DESENVOLVIMENTO

O aplicativo AMU produto final do trabalho corrente destina-se a auxiliar a utilização dos medicamentos por grupos de pessoas com alguma comorbidade. Logo, para isso é necessário apresentar algumas bases de que propiciem a adequação de forma mais simples. A figura 200 demostra alguns componentes:

* Cliente
* Barramento de Serviços
* Inteface de Bulário Eletrônico
* Interface de Integração de Usuário
* Interface de Acesso ao Banco de Dados

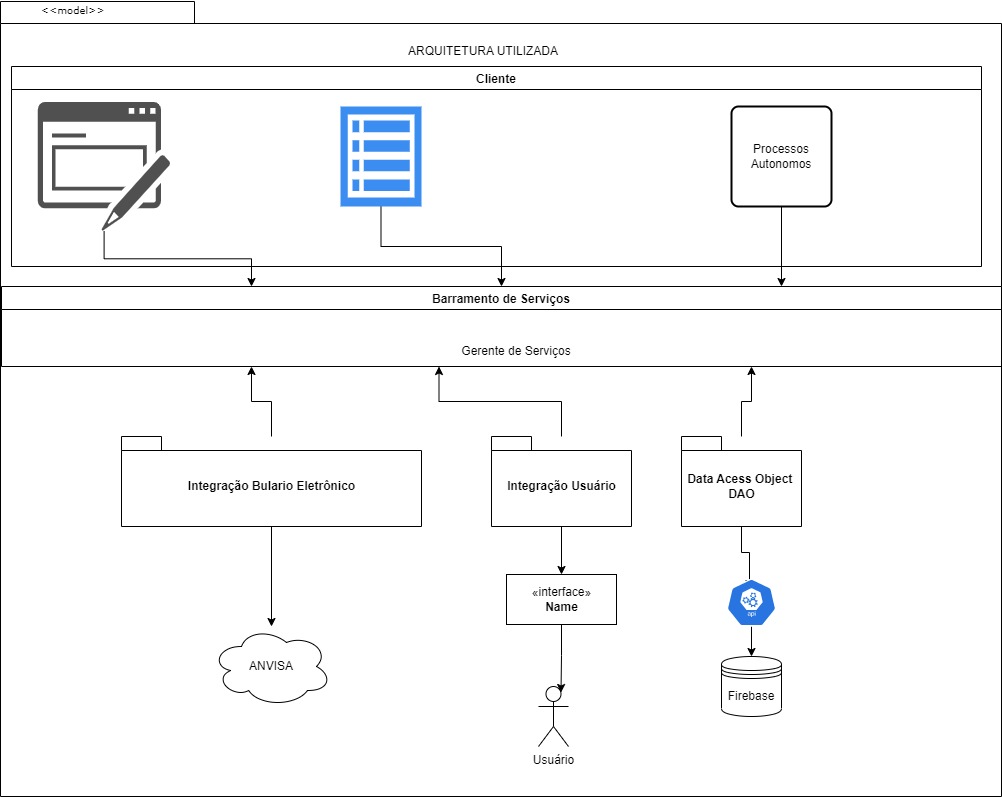


Figura 2000

Explicando a arquitetura de alto nível

1. **Cliente**

Objetivamente falando seria qualquer interface ou recurso utilizado pelo usuário para consumir ou solicitar um serviço do aplicativo. Pode ser um formulário ou uma lista. Assim como pode ser um serviço autônomo monitorando horários por exemplo.

1. **Barramento de Serviços**

Objetiva criar uma interface de acesso a todos os recursos do aplicativo. Para realizar 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. (1)

1. **Interface de Integração Bulário Eletrônico**

Esse componente possibilita criar uma ponte entre o serviço e a consulta a fontes de dados de medicamentos. Aqui 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 chegando ao ponto de trocar a fonte de dado e a mudança não gerar nenhum tipo de problema no restante do projeto. (2)

1. **Interface de Integração de Usuário**

Seguindo o mesmo padrão “Strategy” abordado no tópico “Integração de Bulário Eletronico” é necessário criar recursos específicos para os tipos de comorbidades apresentadas. Por exemplo temos uma interface para pessoas com visão reduzida ou mesmo sem visão. Segue a lista das interfaces:

1. Usuario comum
2. Usuario PCD
   1. Usuário com perda de visão ou reduzida
   2. Usuario com perda auditiva
   3. Usuário com TEA
3. Usuario Idoso

Utilizando a interface podemos por exemplo avisar que é horário de 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.

1. **Interface de Acesso a Dados**

O componente de acesso dados além de abstrair o uso da API do firebase possibilita que o usuário tenha acesso em qualquer dispositivo instalado em virtude do Banco de dados está remoto. Utilizamos o padrão de projeto “DAO”. Esse padrão possibilita a troca do banco de dados. Apenas trocando a fábrica da implementação.

