

SERVICE ADVISOR: UM APLICATIVO MÓVEL PARA MEDIAÇÃO ENTRE SOLICITANTES E PRESTADORES DE SERVIÇOS

Cairo Araújo Santos
Graduando em Sistemas de Informação – Uni-FACEF
cairo.a.santos@gmail.com

Miguel Gomes Aleixo Ferreira Lima
Graduando em Sistemas de Informação – Uni-FACEF
miguelaleizo@gmail.com

Orientador: Prof. Dr. Daniel Facciolo Pires
Doutor em Física Médica – Uni-FACEF
daniel@facef.br

RESUMO

Diariamente inúmeras pessoas podem se beneficiar de recursos da tecnologia da informação e comunicação, dentre eles se destacam as aplicações móveis que surgiram para apoiar a produtividade e proporcionar maior facilidade na rotina de seus usuários. Este artigo tem como objetivo o desenvolvimento de uma aplicação que auxilia o encontro de solicitantes e prestadores de serviço. O usuário do tipo solicitante poderá procurar por prestadores de serviço de acordo com a categoria desejada, podendo se comunicar com o prestador e avaliar o serviço eventualmente contratado. O usuário do tipo prestador pode oferecer seus serviços para que fiquem disponíveis para solicitações, podendo aumentar a visibilidade do seu trabalho. Este artigo utilizou-se de técnicas e métodos de desenvolvimento para aplicações móveis nativas, bem como criou artefatos da engenharia de software para processos de modelagem e documentação do sistema. Os resultados encontrados mostraram que foi possível desenvolver uma solução na forma de um *MVP (Minimum Viable Product)* e a implementação de novas funcionalidades podem torná-lo um produto comercializável.

Palavras-chave: Aplicações móveis. Prestadores de serviço. Solicitações.

ABSTRACT

Every day countless people can benefit from information and communication technology resources, among which stand out the mobile applications that emerged to support productivity and provide greater ease in the routine of its users. This article aims to develop an application that helps to find requesters and service providers. The requesting user can search for service providers according to the desired category, being able to communicate with the provider and evaluate the service eventually contracted. The service provider type user can offer his services so that they are available for requests, which can increase the visibility of his work. This article used development techniques and methods for native mobile applications, as well as created software engineering artifacts for system modeling and documentation processes. The results found showed that it was possible to develop

a solution in the form of an MVP (Minimum Viable Product) and the implementation of new features can make it a marketable product.

Keywords: *Mobile applications. Service providers. Requests.*

1 INTRODUÇÃO

Por meio da análise e percepção empírica dos autores, foi possível observar que atualmente, muitas pessoas em sua rotina diária necessitam de um serviço específico, porém não são qualificadas para executá-lo, ao passo que existem outras pessoas que possuem a capacidade de realizá-lo, contudo na maior parte dos casos, subsiste o desencontro de ambos, devido a fatores como falta de recursos de comunicação, visibilidade e etc. Isso implica que solicitantes tenham dificuldade de encontrar prestadores de serviços eficientes e aos prestadores dificuldades para conseguir clientes para oferecer um serviço de qualidade.

Posto ao contexto atual, surgiu a ideia de desenvolver uma aplicação para *smartphones* que facilita o encontro desses dois tipos de público, os solicitantes e os prestadores de serviço.

O objetivo deste artigo é desenvolver um aplicativo para dispositivos móveis intitulado *Service Advisor*, que apoia e facilita o encontro de pessoas que possuam habilidades em serviços domésticos ou em outro ramo com pessoas que necessitam dos mesmos.

Os artefatos da engenharia de software são utilizados para auxiliar no desenvolvimento do aplicativo, deste modo foram criados diversos diagramas e documentos como: 5W2H, BPMN, Canvas, Caso de Uso, Diagrama de Classe, EAP, Modelagem de Dados e Requisitos gerais com o intuito de apresentar uma visão geral do projeto e também apoiar a organização do mesmo.

Visando maior abrangência pelo aplicativo móvel, o desenvolvimento foi executado em *React Native*, onde o aplicativo gerado pode ser utilizado nos sistemas operacionais *Android* e *iOS*. Na comunicação entre a aplicação e o banco de dados, foram criadas *APIs* em *NodeJS*. Para armazenar e controlar os dados dos usuários do sistema, foi criado um banco de dados por meio do SGBD denominado *PostgreSQL*.

Através desta aplicação o usuário conseguirá selecionar de acordo com as categorias, qual serviço deseja solicitar para que o prestador possa realizá-lo. As categorias não são limitadas, podendo ter uma grande variedade de serviços, fazendo com que os prestadores ganhem maior visibilidade sob o seu trabalho. Com o sistema de avaliação, será possível que os prestadores sejam qualificados de acordo com o serviço proporcionado, para que próximos solicitantes consigam escolher melhor os serviços de acordo com cada avaliação.

Por meio do aplicativo, o prestador publicará os serviços que está apto a realizar, especificando como ele será realizado, em quanto tempo e a faixa de preços que poderá ser cobrado. As pessoas que necessitam do serviço irão solicitar através da postagem do prestador, a parte de pagamento será realizada diretamente entre o solicitante e o prestador, sem intermédio do aplicativo. Após a finalização do serviço, o solicitante avaliará o serviço que foi realizado.

Na seção 2 é apresentado o referencial teórico com as vantagens das aplicações móveis e todas as tecnologias que estão envolvidas no desenvolvimento do aplicativo. Já na seção 3 é apresentado o tema Empreendedorismo, mostrando o modelo de *startup* e o modelo de negócio *Canvas*, que foi desenvolvido durante o projeto. Posteriormente, na seção 4 são apresentados os artefatos de modelagem e análise necessários para o desenvolvimento, como o BPMN, Diagrama de Caso de Uso e a Modelagem de Dados. A seção 5 é apresentada uma análise do que foi percorrido no artigo juntamente com a apresentação do resultado obtido, com partes da implementação do código e telas do aplicativo. E por fim a seção 6 é abordada a conclusão juntamente com os objetivos alcançados e possíveis melhorias futuras.

2 REFERENCIAL TEÓRICO

2.1 APLICAÇÕES MÓVEIS

Atualmente as aplicações móveis vem adquirindo cada vez mais força no mercado, principalmente no ramo de jogos e aplicativos que facilitam o dia-a-dia do usuário. Gasparotto (2019) diz que a grande vantagem dos aplicativos de um *smartphone* é a capacidade que os mesmos possuem de se comunicar com os dispositivos nativos do aparelho, como câmera, microfone, *GPS* entre outros. Isso faz com que a aplicação fique mais completa. Um grande exemplo são os aplicativos como *Facebook* e *Instagram*, onde o usuário consegue capturar uma foto do seu próprio dispositivo móvel (*smartphone*), e publicá-la rapidamente em sua rede social, sem a necessidade de uma máquina fotográfica e uma transferência de arquivos em seu computador pessoal para a publicação da fotografia, como acontecia anteriormente.

Complementando as aplicações móveis, segundo Yeepley (2016), a primeira coisa a levar em conta, é que um aplicativo deve ser eficaz e eficiente. Isto significa que você deve dar ao usuário o que ele está procurando de forma fácil e rápida. Um aplicativo deve atender às necessidades no menor tempo possível, sem exigir que o usuário tenha grande conhecimento do seu uso ou exigir um longo processo de aprendizagem. Isso faz com que a aplicação ganhe notoriedade entre outras existentes no mercado atual.

2.2 REACT NATIVE

Os *frameworks* fazem parte da rotina de muitos funcionários que atuam na área de tecnologia da informação, pois com a alta demanda de desenvolvimento de *softwares* é necessário uma ferramenta em que o desenvolvedor possa ser mais rápido e produtivo. No cenário atual encontra-se uma gama de *frameworks* específicos para cada propósito, dentre os mais conceituados está o *React Native*.

De acordo com Becker (2020), *React Native* é um *framework* baseado no já aclamado *React*, desenvolvido pela equipe do *Facebook*, que possibilita o

desenvolvimento de aplicativos para celulares, tanto para *Android*, como para *iOS*, utilizando apenas a linguagem de programação, o *Javascript*.

