iBeauty – Aplicativo *Mobile* de serviços de beleza

Geovana Cordeiro de Oliveira[1], Millena Moura dos Santos[1], Victória Luísa Teixeira Lemos[1]

[1]Ciência da computação – Instituto de Educação Superior de Brasília SGAS Quadra 613/614 - Asa Sul, Brasília - DF, 70200-730

{geovana.oliveira, millena.santos, victoria.lemos}@iesb.edu.br

Abstract. *iBeauty consists of an application focused on beauty and aesthetics, where beauty salons register, offering various services, and users register with the intention of looking for the perfect salon/professional to meet their needs at the moment, in addition to being able to search them through various filters, such as: shorter distance from your home, home services and request for scheduling services.*

Resumo. *O iBeauty consiste em um aplicativo voltado para a beleza e estética, onde salões de beleza se cadastram, oferecendo diversos serviços, e os usuários se cadastram na intenção de procurar o salão/profissional perfeito para atender a sua necessidade no momento, além de conseguir procurá-los através de diversos filtros, como: menor distância de sua residência, serviços em domicílio e solicitação de agendamento de serviços.*

1. Introdução

O iBeauty é um aplicativo *mobile* que foi desenvolvido tanto para os profissionais de beleza, estética e cuidados pessoais, quanto para os clientes que procuram esses serviços. O aplicativo oferece também, através do *chatbot* integrado, informações de beleza como curiosidades, passo-a-passos simples de cuidados pessoais, entre outros informativos que o cliente deseja saber.

Visto isso, o aplicativo traz facilidade para os usuários encontrarem o salão mais perto de casa, ou o salão com opção de serviços em domicílio, e além disso, dá oportunidade para pequenas empresas se tornarem mais conhecidas, se cadastrando no app e aparecendo nas pesquisas dos clientes.

2. Problema

Com a chegada da pandemia, em decorrência do vírus *SARS-CoV-2*, alguns segmentos do comércio foram afetados de diversas formas. Uma das limitações sofridas pela população foi a proibição do funcionamento de vários serviços, dentre eles: salões de beleza e barbearias. Portanto, a procura de serviços de beleza e cuidados pessoais foi afetada, pois começou a ser considerada um risco à saúde.

Além disso, a rotina da maioria das pessoas mudou, faltando também, tempo para se cuidarem. Por isso, o aplicativo iBeauty chegou, para que a população do Distrito Federal tenha a opção de contratar serviços de estética em domicílio também, não sendo necessário ir a um estabelecimento e diminuindo o risco à sua saúde.

Outro público que pode ser afetado positivamente pela chegada do iBeauty, é o público empreendedor que está chegando agora no ramo de estabelecimentos de beleza. Utilizando o aplicativo *mobile* desenvolvido neste projeto, podem se tornar mais vistos e mais conhecidos, aparecendo nas pesquisas dos clientes, através do mecanismo de busca.

3. Objetivos

3.1. Objetivo geral

O objetivo geral do projeto consiste em desenvolver uma solução para eliminar a dificuldade que a maioria dos usuários têm de encontrar estabelecimentos e fornecedores que ofereçam serviços de beleza, estética e cuidados pessoais. Além de possibilitar a visualização do perfil dos estabelecimentos, dos serviços que são ofertados por aquele fornecedor, a modalidade de atendimento e a solicitação de agendamento para que o atendimento seja realizado, tudo isso com conforto e praticidade.

3.2. Objetivo específico

O objetivos específicos do projeto consistem em:

a) Desenvolver um aplicativo destinado ao cadastro de estabelecimentos de beleza e clientes que procuram esses serviços.
b) Centralizar diversos pontos levados em consideração ao escolher o salão que cuidará de sua beleza, como distância de sua casa, solicitação de agendamento via aplicativo e opção do tipo de atendimento ao serviços, em domicílio ou presencialmente.
c) Aproximar os usuários da plataforma, trazendo conhecimento sobre cuidados pessoais, curiosidades, entre outros assuntos relacionados à beleza e estética, através do *chatbot* implementado no aplicativo.
d) Dar oportunidade à pequenas empresas se tornarem conhecidas, se cadastrando no iBeauty e ganhando visibilidade ao aparecerem como sugestão nas pesquisas dos clientes.

