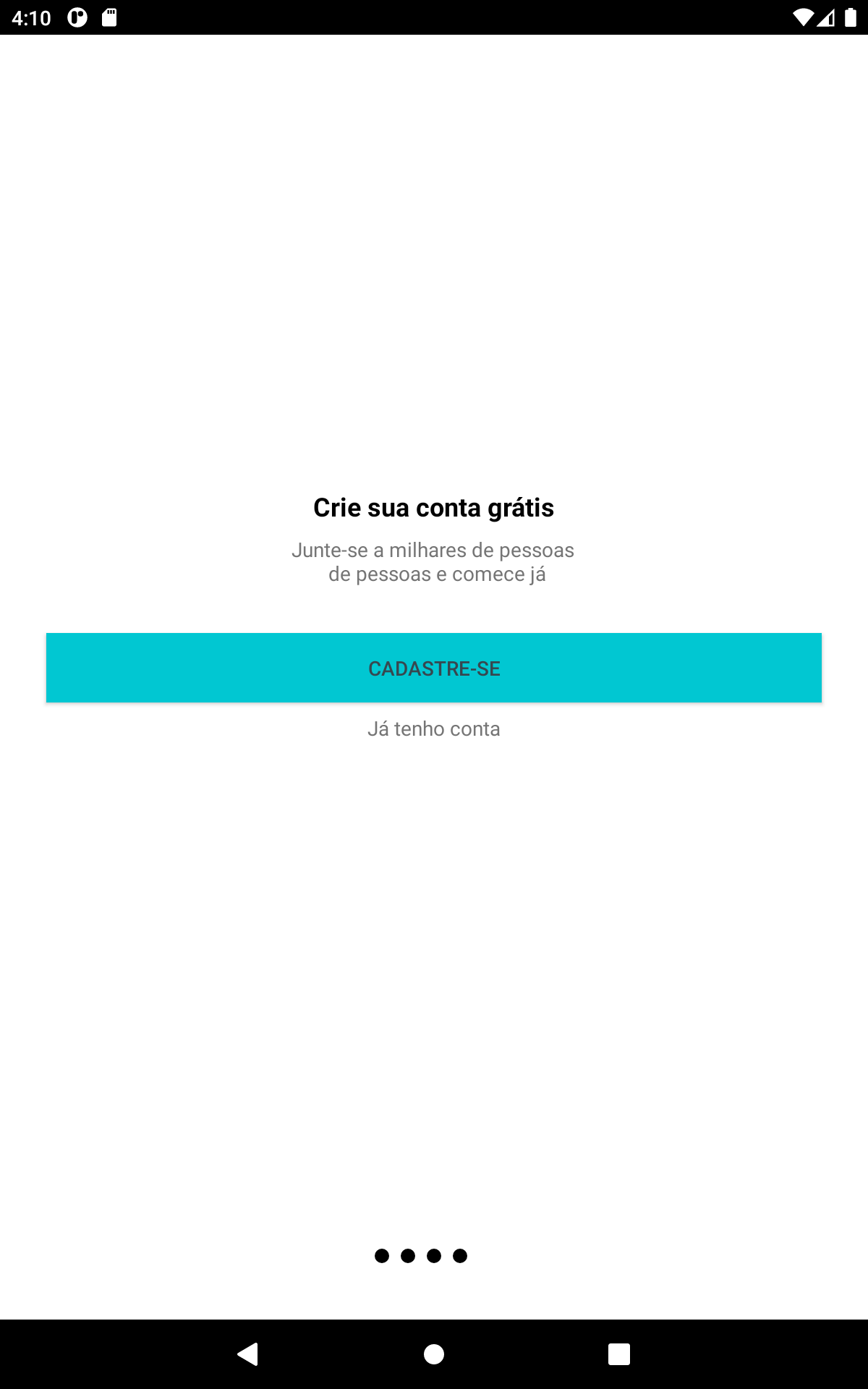
No presente trabalho analisa o uso de medicamentos por pessoas com perfil específico como idosos e PcDs.

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 acesso informando o e-mail e senha (Figura 21).