Antes do surgimento do *React Native*, desenvolver aplicações para celulares em *Android* ou *iOS* era algo relativamente complexo cita Becker (2020), pois além de ter que aprender as linguagens dos respectivos sistemas operacionais *Objective-C* (*iOS*) e *Java* (*Android*), o desenvolvedor não aproveitava praticamente nada do código de uma plataforma para outra, fazendo assim com que as empresas contratassem um time de desenvolvimento para *Android* e outro para *iOS*, tornando o projeto muito lento e caro. Porém, com o *React Native*, o código pode ser reaproveitado em até 100% entre as plataformas, podendo fazer com que o custo e a duração do projeto caiam pela metade.

O *React Native* é também considerado um *framework* inovador, pois todo o código desenvolvido é convertido para a linguagem nativa do sistema operacional e também possui diversas funcionalidades muito interessantes que aumentam a praticidade e a produtividade do desenvolvimento, argumenta Becker (2020). Uma delas é o *Hot Reloading*, que faz com que o programa fique rodando em tempo desenvolvimento, e a cada atualização no código uma versão nova do arquivo modificado é injetado na aplicação, levando menos de 1 segundo para atualizar. Para o desenvolvimento mobile isso é um grande passo, pois em outros *frameworks* nativos, a cada mudança no código, a aplicação precisava ser recompilada por completo, levando muito mais tempo.

2.3 REDUX

Segundo Devmedia (2020), *Redux* é uma implementação da arquitetura *Flux*, que propõe uma solução ao problema de compartilhamento de estados nas aplicações que desejam criar um fluxo unidirecional de dados, fazendo com que esses dados sejam consumidos por qualquer parte da aplicação. Em suma pode se dizer que o *Redux* é um controlador geral de estados para sua aplicação, esclarece Kröger (2017).

Esse *framework* é fragmentado em 3 partes fundamentais para o seu funcionamento, são eles *store*, *reducers* e *actions*. Kröger (2017) discorre sobre eles da seguinte forma:

A *store* é o nome dado para o conjunto de estados da sua aplicação, ela é semelhante a um grande centro de informações, que possui disponibilidade para receber e entregar exatamente o que o seu componente requisita, seja uma função ou até a própria informação. Tecnicamente, a *store* é um objeto em *Javascript* que possui todos os estados dos seus componentes.

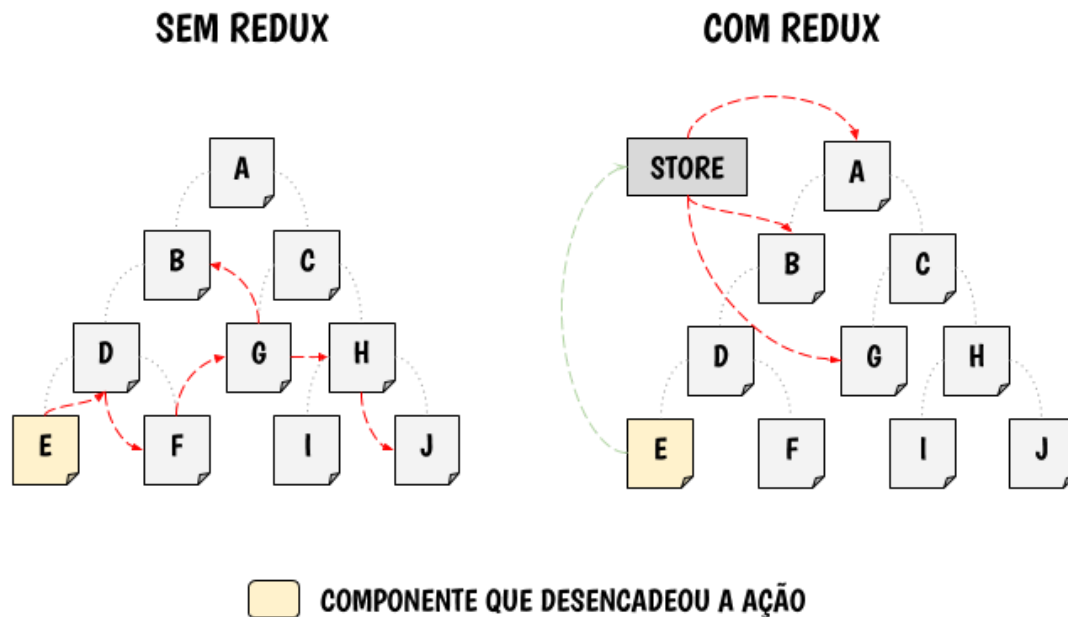
Sobre os *reducers* o autor diz que cada dado da *store* deve ter seu próprio *reducer*, ele é encarregado de lidar com todas as ações, como algum componente da aplicação pedindo para alterar algum dado da *store*.

Por fim as *actions* são responsáveis por requisitar algo para o *reducer*, elas são funções puras e por esse motivo devem apenas enviar os dados ao *reducer*, nada além disso.

Um exemplo das vantagens de se utilizar o *Redux* em sua aplicação é que em certos casos, dois componentes vão tentar dividir os mesmos dados, o problema é que, se esses componentes não estiverem próximos um do outro, o

componente “ancestral” mais próximo terá que repassar os dados através de vários componentes intermediários afirma Ianakiara (2018). Além disso o *Redux* conta com uma ampla documentação sobre o *framework* e também há uma comunidade forte e ativa de desenvolvedores, que podem oferecer o suporte que você precisa para a implementação do *framework*, afirma o autor.

Figura 1 - Compartilhamento de estado em uma aplicação com e sem *redux*



Fonte: DEVMEDIA, 2020.

2.4 NODEJS

NodeJS é uma plataforma para construir aplicações escaláveis de alta performance utilizando apenas *Javascript*. Com um *loop* de eventos, o *NodeJS* interpreta, em uma única *thread*, as requisições de forma assíncrona em vez de sequenciais, fazendo com que não haja bloqueios, e isso o torna incrivelmente rápido, perfeito para lidar com um número muito alto de requisições, comenta Santos (2016). Por esses motivos que apesar de novo, o *NodeJS* já é utilizado por grandes empresas no mercado de tecnologia, como *Netflix*, *Uber* e *LinkedIn*, segundo Tessis (2017) em um estudo recente.

De acordo com Escudelario (2018), em virtude de seu modelo assíncrono consumir pouquíssimos recursos do *hardware*, o principal uso do *NodeJS* está na construção de APIs (*Application Programming Interface*), que são um tipo de “ponte” que conecta as aplicações, como por exemplo uma conexão entre um site e um banco de dados frisa Fernandes (2018). As APIs podem ser utilizadas para qualquer tipo de negócio ou nicho de mercado. A maioria das pessoas não sabem do que se trata uma API, pois são operadas apenas por

profissionais de programação e invisíveis ao usuário comum, que é capaz de ver apenas a interface gráfica dos *softwares* e aplicativos, reitera Fernandes (2018).











2.5 POSTGRESQL

Sistema Gerenciador de Banco de Dados (SGBD) é um conglomerado de softwares construídos para disponibilizar uma interface capaz de administrar todas as informações de determinada base de dados. Scudero (2016) alega que através dos SGBDs o usuário consegue controlar, organizar, acessar e proteger todas as informações de sua empresa.

Soares (2020) mostra que o SGBD intitulado PostgreSQL ocupa o quarto lugar em um estudo realizado pelo site *db-engines.com*, que classifica sistemas de gerenciamento de banco de dados conforme a sua popularidade. Além disso, de acordo com os autores desse *ranking* (DB-ENGINES, 2020), a lista de classificação é atualizada mensalmente.