4. Referencial teórico

Algumas tecnologias e ferramentas foram utilizadas no desenvolvimento do aplicativo iBeauty, para que alguns recursos pudessem ser usados. Dentre essas ferramentas e tecnologias, estão incluídos os tópicos de linguagem de programação, arquitetura, banco de dados e a ferramenta de *chatbot*.

4.1. Linguagem *Kotlin*

O aplicativo iBeauty foi desenvolvido utilizando a linguagem *Kotlin*. *Kotlin* é uma linguagem com tipagem estática, ou seja, o tipo das variáveis declaradas sempre será seguido independente do conteúdo da mesma. Com isso, a linguagem de tipagem estática acaba sendo mais rigorosa, por exemplo ao tentar mudar o tipo da variável, não é permitido.

Apesar da rigidez, esse tipo de linguagem traz benefícios também, como segurança e melhor performance: Quanto à segurança, linguagens como *Kotlin* tendem a deixar os programadores mais cientes dos erros que foram cometidos inadvertidamente ao escrever o código e avisá-los no início da compilação, em vez de observá-los durante a execução do programa. Quanto à performance, como citado anteriormente, o *Kotlin* impede que o tipo da variável seja modificado, evitando assim, comportamentos imprevisíveis e defeitos não intencionais do sistema.

4.2. Arquitetura MVVMi

O desenvolvimento de aplicativos para o sistema operacional *Android* não é algo trivial. MVVM (*Model-View-ViewModel*) é um padrão de software arquitetural, isto é, uma solução genérica e reutilizável para resolver um problema que acontece numa arquitetura de um software dado um contexto.

O objetivo do MVVM é prover uma separação de responsabilidades, entre a view e sua lógica, com isso benefícios são agregados, como o aumento da testabilidade da aplicação. O padrão MVVM é baseado no padrão MVC - *Model-View-Controller*. Estruturalmente, uma aplicação que usa o padrão MVVM consiste basicamente em três componentes principais: o modelo (*Model*), a visão (*View*) e a *ViewModel*.

No componente *Model*, a camada de modelo é a lógica de negócios que impulsiona a aplicação e quaisquer objetos de negócios. A *View* é a interface do usuário e a *ViewModel* age como a cola em aplicações MVVM. Já a *ViewModel* coordena as operações entre a *View* e a *Model*.

Nesse tipo de arquitetura, um dos princípios é ter uma arquitetura limpa que possa ser altamente testada em unidades, tendo a separação de interesses como prioridade máxima. Portanto, foi adicionada outra camada entre o *ViewModel* e o repositório, responsável por toda a lógica de negócios específica da função. Isso leva a um *ViewModel* simplificado que é responsável apenas pela lógica de apresentação. Essa nova camada é chamada de *Interactor* e a nova sigla utilizada para essa nova arquitetura é MVVMi.