1. **Problema analisado no trabalho**

No presente trabalho analisa-se o uso de medicamentos por pessoas com perfil específicos como idosos e pcds

1. **Aplicativo**

**Primeiro Acesso**

No primeiro acesso do aplicativo alguns slides de boas-vindas são apresentados. Então é apresentado ao usuário criar um cadastro novo ou efetuar o login informando e-mail rio e senha. Após efetuar o cadastro o aplicativo exibirá a janela principal. Se o usuário possuir um cadastro feito anteriormente escolherá a opção efetuar login.

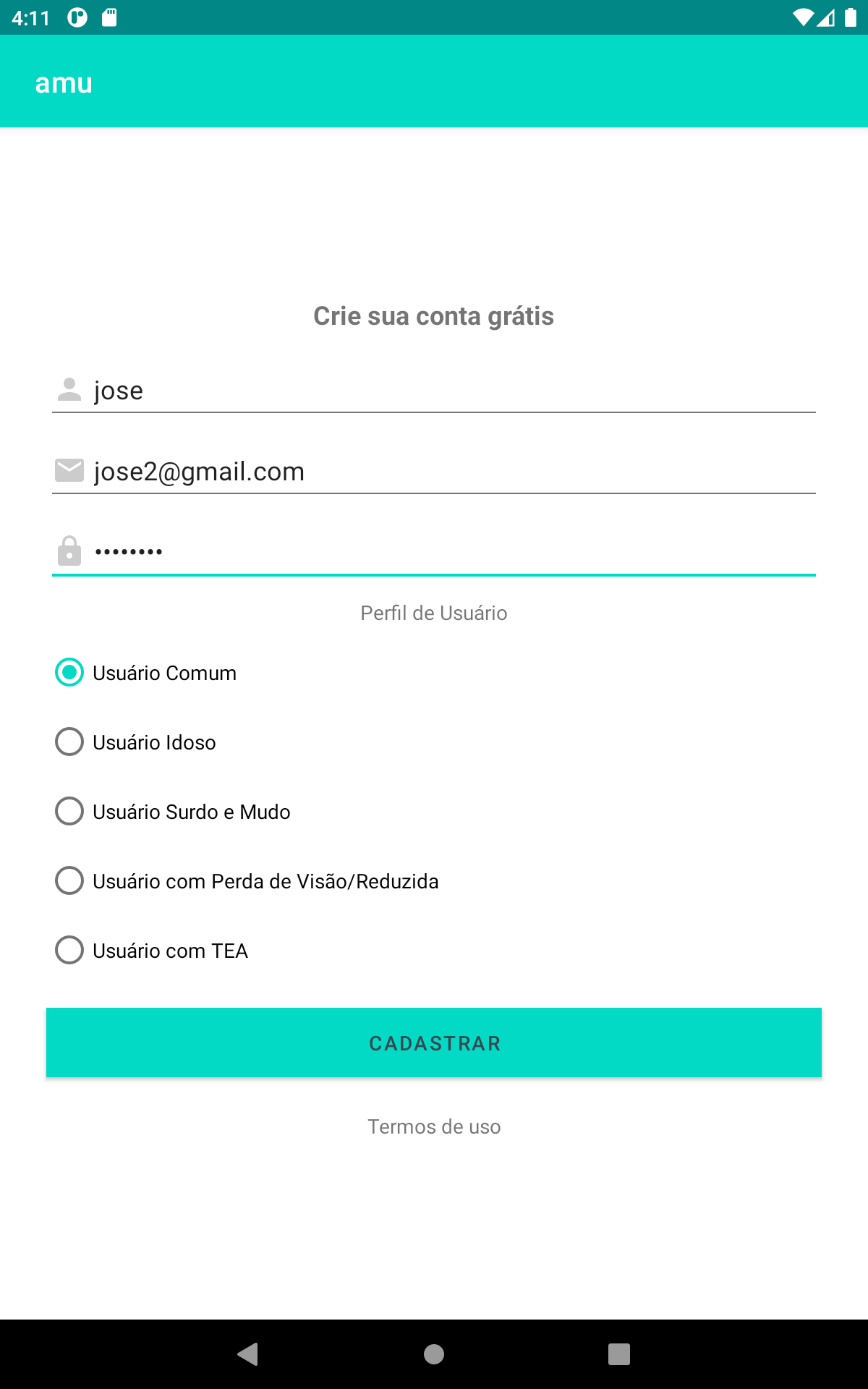
Tanto no primeiro caso quanto no segundo caso o aplicativo exibirá a janela principal.