Figura 2 - Lista de classificação de SGBDs conforme sua popularidade

350 systems in ranking, January 2020

Rank			DBMS	Database Model	Score		
Jan 2020	Dec 2019	Jan 2019			Jan 2020	Dec 2019	Jan 2019
1.	1.	1.	Oracle +	Relational, Multi-model 	1346.68	+0.29	+77.85
2.	2.	2.	MySQL +	Relational, Multi-model 	1274.65	-1.01	+120.39
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model 	1098.55	+2.35	+58.29
4.	4.	4.	PostgreSQL +	Relational, Multi-model 	507.19	+3.82	+41.08
5.	5.	5.	MongoDB +	Document, Multi-model 	426.97	+5.85	+39.78
6.	6.	6.	IBM Db2 +	Relational, Multi-model 	168.70	-2.65	-11.15
7.	7.	8.	Elasticsearch +	Search engine, Multi-model 	151.44	+1.19	+8.00
8.	8.	7.	Redis +	Key-value, Multi-model 	148.75	+2.51	-0.27
9.	9.	9.	Microsoft Access	Relational	128.58	-0.89	-13.04
10.	11.	10.	SQLite +	Relational	122.14	+1.78	-4.66
11.	10.	11.	Cassandra +	Wide column	120.66	-0.04	-2.32
12.	12.	12.	Splunk	Search engine	88.67	-1.85	+7.25
13.	13.	13.	MariaDB +	Relational, Multi-model 	87.45	+0.66	+8.63
14.	14.	15.	Hive +	Relational	84.24	-1.81	+14.33
15.	15.	14.	Teradata +	Relational, Multi-model 	78.29	-0.21	+2.10

Fonte: SOARES, 2020.

Para Matthew e Stones (2005) o PostgreSQL é um excelente Sistema Gerenciador de Banco de Dados pois permite ser utilizado por qualquer uma das principais linguagens de programação como *Java*, *Python*, *PHP* e muitas outras, além de possuir código aberto, ser gratuito para o uso e apresentar inúmeros recursos que beneficiam o dia-a-dia de seus usuários.

Muitos desses recursos estão destinados a ajudar os desenvolvedores a criar aplicativos e os administradores a proteger a integridade dos dados (PostgreSQL, 2020).

3. CONTEXTO EMPREENDEDOR DO PROJETO

Empreender é a capacidade que uma pessoa possui de observar problemas e oportunidades, apresentar soluções e investir recursos na criação de algo positivo para o meio em que convive, afirma Bueno (2019). O mesmo pode ser um negócio, um projeto ou até um movimento que produz mudanças reais e afeta o dia a dia de nossa sociedade, sua essência está na percepção e no aproveitamento das novas oportunidades no âmbito dos negócios. Empreendedor é o indivíduo que sai da zona de conforto e parte para a ação, sua função é colocar em prática novas ideias através da criatividade, resiliência e ousadia.

Nogueira (2019) diz que atualmente a tecnologia da informação entra como um dos ramos mais promissores e flexíveis para o investimento, em razão disto o termo Startup vem crescendo cada vez mais.

3.1 STARTUP

Bicudo (2016) frisa que Startup é uma empresa jovem, com um modelo de negócios repetível e escalável diante da incerteza e das soluções a serem desenvolvidas. Elas não se limitam somente aos negócios digitais, as startups precisam de inovação para não serem consideradas uma empresa de modelo tradicional. Realmente, startup se tornou um termo da moda, e o empreendedorismo tornou-se o sonho de muitas pessoas no Brasil e no exterior. Um erro comum na definição de startups é que elas são apenas empresas de Internet. Não necessariamente, elas são apenas mais frequentes na Internet, porque são muito mais baratas e fáceis de espalhar para criar empresas on-line em comparação com, por exemplo, o agronegócio.

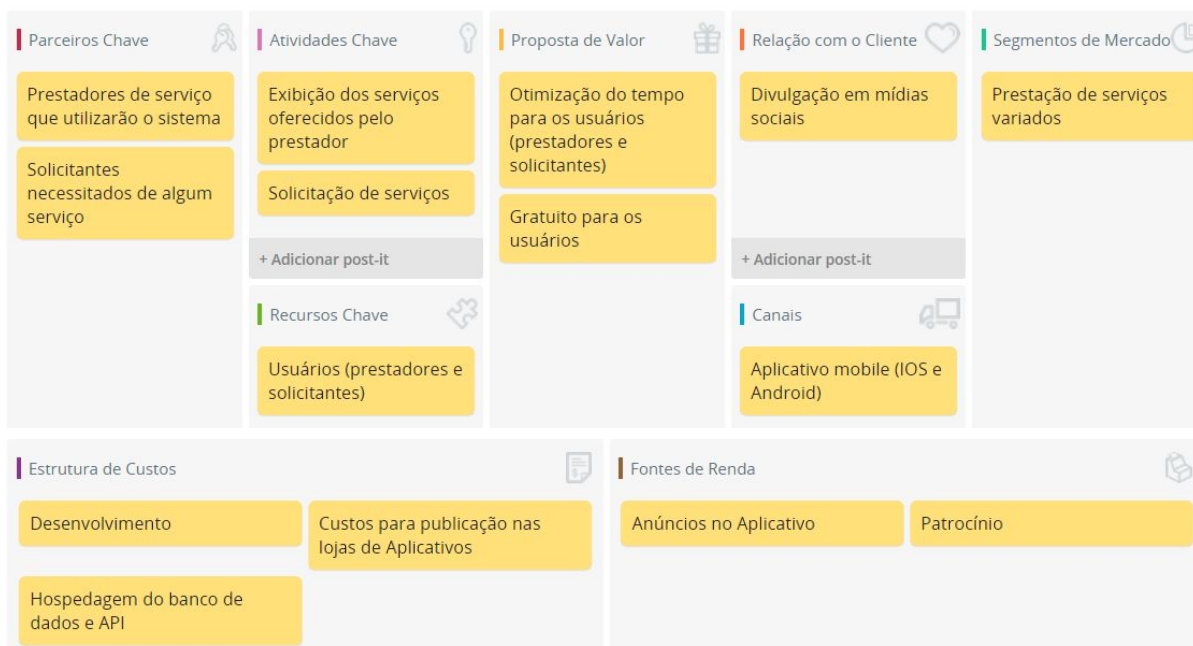
3.2 CANVAS

O Canvas é uma solução simples e eficiente que ajuda o empreendedor a visualizar melhor as questões estratégicas do seu negócio, seu principal objetivo é construir um modelo inovador de planejamento de negócios, que traga praticidade e dinamismo à análise organizacional. Trata-se, de um mapa visual projetado para entender se cada um pilares do negócio obtém a devida a atenção, visto que o método é um padrão capaz de fornecer uma visão prática do formato do modelo, diz Camargo (2019).

Apresentado pelo empreendedor, palestrante, consultor e teórico da administração, Alexander Osterwalder, o Canvas pode ser definido como uma tabela com 9 quadrantes a serem contemplados. A Figura 3 ilustra o modelo Canvas

desenvolvido para este projeto, juntamente com a descrição de cada componente contextualizado ao aplicativo Service Advisor.

Figura 3 - Modelo Canvas do projeto



Fonte: Os autores.

- 1) **Parceiros chave:** o prestador de serviço, que utilizará o aplicativo para divulgar seus serviços, e os solicitantes para buscarem pelo serviço que necessitarem.
- 2) **Atividades chave:** exibir os serviços dos prestadores como uma forma de vitrine virtual, para que eles possam ser solicitados.
- 3) **Proposta de valor:** otimizar o tempo do prestador e do solicitante por meio de uma forma simples e rápida de solicitar um serviço, além de ser gratuito para os usuários, sem cobrança para o prestador de serviços anunciar os seus serviços, e custo zero para o solicitante, além do valor do serviço realizado, que deverá ser pago diretamente para o prestador.
- 4) **Relação com o cliente:** para atingir nosso público alvo, o principal meio utilizado serão as redes sociais onde podemos atingir uma grande quantidade de pessoas rapidamente.
- 5) **Segmentos de mercado:** a aplicação é direcionada aos prestadores de variados e seus clientes em potencial.
- 6) **Recursos chave:** Os usuários do aplicativo, prestadores e solicitantes, que tenham um dispositivo móvel com acesso a internet.
- 7) **Canais:** O nosso canal de comunicação com o cliente será exclusivamente o aplicativo, estando disponível para as plataformas *Android* e *IOS*, por conta do fato de que grande maioria da população tem um celular com acesso a internet.
- 8) **Estrutura de Custos:** um dos custos é o desenvolvimento, que necessita de bastante tempo dos desenvolvedores, temos também gastos com a hospedagem do

banco de dados e das APIs, além do valor que será desembolsado para publicar o aplicativo na *APP Store (IOS)* e *Google Play (Android)*.