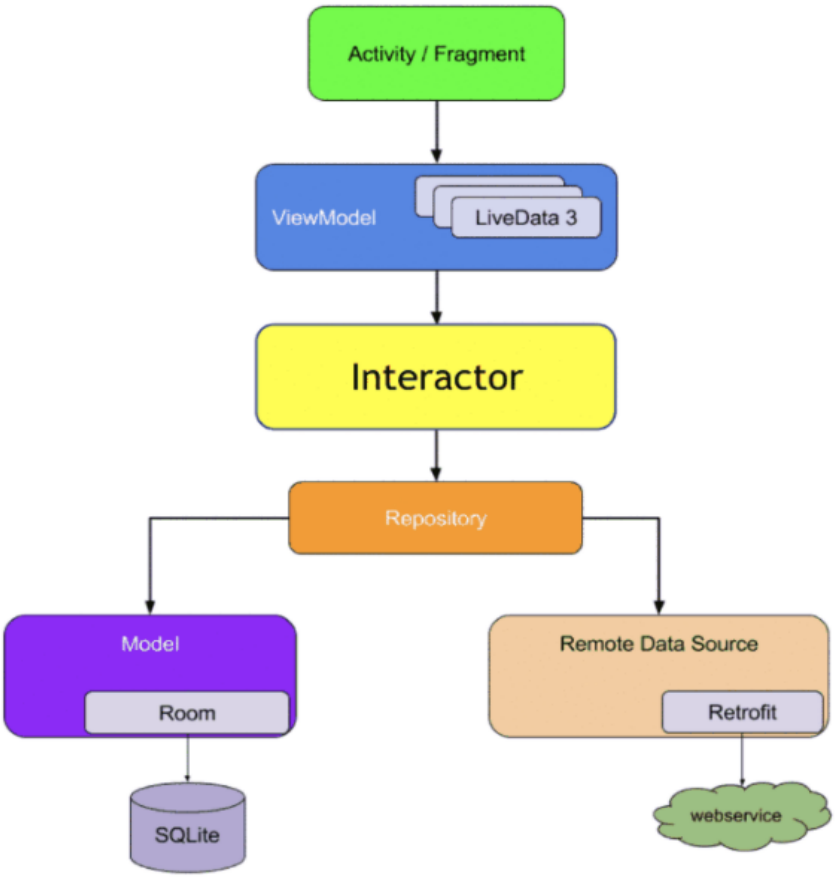

Figura 1. Arquitetura MVVMi.

4.3. Firebase

O Firebase é uma plataforma de desenvolvimento de aplicativos móveis e web, desenvolvida pelo Google. Essa plataforma auxilia na expansão da base de usuários da aplicação que está sendo desenvolvida. Além disso, para uso inicial, essa plataforma é gratuita, cobrando apenas para níveis mais avançados de aplicativos.

Neste aplicativo móvel foi utilizada a plataforma Firebase, tanto para autenticar os usuários do aplicativos, como para centralizar a base de dados de usuários cadastrados na aplicação, tendo como objetivo também, melhorar o rendimento das aplicações com a implementação de diferentes funcionalidades.

4.3.1. Firebase Authentication

O Firebase Authentication é um recurso da plataforma Firebase, utilizado para fornecer a autenticação para os usuários de uma aplicação. Ele oferece autenticação de diversas formas, como via email e senha, via conta Google, Facebook, entre outros.

4.3.2. Firebase Realtime Database

O Firebase Realtime Database é um banco de dados que a plataforma Firebase oferece, e é hospedado na nuvem. Os dados são armazenados em formato JSON (JavaScript Object Notation), e são sincronizados com todos os clientes conectados em tempo real.

4.4. API REST

API (*Application Programming Interface*, ou Interface de Programação de Aplicativos) é um conjunto de normas de desenvolvimento, que fazem parte de uma interface, e que a partir delas, é possível criar softwares, aplicativos, entre outros.

A API REST, também conhecida como API RESTful, é uma interface de desenvolvimento de aplicações (API ou API da Web) que está em conformidade com as restrições do estilo de arquitetura REST, e que permite a interação com os serviços da Web RESTful. REST é a abreviatura em inglês para Transferência Representacional de Estado. A arquitetura REST foi criada pelo cientista da computação Roy Fielding.

4.5. Google Maps

Google Maps é um serviço que oferece o recurso gratuitamente, via API, de mapas/localização. Esse recurso permite aos usuários visualizarem o mundo real por meio de mapas estáticos ou interativos com base nos dados do Google Maps, facilitando a ida do usuário ao local solicitado.

4.6. DialogFlow

Dialogflow é uma plataforma de processamento de linguagem natural que permite projetar e integrar facilmente interfaces de usuário de conversação com aplicativos móveis, aplicativos da web, dispositivos, robôs, sistemas interativos de resposta de voz e muito mais. Com esta plataforma, é possível fornecer aos usuários maneiras novas e envolventes de interagir com a aplicação.

O Dialogflow pode analisar vários tipos de entrada de clientes, incluindo texto ou entrada de áudio (como de um smartphone ou gravação), sendo capaz de responder aos clientes de várias maneiras e por meio de texto ou fala sintetizada.