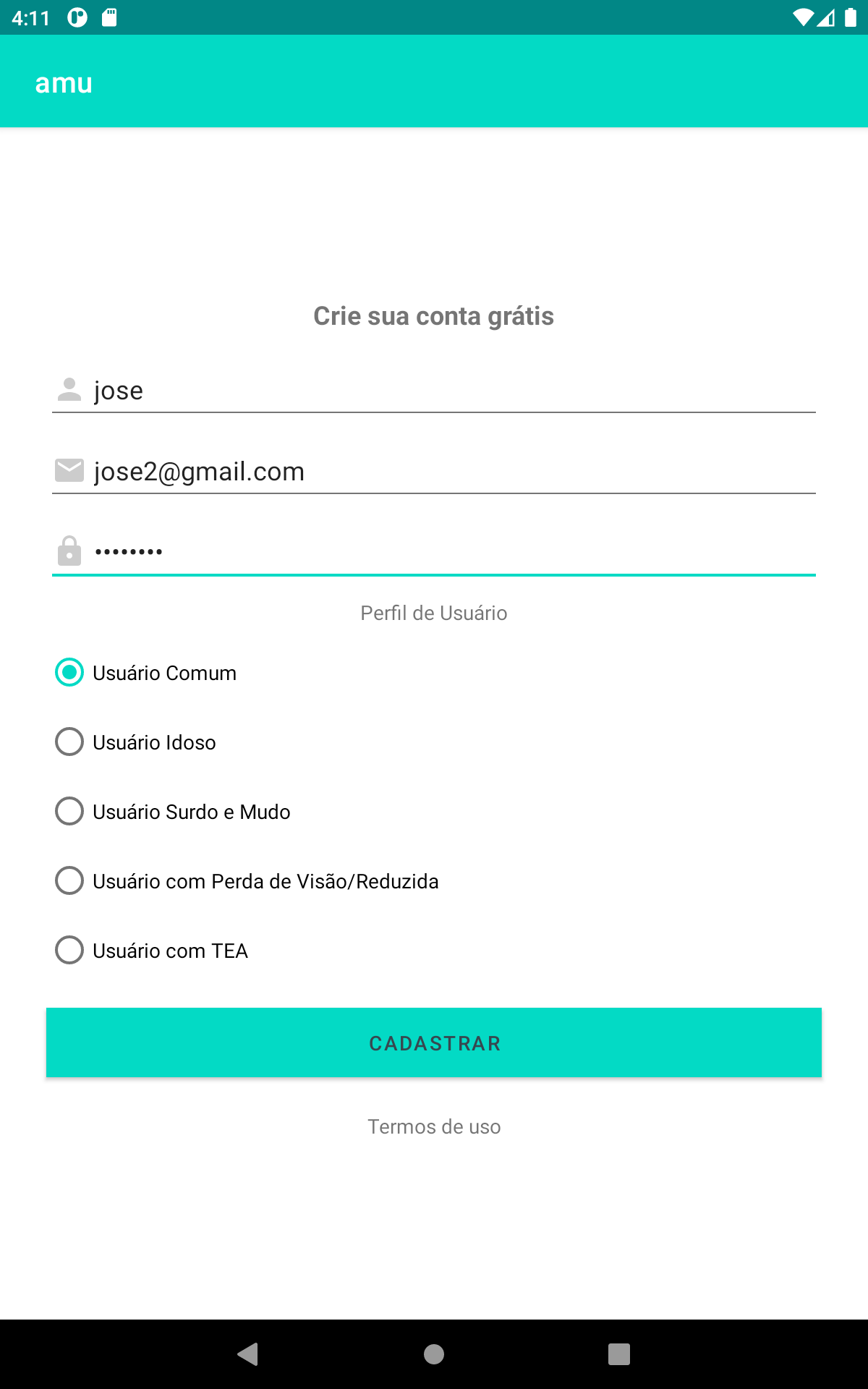
Figura 21-Janela de Acesso ao Aplicativo AMU



Fonte: Aplicativo AMU

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).

Figura 22-Cadastro de Usuário

****

Fonte: Aplicativo AMU

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

Figura 23-Autenticação de Usuário



Fonte: Aplicativo AMU

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

Figura 24-Janela Principal



Fonte: Aplicativo AMU

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, 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.

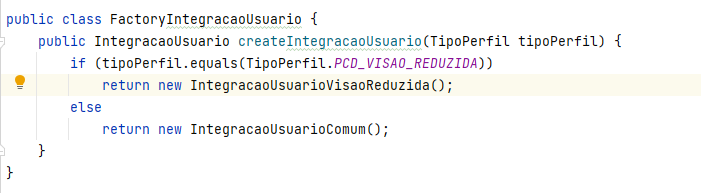
Nesta primeira versão do aplicativo AMU, o usuário sem necessidade de auxílio para acessibilidade também poderá utilizar todas as funcionalidades. O usuário idoso e/ou com perda de visão ou reduzida terá, através do uso da voz, acesso a algumas funcionalidades. Para usuários surdos/mudos e TEA não há funcionalidades adaptadas.

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 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.

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

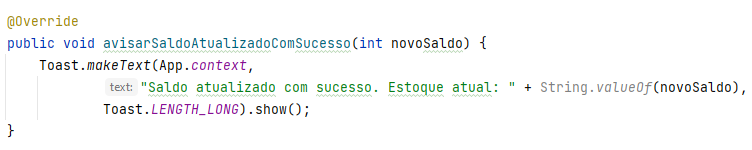


Fonte: Aplicativo AMU

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 respectivamente*.*

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.

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

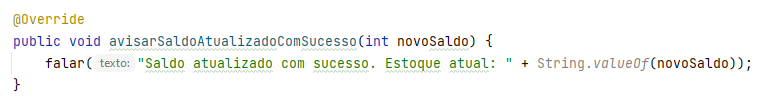


Fonte: Aplicativo AMU

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.

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

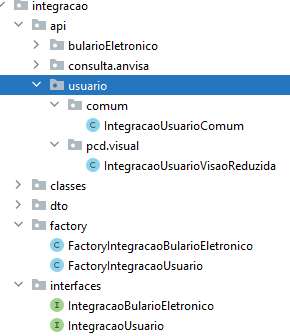


Fonte: Aplicativo AMU

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.

Figura 28-Estrutura de Pacotes da Integração Bulário Eletrônico



Fonte: Aplicativo AMU

Na Figura 28 é apresentado 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**

Como 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 com visão reduzida/perda, foram desenvolvidos alguns comandos por voz. Partimos da premissa que para o perfil de visão reduzida/perda será necessário 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-Jabela principal com a Lista de Medicamentos



Fonte: Aplicativo AMU

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:

Figura 30-Representa os comandos por voz



Fonte: Autor

**4.1.4. Comandos Disponíveis**

O AMU falará os comandos por voz disponíveis

* **Lista de Medicamentos**

O AMU lerá as informações principais de cada medicamento previamente cadastrado

* **Detalhe do Medicamento “apelido”**

O AMU 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 AMU 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 AMU 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 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.

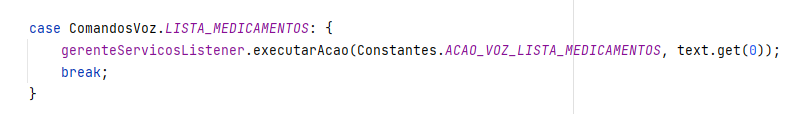
O *callback* é muito comum na linguagem Java, por exemplo, durante a busca de dados ou reprodução de mídia. Isso porque permite ao programador especificar o que deve ocorrer quando a execução acabar.

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 22 a 24) 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.