9) Fontes de Renda: para obter viabilizar o retorno dos custos do projeto, levamos em conta exibir alguns anúncios no aplicativo ou até mesmo receber um patrocínio de uma empresa querendo expor sua marca.

4 ANÁLISE E MODELAGEM DO SISTEMA

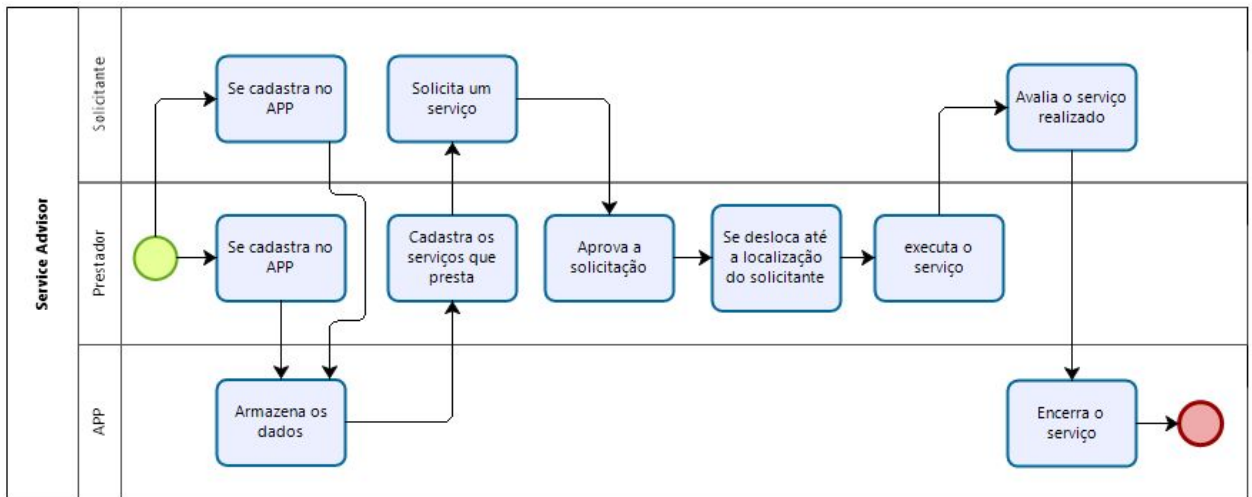
Para Michelazzo (2006), a documentação de software é parte integrante de qualquer sistema, e se torna tão importante quanto às questões de segurança, pois sem a devida documentação, pontos vulneráveis da aplicação não são expostos com facilidade, podendo ocasionar diversos problemas em seu software e até mesmo a sua falência.

A seguir serão apresentados diferentes métodos e diagramas utilizados para o desenvolvimento deste projeto como BPMN, Caso de Uso e Modelagem do Banco de Dados. Outros documentos como a Estrutura Analítica do Projeto (EAP), 5W2H, Diagrama de classe e Requisitos gerais do Sistema estão disponíveis em (LIMA e SANTOS, 2020).

4.1 BPMN

Conforme Oliveira (2018), este modelo foi projetado principalmente para melhorar a comunicação entre departamentos e pessoas, porque constrói e mostra o processo e suas etapas. Porém, seu papel é muito mais do que isso, pode ilustrar o processo de uma maneira diferente. Só podemos mudar o que entendemos e apenas o que podemos visualizar. A notação BPMN especifica o processo de negócios no diagrama, que é fácil de ler para usuários técnicos e usuários de negócios. Os gráficos BPMN são intuitivos e podem representar detalhes complexos do processo. A semiótica BPMN é usada como uma linguagem padrão, eliminando assim a lacuna de comunicação entre a modelagem e a execução do processo. A Figura 4 ilustra o BPMN do projeto Service Advisor.

Figura 4 - BPMN

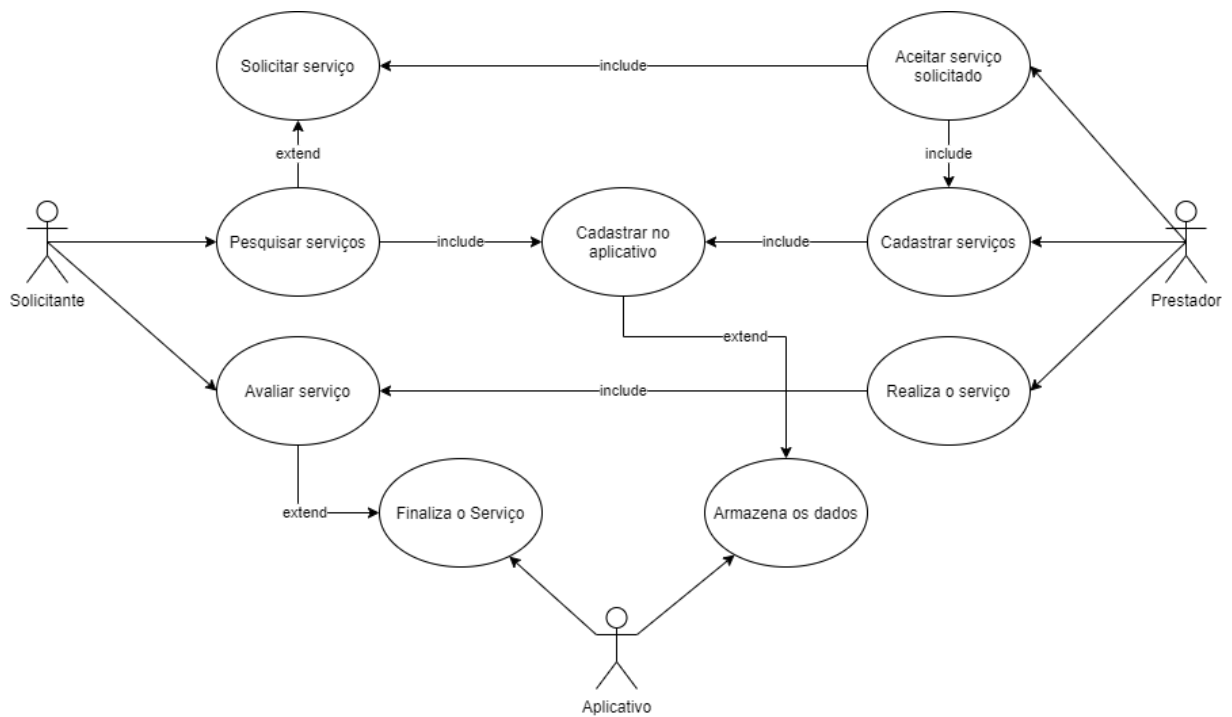


Fonte: Os autores.

4.2 DIAGRAMA DE CASO DE USO

Segundo Vieira (2015), os diagramas de casos de uso ajudam a investigar os requisitos funcionais do sistema e descrevem um conjunto de funções do sistema e sua interação com elementos externos e entre si. Cenário: Ao falar sobre casos de uso, devemos ter em mente o conceito de cenários, ou seja, exemplos de casos de uso. A cena pode ser entendida como uma série de etapas que descrevem a interação entre o usuário e o sistema. Em resumo, os diagramas de casos de uso facilitam a comunicação entre o cliente e o analista, apresentam as principais funções do sistema com o cliente como centro, descrevem os cenários de interação entre o interior e exterior do sistema e concentram-se no usuário. É amplamente utilizado no estágio de coleta de requisitos. A Figura 5 ilustra o Caso de Uso do projeto Service Advisor.