Quando o usuário já tiver feito login pelo menos uma vez o aplicativo utilizará o recurso FirebaseAuth 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 login e integração para os usuários finais. Ele oferece uma solução de identidade completa, compatível com contas de e-mail/senha, autenticação por telefone, login do Google, Twitter, Facebook, GitHub. O login por e-mail e senha foi escolhido por apresentar simplicidade e os e-mails serem únicos em qualquer servidor de e-mail. (Referencia 3)

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

**Cadastro de Usuário**

****

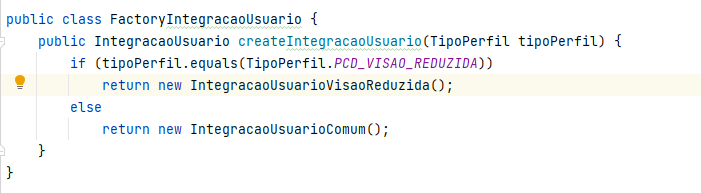
Nesta primeira versão do aplicativo AMU o usuário comum poderá utilizar todas as funcionalidades. O usuário com perda de visão ou reduzida terá através do uso da voz acesso a algumas funcionalidades.

Para usuários idosos, surdos/mudos e TEA não tem 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, para sustentar esse recurso, utilizamos padrões de Projetos. O principal padrão de projeto adotado foi o Strategy.

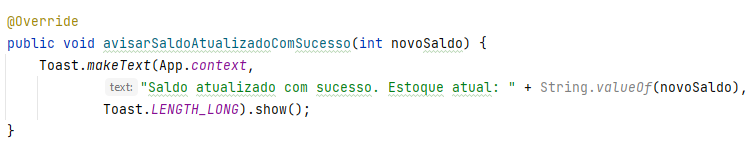
Quando o aplicativo inicia pelo cadastro inicial sabe qual o perfil do usuário. Uma variável global com o nome “integracaoUsuario” então é inicializada por uma fábrica de objetos. Para exemplificar o uso desse recurso, quando o aplicativo dispara um alarme indicando que um medicamento deve ser tomado 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 “integracaoUsuario” que disparará um alarme em audio. 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 IntegracaoUsuarioComum ou IntegracaoUsuarioVisaoReduzida e etc.

O fragmento de código abaixo demonstra o uso de uma fábrica de objeto para ser utilizado conforme o perfil.



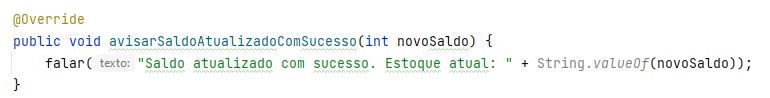
Quanto ao uso do Strategy abaixo temos dois fragmentos de código.

O primeiro fragmento dá suporte ao perfil de usuário comum.

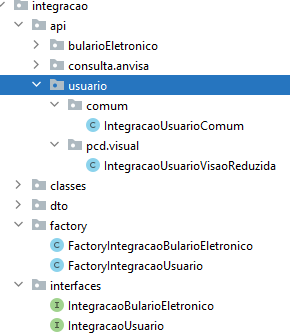


Na imagem acima ao atualizar o saldo de medicamentos uma mensagem é exibida na tela.

O segundo fragmento dá suporte ao usuário com visão reduzida/perda



Para dar suporte a estrutura mencionada acima foi necessário dividir o projeto em camadas. Abaixo temos uma imagem que exibe a estrutura da camada de Api-integração



**Dicionário**

**ESB**

Um ESB, ou barramento de serviço corporativo, é um padrão pelo qual um componente de software centralizado realiza integrações a sistemas back-end (e conversões de modelos de dados, conectividade profunda, roteamento e solicitações) e disponibiliza essas integrações e conversões como interfaces de serviço para reutilização por novos aplicativos. O padrão ESB é geralmente implementado usando um tempo de execução de integração e um conjunto de ferramentas especialmente projetados que garantem a melhor produtividade possível.

**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. A Figura 1.4 apresenta um diagrama que mostra a estrutura básica do padrão.

**Referências**

1. Design Patterns com Java – Projeto orientado a objetos guiado por padrões
   * Pagina 233
2. Design Patterns com Java – Projeto orientado a objetos guiado por padrões.
   * Pagina 200
3. Autenticação no Utilizando Firebase
   * Acessado em 28/12/2022
   * <https://firebase.google.com/products/auth?gclid=CjwKCAiA76-dBhByEiwAA0_s9To21pNpQy3x9y79Uyl8YgNx3iwk0U0Ox8dx45IfKmPpNjXxG22eVhoCC60QAvD_BwE&gclsrc=aw.ds>
4. Uso de Polimorfismo em java
   * Acessado em 28/12/2022
   * <https://www.devmedia.com.br/uso-de-polimorfismo-em-java/26140>
5. Definição de Barramento de Serviços
   * Acessado em 07/12/2022
   * <https://www.ibm.com/br-pt/cloud/learn/esb#:~:text=Um%20ESB%2C%20ou%20barramento%20de,como%20interfaces%20de%20servi%C3%A7o%20para>