4.7. Requisições HTTP

HTTP é o protocolo que define as regras para a comunicação entre o cliente e servidor, gerenciando a troca de dados na comunicação, são enviadas requisições conhecidas como request e o servidor responderá as respostas conhecidas como responses. Sua estrutura é simples e altamente extensível, o mecanismo de enquadramento adiciona uma camada intermediária entre a sintaxe e o protocolo de transporte subjacente, utilizando os métodos GET, usado quando o cliente deseja obter recursos e POST usado quando o cliente deseja enviar dados indicando para o servidor qual a ação que o cliente deseja realizar.

4.7. S.O.L.I.D.

O termo S.O.L.I.D. é um acrônimo, que se refere a cinco princípios da programação orientada a objetos que facilitam no desenvolvimento de software, tornando-os fáceis de manter e estender. Esses princípios ajudaram a escrever códigos mais limpos, separando responsabilidades, diminuindo acoplamentos e estimulando o reaproveitamento do código.

Este acrônimo é formado pelos seguintes conceitos:

- Princípio da responsabilidade única (S);
- Princípio aberto-fechado (O);
- Princípio da substituição de Liskov (L);
- Princípio da segregação da interface (I);
- Princípio da inversão de dependência (D).

Figura 2. Demonstração dos princípios do S.O.L.I.D.

4.9. Heroku

O Heroku é uma plataforma de serviços em nuvem (PaaS - Plataform as a service) que suporta vários tipos de linguagem de programação, e possibilita aos desenvolvedores construir, rodar e operar aplicações inteiramente na nuvem com diversas finalidades, sejam elas de hospedagem, testes em produção ou até escalar suas aplicações.

5. Metodologia

O desenvolvimento do aplicativo iBeauty foi feito na linguagem *Kotlin*, sendo utilizada a IDE *Android Studio*, que é um ambiente para desenvolvimento de aplicações para a plataforma *Android*. A divisão de telas foi feita da seguinte forma: interfaces antes da autenticação, e telas após a autenticação no aplicativo. Sendo assim, foram criadas duas *Activity*, uma para cada situação citada, e todo o resto do aplicativo foi desenvolvido por fragmentos. Para cada tela, foi criado um fragmento, que é inflado sempre que a *Activity* o chama.

Neste projeto, foi utilizado o Firebase Realtime Database, para que fosse construído a base de dados da aplicação, contendo dados e informações do cliente e do estabelecimento. No desenvolvimento da base de dados, na parte do cliente, o banco de dados foi alimentado

através do cadastro e autenticação. A autenticação foi feita usando o Firebase Authentication, através do método de e-mail e senha. No momento que a autenticação é feita com sucesso, a tela de dados pessoais é apresentada, para que o cliente insira os dados solicitados (Nome, data de nascimento, endereço, CEP e telefone), sendo todos os campos obrigatórios. A validação dos campos foi feita, e com isso, caso o campo não tenha sido alimentado, é apresentado um alerta no canto direito do campo de texto.

Já na base de dados de estabelecimentos, foi cadastrado diretamente na plataforma Firebase através de um *import* de um arquivo em formato JSON, contendo os seguintes atributos relacionados ao estabelecimento: Nome, cidade, longitude, latitude, descrição, instagram, facebook, e-mail, telefone, se oferece serviços a domicílio ou não, URL da logo e a lista de serviços oferecidos pelo estabelecimento/profissional.

Após o cliente fazer o seu cadastro e login, ele será direcionado para a primeira tela do aplicativo: a *Home*. A *Home* do aplicativo traz alguns pontos durante a rolagem da tela, que podem ser relevantes para o cliente que está navegando pela aplicação, como a área de busca de estabelecimentos e a área de explorar onde são apresentadas algumas fotos relacionadas à beleza.