Figura 5 - Caso de Uso



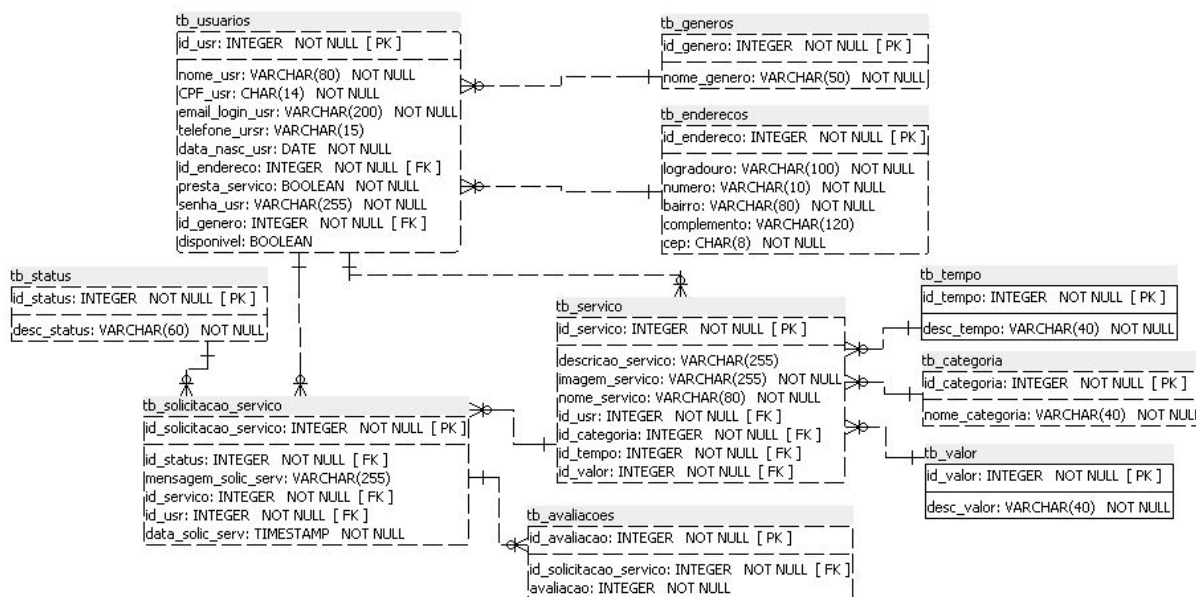
Fonte: Os autores.

4.3 MODELAGEM DE BANCO DE DADOS

A Modelagem de Banco de Dados é crucial para qualquer implementação de software, pois com ela os analistas e desenvolvedores conseguem ter uma visão abrangente geral do projeto.

Para Miranda (2017), um dos principais objetivos da modelagem de dados é representar o ambiente observado, com isso ela auxilia a organizar a forma de pensamento sobre os dados, demonstrando o significado e a aplicação prática dos mesmos. A figura 6 ilustra a modelagem de dados desenvolvida para a implementação do banco de dados da aplicação Service Advisor.

Figura 6 - Modelagem de banco de dados do projeto



Fonte: Os autores.

5 IMPLEMENTAÇÃO DO SOFTWARE

5.1 TELAS DO APLICATIVO E SUAS FUNCIONALIDADES

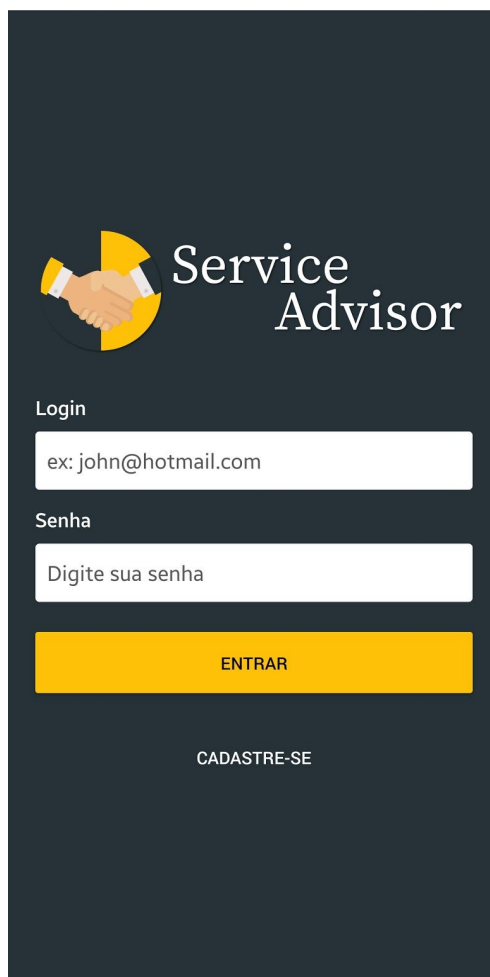
Nesta seção serão apresentadas todas as telas presentes no *Minimum Viable Product* (MVP) do aplicativo móvel nomeado Service Advisor.

A aplicação contempla funcionalidades para ambos usuários, deste modo para fácil compreensão, a narrativa será segmentada em 2 partes, a primeira contendo a visão do solicitante e a segunda a visão do prestador, juntamente com as especificações de cada tela/funcionalidade.

5.1.1 AUTENTICAÇÃO DE USUÁRIO (SOLICITANTE E PRESTADOR)

Para ter acesso a todas funcionalidades da aplicação, primeiramente é necessário que o prestador ou solicitante tenha um usuário antecipadamente cadastrado em nossa base de dados. A Figura 7 ilustra o processo de autenticação, onde o usuário digitará seus dados já cadastrados anteriormente.

Figura 7 - Tela de login



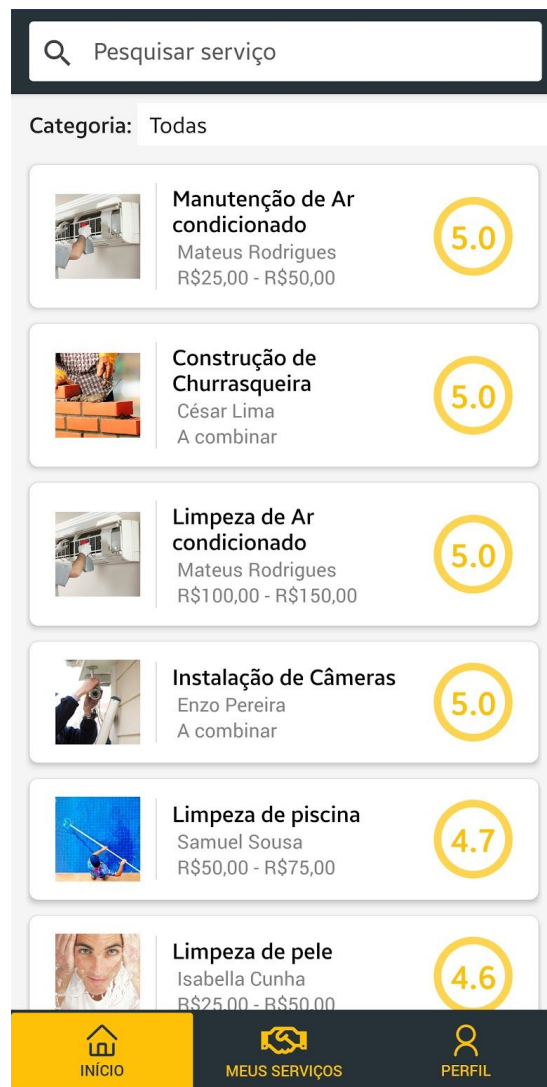
The image shows a login screen for a system called "Service Advisor". The background is dark blue. At the top left, there is a logo consisting of two hands shaking, with one hand in yellow and the other in white. To the right of the logo, the text "Service Advisor" is written in a white serif font. Below the logo and text, there is a "Login" section. It contains two white input fields. The first field is for the email address, with the placeholder text "ex: john@hotmail.com". The second field is for the password, with the placeholder text "Digite sua senha". Below these fields is a large yellow button with the text "ENTRAR" in black. At the bottom of the screen, there is a link that says "CADASTRE-SE" in white.

Fonte: Os autores.

5.1.2 LISTAGEM DE SERVIÇOS (SOLICITANTE)

Para o usuário que deseja solicitar um serviço, o mesmo terá acesso aos cards disponíveis contendo informações básicas como o nome do serviço, nome de quem o realiza, valor estimado e a avaliação baseada em solicitações passadas (Figura 8).

Figura 8 - Tela de consulta de serviços




Fonte: Os autores.