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



Fonte: Aplicativo AMU

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*.*

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

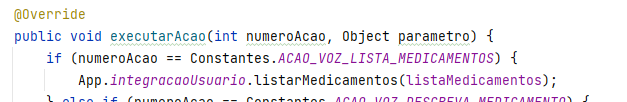


Fonte: Aplicativo AMU

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

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

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



Fonte: Aplicativo AMU

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

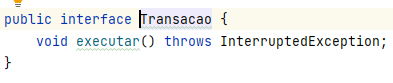
**4.1.5. Alarmes**

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.

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 transações responsável por representar o componente intermediário.

Figura 34- Interface de Transações



Fonte: Aplicativo AMU

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

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.

Figura 36-Implementação da execução do Proxy

### 

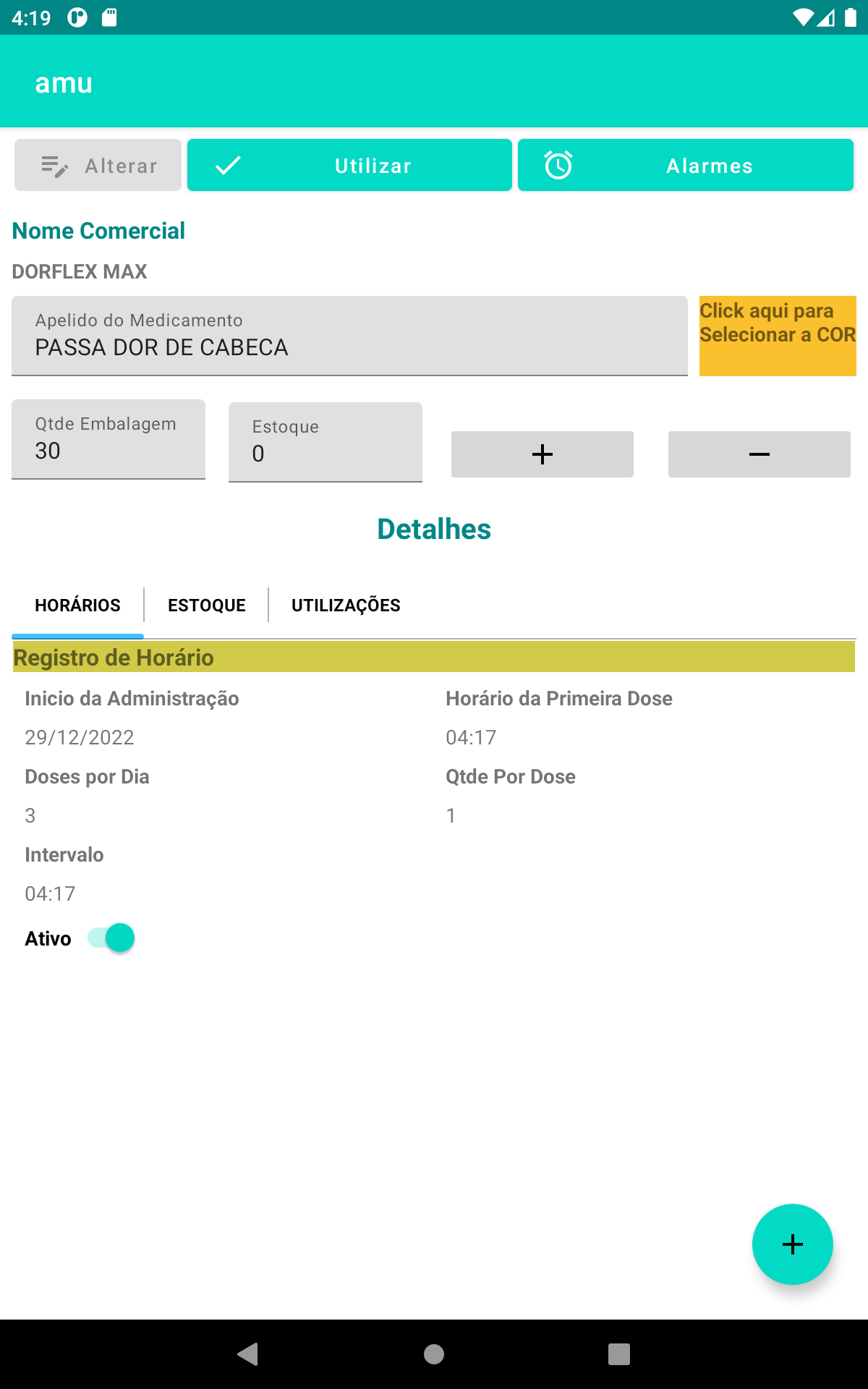
Fonte: Aplicativo AMU

### ]

**4.1.6. Detalhes do Medicamento**

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

Figura 37--Detalhes do Medicamento



Fonte: Aplicativo AMU

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: Aplicativo AMU

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

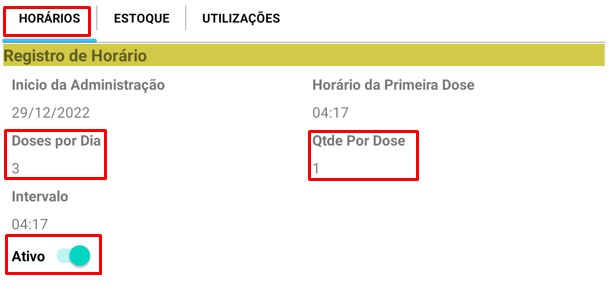


Fonte: Autor

São feitas algumas críticas para tornar o processo válido:

* 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: Aplicativo AMU

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.

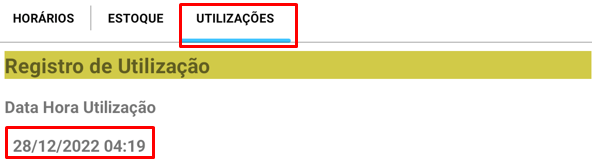
Figura 41-Lista de Saldo de Estoque



Fonte: Aplicativo AMU

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

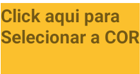
Figura 42-Lista de Utilizações



Fonte: Aplicativo AMU

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.