No rodapé de todos os fragmentos inflados após a autenticação do cliente, é apresentado um menu de navegação contendo cinco ícones representando principais funcionalidades do app, para que o usuário possa acessá-las de forma mais rápida, independente de qual tela ele está naquele momento. Os cinco ícones são: uma casa, que direciona para a Home; uma lupa, direcionando para a tela de busca de estabelecimentos; um balão de mensagens, que dará acesso ao chatbot desenvolvido; uma câmera fotográfica, que irá dar acesso à tela de Explorar, que traz fotos relacionadas à beleza, como já citado anteriormente; e por último um ícone de uma silhueta de rosto, que leva à tela de perfil do cliente, que contém os dados base do cliente, e a opção de fazer *logout* no aplicativo, ou seja, sair da sua conta.

O chatbot do aplicativo iBeauty foi implementado utilizando o DialogFlow, que é uma plataforma que oferece o recurso de criação de fluxos de conversação de um usuário com um bot. Essa funcionalidade foi implementada do seguinte modelo: é utilizado como um recurso para o usuário perguntar sobre assuntos de beleza, por exemplo: usuário pergunta sobre unhas, e o bot traz um link sobre unhas.

O deploy foi feito usando o Heroku, que é uma plataforma de nuvem que faz deploy de várias aplicações de back-end. Diante disso, o back-end do chat foi implementado de acordo com a arquitetura MVVMi, usando as classes *Interactor*, *Repository* e *ViewModel* para sua funcionalidade.

O iBeauty oferece um recurso de busca de salões através da tela de busca. Nessa tela são listados os estabelecimentos de acordo com o que o usuário digitou na barra de pesquisa, e então o usuário poderá escolher qual estabelecimento ele quer ver o perfil, localização, contatos, e solicitar um agendamento de horário. Utilizando o conector Adapter, esse recurso foi implementado no aplicativo.

6. Resultados obtidos

7. Conclusão

A proposta deste Projeto Integrador foi desenvolver uma aplicação para dispositivos móveis, com o objetivo de trazer uma solução para eliminar a dificuldade que a maioria dos usuários têm de encontrar estabelecimentos e fornecedores que ofereçam serviços de beleza, estética e cuidados pessoais.

Com o desenvolvimento do aplicativo iBeauty, considera-se que será possível contribuir com uma grande parcela de comerciantes e com a sociedade, trazendo benefícios para os usuários e visibilidade para os empreendedores que buscam ofertar seu serviço de forma dinâmica, diferenciada e democrática.

Diante do exposto, os clientes e empreendimentos já cadastrados, estarão habilitados a identificar interesses em comum, assim como expor a ementa dos serviços ofertados e a forma de atendimento com que o serviço poderá ser realizado.

8. Referências bibliográficas

Dias, E. (2020) “Kotlin: FAQ - Conheça melhor essa linguagem”, https://www.devmedia.com.br/kotlin-faq-conheca-melhor-essa-linguagem/40754.

Macoratti, J. C. “Compreendendo o padrão MVVM : Model-View-ViewModel”, http://www.macoratti.net/.

Content, R. R. (2019) “Conheça Firebase: a ferramenta de desenvolvimento e análise de aplicativos mobile”, https://rockcontent.com/br/blog/firebase/.

Simpson, L. (2017) “Clean Architecture with MVVMi, Architecture Components & RxJava”, https://medium.com/@thereallukesimpson/clean-architecture-with-mvvmi-architecture-components-rxjava-8c5093337b43.

Loss, F. S.; Dal Ponte, M. S. (2015) “Desenvolvimento de um aplicativo web para salões de beleza utilizando Ruby on Rails”, http://repositorio.utfpr.edu.br/jspui/handle/1/15478.

https://medium.com/desenvolvendo-com-paixao/o-que-%C3%A9-solid-o-guia-completo-para-voc%C3%AA-entender-os-5-princ%C3%ADpios-da-poo-2b937b3fc530.

https://miro.medium.com/max/700/1*YYOngUyaO6uMOEtwWvy_Ig.gif

https://miro.medium.com/max/2870/1*NP2y1DuTdJEXb-S3baFvyA.png

https://firebase.google.com/docs/android/setup?hl=pt-br

https://developer.android.com/guide?hl=pt-br