Para melhor experiência do usuário e otimização á busca pelos serviços, a opção de filtragem (Figura 8) é essencial, através dela é possível consultar os serviços separados pelo tipo de categoria, além da opção de pesquisar manualmente, no campo de texto na parte superior da tela de listagem de serviços (Figura 8).

Figura 9 - Tela de serviço detalhada

← Instalação de Câ...



Avaliação: 5.0

Prestador: Enzo Pereira


Tempo: mais de 5 horas

Valor: A combinar

Descrição:
Faço a instalação COMPLETA de sistema de câmeras de segurança, o valor é de acordo com o cômodo e quantidade de câmeras

Mensagem (opcional)

← Instalação de Câ...



Avaliação: 5.0

Prestador: Enzo Pereira

Tempo: mais de 5 horas

Valor: A combinar

Descrição:
Faço a instalação COMPLETA de sistema de câmeras de segurança, o valor é de acordo com o cômodo e quantidade de câmeras

Mensagem (opcional)

As câmeras serão apenas do lado de fora da casa

QUERO ESSE SERVIÇO!

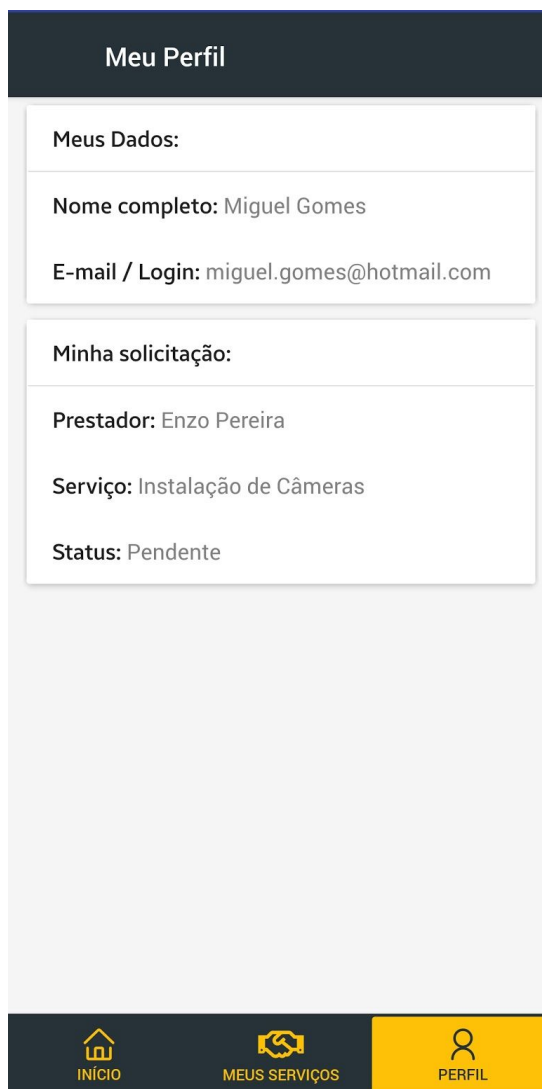
Fonte: Os autores.

Ao escolher o serviço desejado, o usuário será direcionado para a tela de detalhamento (Figura 9), que contempla todas as informações necessárias para a conclusão da solicitação como o tempo estimado, descrição detalhada de como o serviço será realizado e por fim, o usuário terá a opção de enviar uma mensagem ao prestador e realizar sua solicitação.


5.1.3 MEU PERFIL (SOLICITANTE)


Após solicitar o serviço desejado, o usuário é automaticamente direcionado para a tela de Meu Perfil, que informa alguns de seus dados pessoais juntamente com o status da solicitação, onde pode acompanhar e aguardar para o prestador aceite a solicitação e realize o serviço, como mostra na Figura 10.


Figura 10 - Tela de Meu Perfil



Meu Perfil	
Meus Dados:	
Nome completo:	Miguel Gomes
E-mail / Login:	miguel.gomes@hotmail.com
Minha solicitação:	
Prestador:	Enzo Pereira
Serviço:	Instalação de Câmeras
Status:	Pendente

 INÍCIO

 MEUS SERVIÇOS

 PERFIL

Fonte: Os autores.

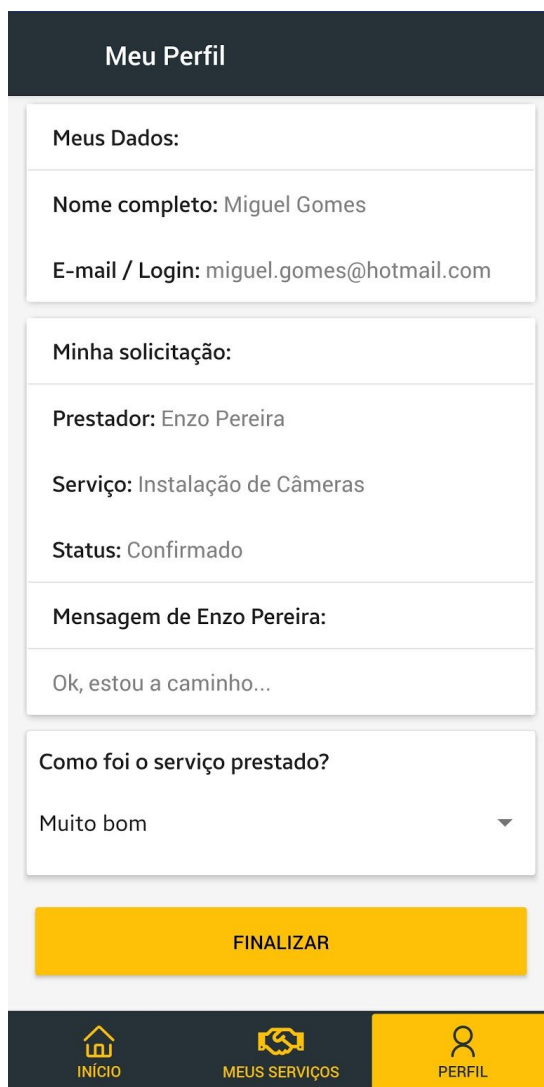
5.1.4 FINALIZAÇÃO DO SERVIÇO (SOLICITANTE)

Após a solicitação do serviço ser confirmada, seu status é alterado automaticamente e é exibida a mensagem que o prestador enviou ao aceitar a

solicitação, neste momento o prestador se dirige ao encontro do solicitante para realizar o serviço como ilustra a Figura 11.

Para que a solicitação seja finalizada, é obrigatório realizar uma avaliação do serviço, fazendo com que os próximos solicitantes tenham a referência da qualidade de todos os serviços disponibilizados pelo prestador (Figura 11).

Figura 11 - Finalização do serviço prestado



Meu Perfil

Meus Dados:

Nome completo: Miguel Gomes

E-mail / Login: miguel.gomes@hotmail.com

Minha solicitação:

Prestador: Enzo Pereira

Serviço: Instalação de Câmeras

Status: Confirmado

Mensagem de Enzo Pereira:

Ok, estou a caminho...

Como foi o serviço prestado?

Muito bom ▼

FINALIZAR

INÍCIO **MEUS SERVIÇOS** **PERFIL**

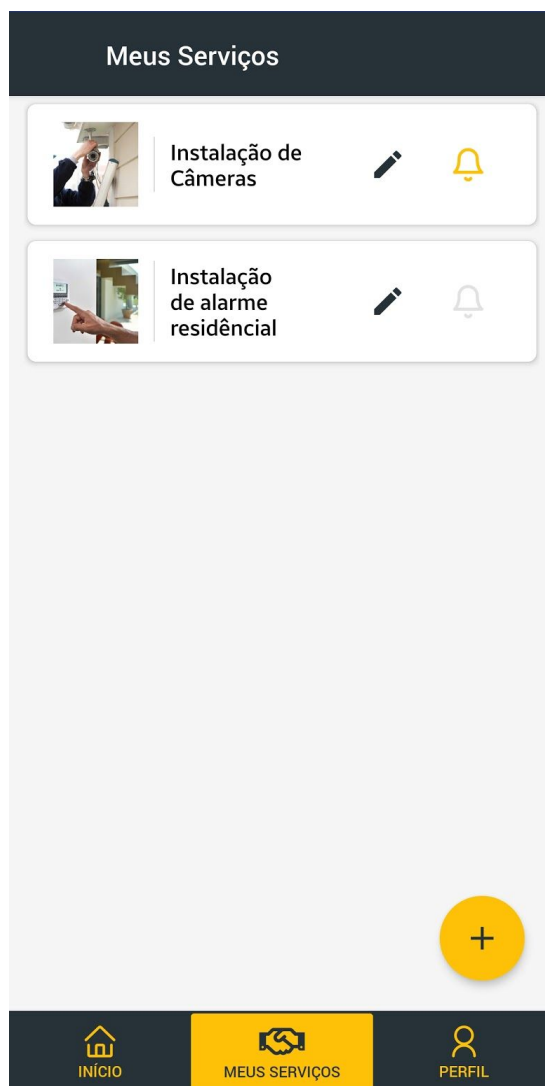
Fonte: Os autores.