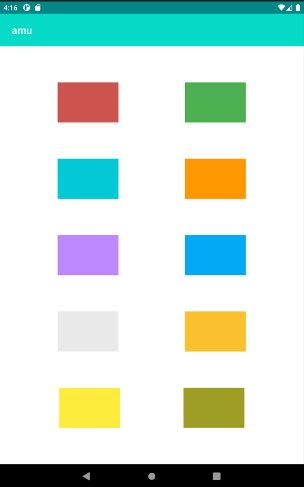
Figura 43-Exibir lista de Cores



Fonte: Aplicativo AMU

O usuário poderá apenas clicar em uma das cores e assim mudar a forma como é apresentada a linha na lista de medicamentos (Figura 43).

Figura 44-Paleta de Cores



Fonte: Aplicativo AMU

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: Aplicativo AMU

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: Aplicativo AMU

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

Figura 47-Cadastro de Horário



Fonte: Aplicativo AMU

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 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 AMU emitirá os seguintes comandos por voz:

* **Horário do Medicamento**

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

* **Estoque Atual**

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

* **Entrada “quantidade”**

O AMU 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”**

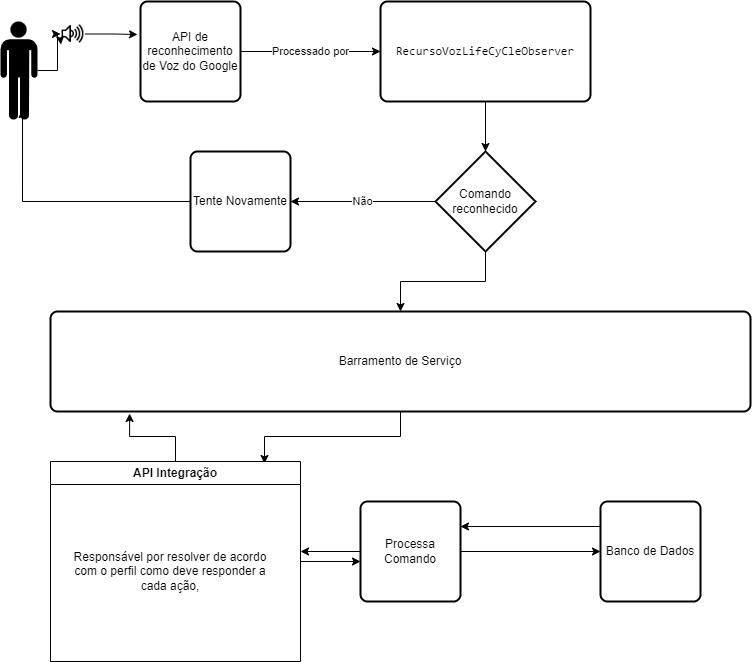
O AMU executará a mesma funcionalidade do botão +. Porém informará o resultado da ação por voz para o usuário ter certeza que deu saída no remédio certo. A palavra quantidade é a máscara para representar o número informado de saída.

* **Voltar**

O AMU voltará para a tela principal.

Similarmente ao comando por voz “lista de medicamentos”, o AMU precisará recorrer ao uso de algumas API para executar a ação “utilizar medicamento”. Vamos descrever agora o processo ilustrado na 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.

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*.

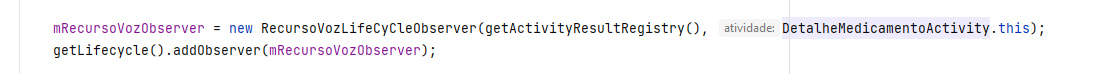
Figura 50-Implementação do Ouvinte no Detalhe do Medicamento



Fonte: Aplicativo AMU

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

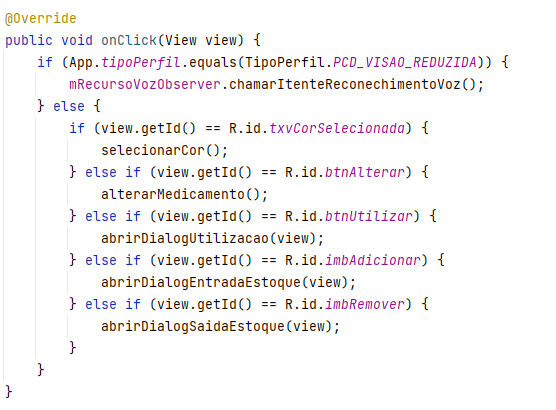
Figura 51-Criação do Recurso de Voz através da Classe RecursoVozObserver



Fonte: Aplicativo AMU

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.

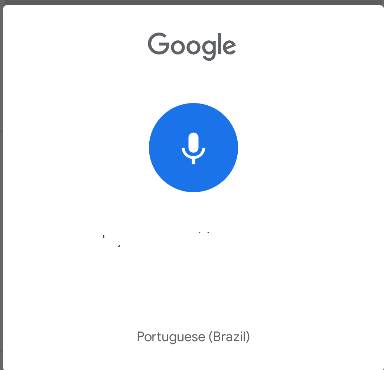
Figura 52-Controle de reconhecimento de Voz



Fonte: Aplicativo AMU

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 AMU o controle passando o conteúdo da voz em formato de texto. O fragmento 1 abaixo de código Java (Figura 54) apresenta esse tratamento.