5.1.5 MEUS SERVIÇOS (PRESTADOR)

Para o usuário que deseja ser um prestador, é necessário que o mesmo, tenha cadastrado anteriormente o serviço que pretende oferecer e aguardar até que alguém o solicite.

A tela de Meus Serviços (Figura 12) exibe os serviços que foram cadastrados pelo prestador, cada card possui um ícone de sino que alerta ao prestador uma solicitação existente no momento e para visualizá-la basta pressioná-lo.

Figura 12 - Tela de serviços do prestador

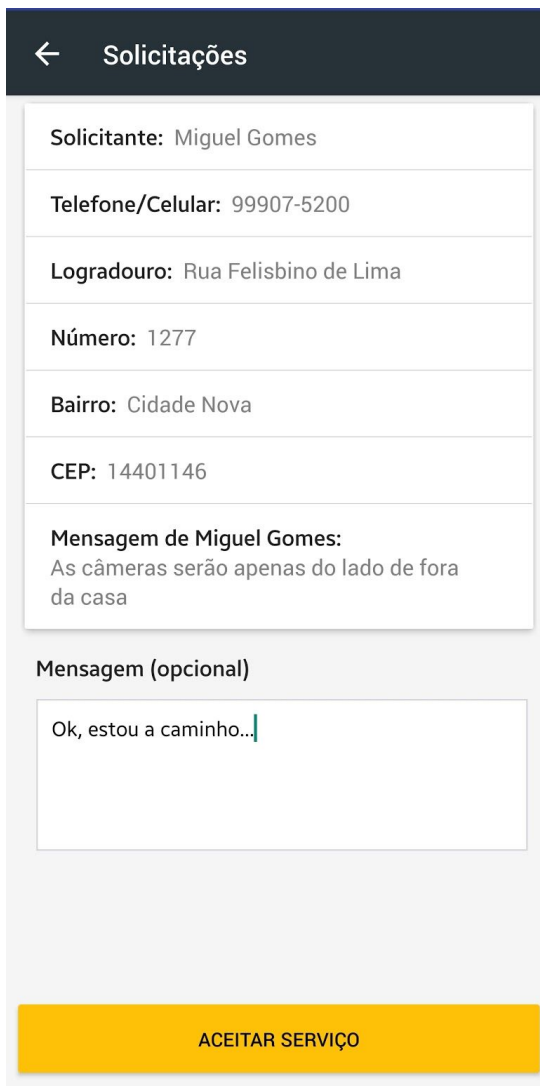


Fonte: Os autores.

Na tela de solicitações do serviço (Figura 13), o prestador consegue visualizar as informações básicas do solicitante, como nome, telefone/celular para algum contato imediato, sua localização através do endereço completo e a mensagem que o solicitante tem a opção de enviar.

Para realizar o serviço é necessário aceitar a solicitação no aplicativo pressionando o botão na parte inferior da tela, se preferir o prestador pode enviar uma mensagem para o solicitante e se locomover até o endereço do mesmo.

Figura 13 - Tela de solicitações



← Solicitações

Solicitante: Miguel Gomes

Telefone/Celular: 99907-5200

Logradouro: Rua Felisbino de Lima

Número: 1277

Bairro: Cidade Nova

CEP: 14401146

Mensagem de Miguel Gomes:
As câmeras serão apenas do lado de fora da casa

Mensagem (opcional)

Ok, estou a caminho...

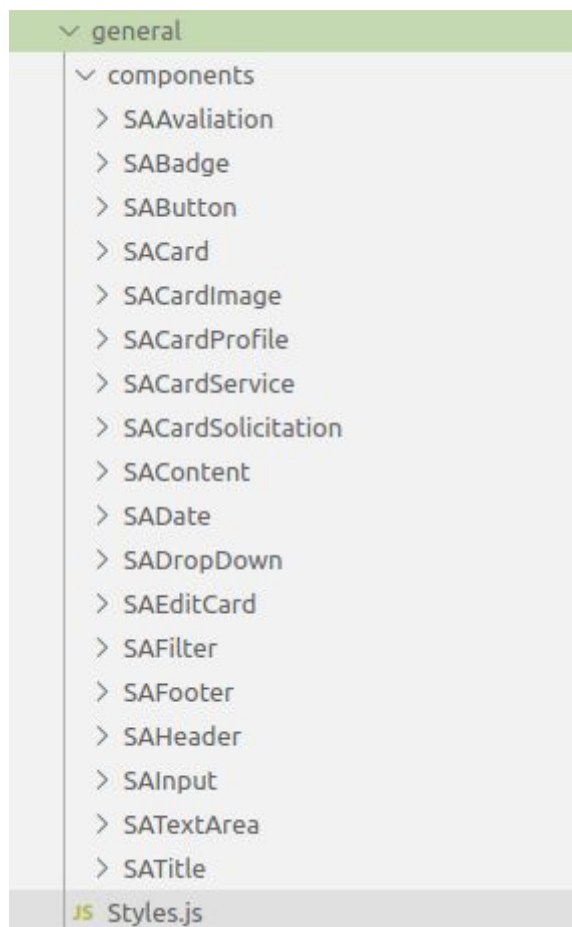
ACEITAR SERVIÇO

Fonte: Os autores.

5.2.1 IMPLEMENTAÇÃO DO FRONT-END

O aplicativo foi desenvolvido utilizando *React Native*, um *framework* que possui como base a linguagem de programação *Javascript*. Nele é trabalhado toda parte de componentização do *software*, fazendo com que o seu desenvolvimento seja realizado de maneira simples e reutilizável.

Figura 14 - Componentes da aplicação



Fonte: Os autores.

A Figura 14 ilustra o exemplo onde cada pasta é um componente utilizado pelas telas do aplicativo como botões, campos de texto, cartões e etc. Uma das vantagens de fragmentar a aplicação por componentes, é o fato de que a manutenção do *software* será extremamente segura, simples e rápida, pois é necessário alterar apenas o componente desejado, para que o mesmo seja atualizado em todas as telas que o consome.

Figura 15 - Trecho implementado para a consulta dos serviços

```
8  export function getServices(idUser, idCategory) {
9      return dispatch => {
10         dispatch({ type: SERVICES_REQUESTING })
11         return api
12             .get(`/services/${idUser}/${idCategory}`)
13             .then(res => {
14                 dispatch({
15                     type: SERVICES_SUCCESS,
16                     payload: res.data
17                 })
18             })
19             .catch(err => {
20                 dispatch({
21                     type: SERVICES_FAILURE,
22                     payload: err
23                 })
24             })
25     }
26 }
```

Fontes: Os autores.

A figura 15 mostra um trecho da arquitetura *Flux* implementada no *software* através da biblioteca *Redux*. Na linha 8 é possível observar a função que contempla a consulta dos serviços por meio de uma requisição para a API realizada na linha 11 onde é enviado a identificação do usuário e a categoria do serviço desejado.