Figura 54-Tratamento de dados após a captura de voz



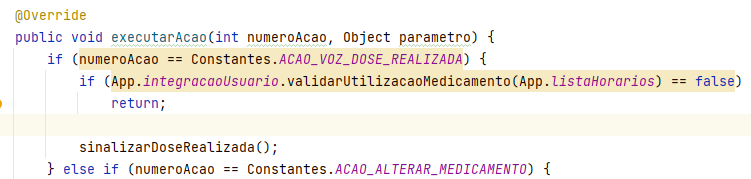
Fonte: Aplicativo AMU

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 GerenteServicos*Listener*. 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 2, exibido na Figura 55, indica a chamada 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

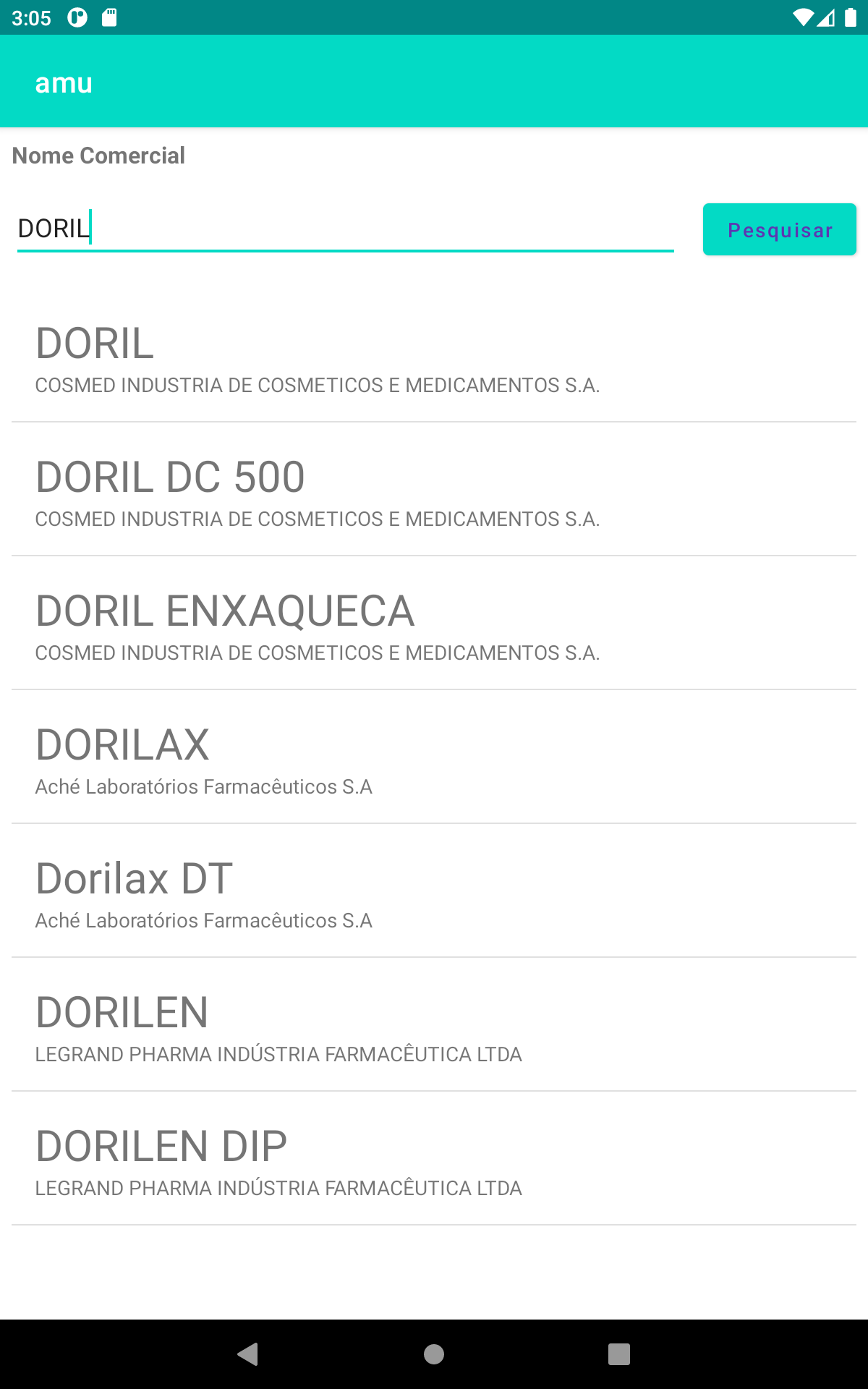


Fonte: Aplicativo AMU

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.

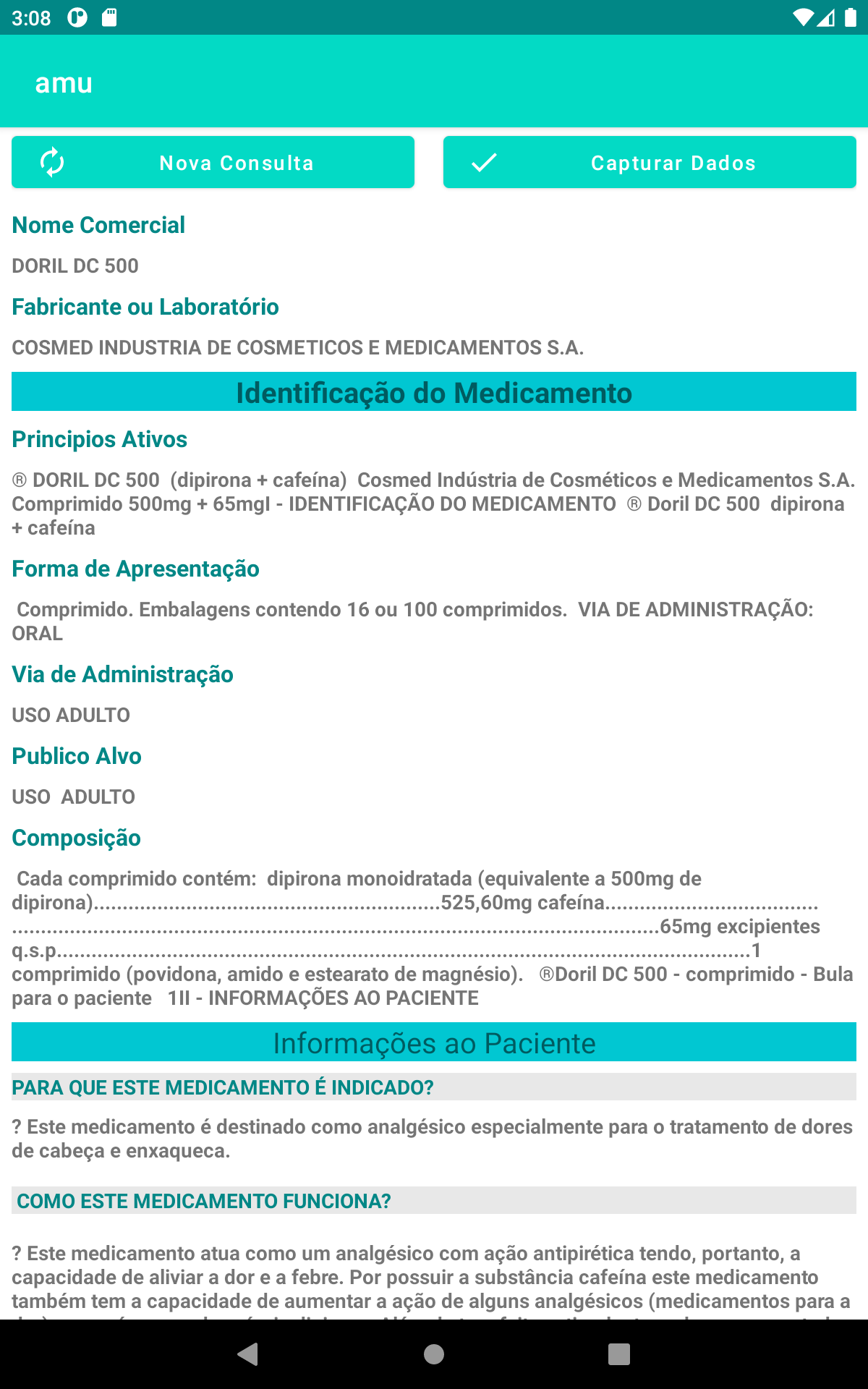
Figura 56-Pesquisa da ANVISA



Fonte: Aplicativo AMU

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.

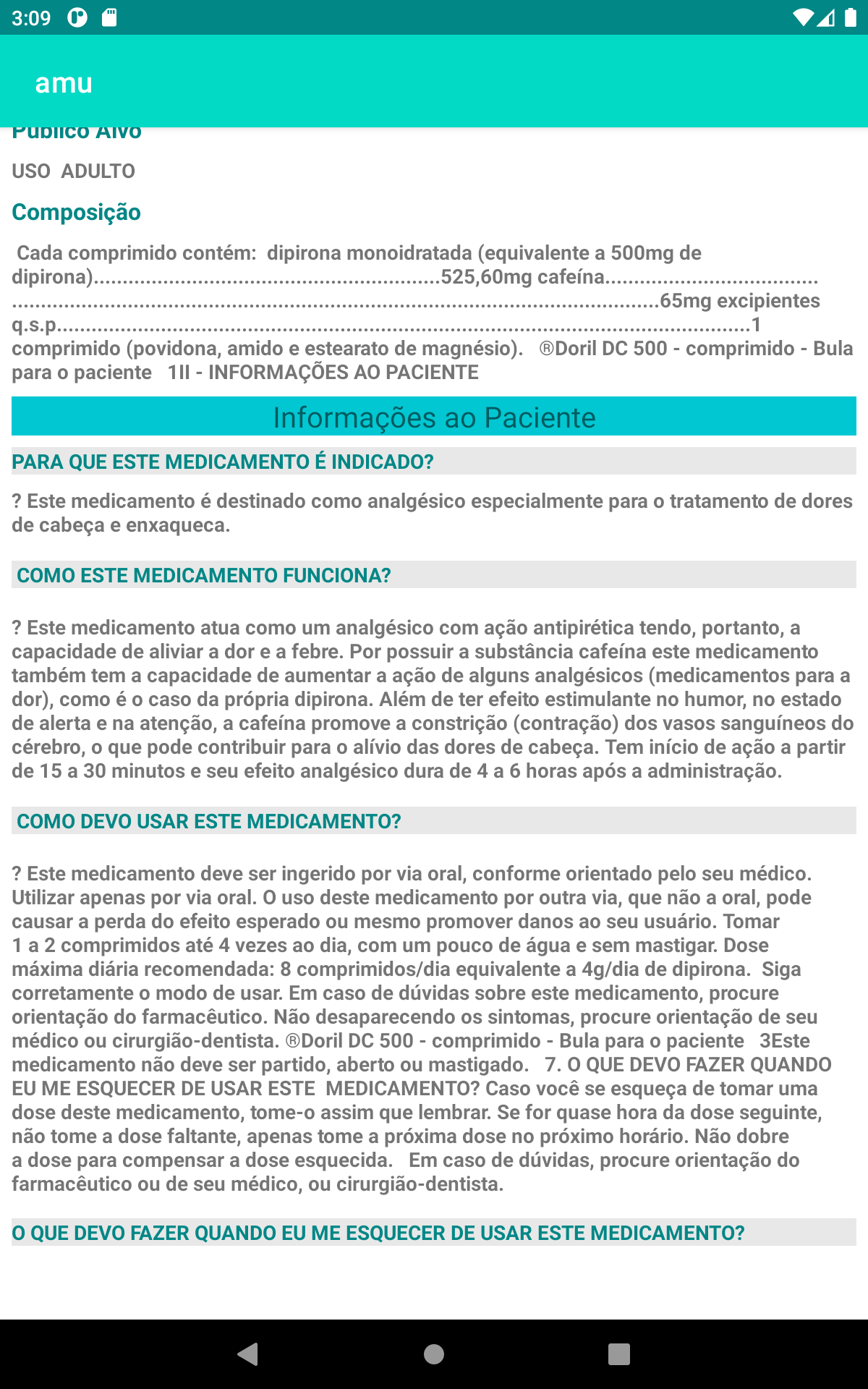
Figura 57-Detalhes da Bula ANVISA - Parte 1



Fonte: Aplicativo AMU

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

Figura 58-Detalhes da Bula ANVISA - Parte 2



Fonte: Aplicativo AMU

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 59 ilustra esse recurso.

Figura 59-Cadastro do Medicamento e horário



Fonte: Aplicativo AMU

Como já explicamos, a tela do medicamento anterior não será necessária 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.

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 meta-dados e dados.

A Figura 60 apresenta esse processo com as respectivas definições.

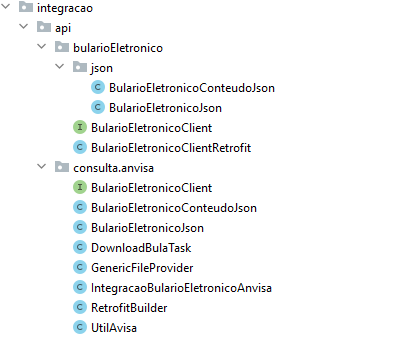
Figura 60-Interface de Bulário eletrônico



Fonte: Aplicativo AMU

Para dar suporte a estrutura mencionada na Figura 60, foi necessário dividir o projeto em camadas. Abaixo temos uma imagem (Figura 61) que exibe a estrutura da camada de API-integração.

Figura 61-Estrutura do Pacote de Integração Bulário Eletrônico

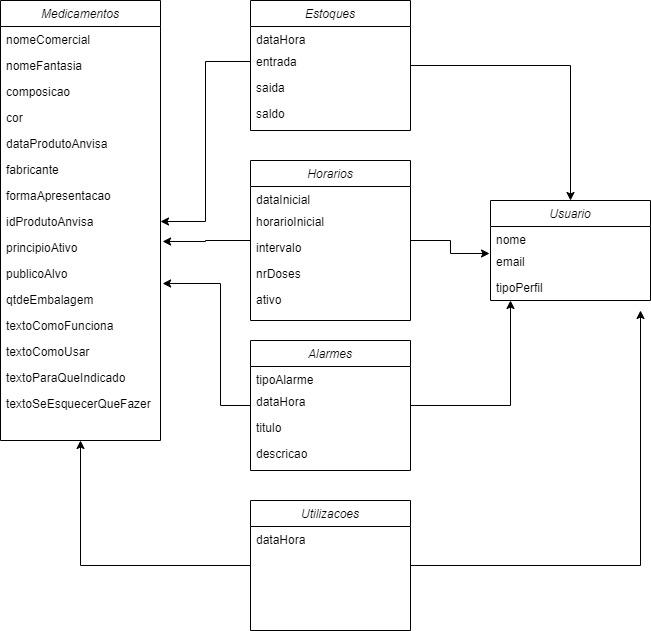


Fonte: Aplicativo AMU

Na Figura 61 é apresentado 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.

4.2. ESTRUTURA DE BANCO DE DADOS AMU

Figura 62-Modelo de Dados

****

Fonte: Aplicativo AMU

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.

**REFERÊNCIAS BIBLIOGRÁFICAS**

ALUR, D.; CRUPI, J.; MALKS, D. **Core J2EE Patterns: Best Practices and Design Strategies.** Palo Alto: Sun Microsystems Press, 2001. p. 460.

BAHIA, PROCEDE. **Manual Técnico Formato JSON.** Processamento e Certificação de documentos eletrônicos. Disponível: <<https://www.procedebahia.com.br/manual_json.pdf>>. Acesso dia 29 de janeiro de 2023.

BEDER, D. **Uma Arquitetura de Software baseada em Padrões de Projeto para o Desenvolvimento de Aplicações Concorrentes Confiáveis:** Técnicas de Engenharia de Software para Estruturação de Software, 2001.

BRASIL, Ministério da Saúde. Resolução-RDC Nº 16, de 2 de março de 2007: **Aprova Regulamento Técnico para Medicamentos Genéricos.** Diário Oficial da União: Brasília, DF, 2007.

BRASIL, Ministério da Saúde. Secretaria de Atenção à Saúde. Departamento de Ações Programáticas Estratégicas. **Política Nacional de Saúde da Pessoa com Deficiência**. Brasília: Editora do Ministério da Saúde, 2010. p. 07.

CASANOVA, M. A.; MOURA, A. V. **Princípios de Sistemas de Gerência de Bancos de Dados Distribuídos: Edição Revisada**. 1999.

CASTRO, S.S.; CESAR, C.L.; CARANDINA, L.; BARROS, M.B.; ALVES, M.C.; GOLDBAUM, M. **Physical disability, recent illnesses and health self-assessment in a population-based study in São Paulo, Brazil.** Disabil Rehabil, 2010.

EDUCATION, IBM Cloud, ESB-Barramento de Serviços Corporativo, **O conceito foi emprestado em uma nova abordagem**, em 2019. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/esb#:~:text=Um%20ESB%2C%20ou%20barramento%20de,como%20interfaces%20de%20servi%C3%A7o%20para>. Acesso em 28/12/2022

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.; (1994)**. Design Patterns: Elements of Reusable Object-Oriented Software.** Reading, MA: Addison-Wesley, 1994.

GUERRA, E; **Design Patterns com java: Projeto orientado a objetos guiado por padrões**, 2012. p. 200-207.

GUERRA, E; **Design Patterns com java: Projeto orientado a objetos guiado por padrões.** Estudos em Design, 2012. p. 14-18.

Hernán **Astudillo**. Financial Systems Architects hernan@acm.org ... OD'2002 - Hernan **Astudillo** ... Programming, Addison-Wesley, **1998**

<https://core.ac.uk/download/pdf/296829559.pdf>

IBGE EDUCA. **Pessoas com deficiência**. 2010. Figura 60. 682 x 560 pixels. Disponível em: <https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/20551-pessoas-com-deficiencia.html>. Acesso em: 16 de Janeiro de 2023.

IBGE, Censo Demográfico, em 2010. Disponível em:<https://www.ibge.gov.br/estatisticas/sociais/populacao/9662-censo-demografico-2010.html?=&t=destaques> . Acesso em 16/01/2023.

KONSTANTINOV, S. **The API eBook.** 2022.

KORTH, H.F.; SILBERSCHATZ, A. **Sistemas de Bancos de Dados.** Makron Books,1994.

NASCIMENTO, E.L., MARQUES, L.A.M. **O deficiente visual e a atenção farmacêutica.**Lat. Am. J. Pharm.Alfenas, v. 28, n.2, 2009. p. 203-10.

PINHEIRO, A.F. **Fundamentos da Engenharia de Software: Conceitos Básicos.** Recife, 2015. p. 244.

SHAW, M.; GARLAN, D. **Software Architecture. Perspectives on an Emerging Discipline**, Prentice Hall, 1996.

Significado de retroalimentação. **Dicionário Online de Português.** Disponível em: <<https://www.dicio.com.br/retroalimentacao/#:~:text=Significado%20de%20Retroalimenta%C3%A7%C3%A3o&text=A%C3%A7%C3%A3o%20de%20enviar%20elementos%20da,As%20respostas%20encontradas%20nesse%20processo>>. Acesso em: 29 de janeiro de 2023.

SOMMERVILLE, I. **Engenharia de Software**, 9a Edição, 2011.

CORDEIRO, A.C. Desenvolvimento do BackEnd de um Aplicativo Mobile para Smartphone, 2014, p.15-16.

**APÊNDICE A - DICIONÁRIO DE DADOS**

**Tabela de Usuários**

**Objetivo:** registrar os usuários que utilizam o aplicativo.

**Campos**

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descrição** | **Observação** |
| idUsuario | Código do Usuário |  |
| Nome | Nome do Usuário |  |
| Email | Email do usuário | Relacionado a Autenticação do Firebase |
| tipoPerfil | Tipo do Perfil | Campo que determina o comportamento do AMU |

**Tabela Medicamentos**

**Objetivo:** Registrar os medicamentos com dados capturados da Anvisa

**Campos**

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descrição** | **Observação** |
| idMedicamento | Código Automático |  |
| idUsuario | Código do Usuário |  |
| nomeComercial | Nome Comercial do Medicamento |  |
| nomeFantasia | Apelido do Medicamento |  |
| qtdeEmbalagem | Quantidade do Medicamento na Embalagem |  |
| Cor | Cor que identifica o medicamento na lista de medicamentos |  |
| composicao | Composição do Medicamento | Importado da bula |
| dataProdutoAnvisa | Data do Produto Anvisa | Importado da Bula. Também utilizado para determinar se a bula foi alterada. |
| idProdutoAnvisa | Chave Identificadora da Bula na Anvisa | Importado da Bula. Também utilizado para determinar se a bula foi alterada. |
| fabricante | Nome do Fabricante da Bula da Anvisa. | Importado da Bula. |
| formaApresentacao | Forma da Apresentação do Remédio | Importado da Bula. |
| principioAtivo | Princípio Ativo | Importado da Bula. |
| publicoAlvo | Público Alvo | Importado da Bula. |
| textoComoFunciona | Como funciona o medicamento | Importado da Bula. |
| textoComoUsar | Como usar o medicamento | Importado da Bula. |
| textoParaQueIndicado | Para que o medicamento é indicado | Importado da Bula. |
| textoSeEsquecerQueFazer | O Que fazer se esquecer de tomar o medicamento | Importado da Bula. |

**Tabela de Estoque**

**Objetivo**: Registrar a quantidade disponível do medicamento para utilização do usuário. O usuário poderá adicionar saldo ou subtrair o saldo.

**Campos**

|  |  |
| --- | --- |
| **Campo** | **Descrição** |
| idEstoque | Código Automático |
| dataHora | Data e Hora do Registro |
| Entrada | Quantidade de Entrada no Estoque. |
| Saída | Quantidade de Saída do Estoque. |
| Saldo | Saldo considerando saldo anterior mais entrada menos saída |

**Tabela de Alarmes**

**Objetivo**: Registrar os alarmes produzidos com base no horário e utilizações.

**Campos**

|  |  |
| --- | --- |
| **Campo** | **Descrição** |
| idAlarme | Código Automático |
| tipoAlarme | Tipo de Alarme |
| dataHora | Data e Hora do Alarme |
| Titulo | Título do Alarme |
| Descricao | Descrição do Alarme |

**Tabela de Horários**

**Objetivo**: Registrar os horários que os medicamentos devem ser tomados

**Campos**

|  |  |
| --- | --- |
| **Campo** | **Descrição** |
| idHorario | Código Automático |
| idMedicamento | Código do medicamento |
| idUsuario | Código do Usuário |
| dataInicial | Data Inicial que o medicamento deve ser administrado |
| horaInicial | Horário da Primeira dose do dia |
| Intervalo | Intervalo entre as doses |
| nrDoses | Número de Doses por dia |
| QtdePorDose | Quantidade que deve ser administrada durante cada dose |
| Ativo | Indica se o horário está ativo ou não. |

**Tabela de Utilização**

**Objetivo**: Registrar quando o usuário utiliza o medicamento.

**Campos**

|  |  |
| --- | --- |
| **Campo** | **Descrição** |
| idUtilizacao | Código Automático |
| idMedicamento | Código do Medicamento |
| idUsuario | Código do Usuário |
| DataHora | Data e Hora da Utilização |