Nas linhas 10, 14 e 20 da figura 11, nota-se uma função chamada *dispatch*, que envia para o *reducer* cada ação de acordo com o andamento da requisição, essas são separadas por 3 tipos:

Carregamento, quando a requisição ainda está sendo processada e não houve nenhum retorno (linha 10);

Sucesso, quando a requisição já foi processada e retorna os dados que o usuário solicitou (linha 15);

Falha, quando a requisição já foi processada e retorna um erro por não conseguir obter os dados que o usuário solicitou (linha 21);

Figura 16 - Trecho implementado para as tratativas recebidas

```
7  export const INITIAL_STATE = {
8    |   isRequesting: false,
9    |   success: false,
10   |   error: false,
11   |   content: []
12   | }
13
14  export default function services(state = INITIAL_STATE, action) {
15    |   switch (action.type) {
16    |     |   case SERVICES_REQUESTING:
17    |     |     |   return {
18    |     |     |     |   ...state,
19    |     |     |     |   isRequesting: true
20    |     |     |   }
21    |     |   case SERVICES_SUCCESS:
22    |     |     |   return {
23    |     |     |     |   ...state,
24    |     |     |     |   isRequesting: false,
25    |     |     |     |   content: action.payload || state.content,
26    |     |     |     |   error: false
27    |     |     |   }
28    |     |   case SERVICES_FAILURE:
29    |     |     |   return {
30    |     |     |     |   ...state,
31    |     |     |     |   isRequesting: false,
32    |     |     |     |   error: true
33    |     |     |   }
34    |     |   default:
35    |     |     |   return state
36    |   }
37  }
```

Fonte: Os autores.

As ações recebidas pelo *reducer* são separadas por tipo e cada uma delas resultará em um objeto diferente a ser construído e retornado para o usuário como mostra as linhas 16, 21 e 28 (Figura 16).

O código-fonte completo da implementação *Front-end* está disponível em (LIMA e SANTOS, 2020).

5.2.2 IMPLEMENTAÇÃO DO BACK-END

Para o tráfego de informações foi criado uma API em *NodeJS*, uma plataforma baseada também na linguagem de programação *Javascript*, que possibilita um controle maior dos dados armazenados, com ela a interface consegue fazer requisições assíncronas, invocando funções diretas no banco de dados e para isso foi utilizado a SGBD denominado PostgreSQL.

Figura 17 - Trecho da API que resulta na consulta de serviços.

```
10  async function getServices(req, res) {
11    try {
12      let rtn = await repository.getServices(req.params);
13
14      if (rtn.executionCode !== 0)
15        return res.status(200).json(rtn);
16
17      res.json(rtn);
18    } catch (err) {
19      return res.status(404).json({ err })
20    }
21  }
```

Fonte: Os autores.

A Figura 17 detalha a função que engloba o retorno de todos os serviços disponíveis para a solicitação. Neste trecho, os dados enviados pela interface são recebidos e tratados por uma instrução denominada *try* (linha 11), que busca executar um bloco de código desejado e especifica caso seja disparado algum erro ou exceção, como é possível observar na linha 18.

Ainda no trecho da instrução *try*, os dados são direcionados para outro arquivo, que tem como objetivo executar a função de consulta dos serviços no banco de dados (Figura 17).

Figura 18 - Trecho da API que requisita funções do banco de dados

```
17  async function getServices(obj) {
18    return pg.request(global.sql)
19      .input('pIdUser', obj.id)
20      .input('pIdCategory', obj.idCategory)
21      .asyncExec(procs.getServices);
22  }
```

Fonte: Os autores.

Na Figura 18, entre as linhas 19 e 20 é possível observar a identificação do usuário juntamente com a categoria do serviço desejado sendo enviadas para o banco de dados.

Finalmente, para a consulta dos serviços é utilizada a linguagem de consulta estruturada SQL, que permite a execução de diversas tarefas em um banco de dados relacional como o PostgreSQL.

Figura 19 - Trecho de uma função em SQL que retorna todos os serviços

```
101 BEGIN
102 if pidcategory > 0 THEN
103 RETURN QUERY SELECT
104     s.id_servico,
105     s.nome,
106     s.descricao,
107     s.imagem,
108     u.nome,
109     c.nome_categoria,
110     t.desc_tempo,
111     v.desc_valor
112
113 FROM
114     public.tb_servico s
115     INNER JOIN tb_usuarios u ON u.id_usr = s.id_usr
116     INNER JOIN tb_categoria c ON c.id_categoria = s.id_categoria
117     INNER JOIN tb_tempo t ON t.id_tempo = s.id_tempo
118     INNER JOIN tb_valor v ON v.id_valor = s.id_valor
119 WHERE u.id_usr <> pIdUser AND s.id_categoria = pidcategory;
```

Fonte: Os autores.

O trecho da Figura 19, mostra a consulta em SQL que recupera as informações desejadas armazenadas no banco de dados, através de parâmetros enviados anteriormente como a identificação do usuário e a categoria do serviço.

O código-fonte completo da implementação *Back-end* está disponível em (LIMA e SANTOS, 2020).

6. CONCLUSÃO

Após a finalização do desenvolvimento do *MVP (Minimum Viable Product)* da aplicação móvel intitulada *Service Advisor*, que apoia e facilita o encontro de solicitantes e prestadores de serviço gerando visibilidade para

prestadores e facilidades para solicitantes, o objetivo principal do projeto foi alcançado.

Com o propósito de apresentar todo conhecimento adquirido ao longo do curso de Sistemas de Informação, o projeto foi elaborado utilizando técnicas e artefatos da Engenharia de *Software* para auxiliar no desenvolvimento da aplicação que foi realizado através do *framework React Native* e a plataforma *NodeJS*.

Visando a evolução do projeto, foi possível notar a possibilidade de melhorias que podem beneficiar ainda mais a experiência do usuário. Para implementações futuras, haverá um *chat* aperfeiçoando a comunicação entre as partes juntamente com o compartilhamento de localização por GPS e também um módulo de segurança, voltado para o público feminino, onde mulheres poderão optar por se comunicar somente com mesmo gênero.

Empregando a aplicação em uma situação real, é possível validar se as funcionalidades estão de acordo com o esperado, e também realizar um levantamento de requisitos para melhorias não contempladas até o momento. Desta forma, pretende-se implementar a aplicação em uma escala reduzida, um condomínio, com alguns profissionais por exemplo, acompanhando e analisando os resultados de como o aplicativo impactará no dia-a-dia de seus usuários.

7. REFERÊNCIAS

BECKER, Lauro. **O que é React Native?**. Orgânica Natural Marketing, 2020. Disponível em: <<https://www.organicadigital.com/blog/o-que-e-react-native/>>. Acesso em: 12 abril de 2020.

BICUDO, Lucas. **O que é uma startup?**. StartSe, 2016. Disponível em: <<https://www.startse.com/noticia/startups/afinal-o-que-e-uma-startup>>. Acesso em: 19 de junho de 2020.

BUENO, Jefferson Reis. **Mas afinal, o que é empreendedorismo?**. Blog SEBRAE, 2019. Disponível em: <<https://blog.sebrae-sc.com.br/o-que-e-empreendedorismo/>>. Acesso em: 18 de junho de 2020.

CAMARGO, Robson. **O que é Canvas? E como pode auxiliar em seus projetos?**. Robson Camargo Projetos e Negócios, 2019. Disponível em: <<https://robsoncamargo.com.br/blog/O-que-e-Canvas/>>. Acesso em: 20 de junho de 2020.

DB-ENGINES. **DB-Engines Ranking**. DB-Engines, 2020. Disponível em: <<https://db-engines.com/en/ranking>>. Acesso em: 15 de agosto de 2020.

DEVMEDIA. **Exemplo de aplicação React com Redux**. DevMedia, 2020. Disponível em: <<https://www.devmedia.com.br/exemplo/exemplo-de-aplicacao-react-com-redux/66>>. Acesso em: 12 de abril de 2020.

ESCUDELARIO, Bruna. **Vale a pena aprender NodeJS?**. iMasters, 2018. Disponível em: <<https://imasters.com.br/back-end/vale-pena-aprender-nodejs>>. Acesso em: 13 de abril de 2020.

FERNANDES, André. **O que é API? Entenda de uma maneira simples**. Vertigo Tecnologia, 2018. Disponível em: <<https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>>. Acesso em: 13 de abril de 2020.

GASPAROTTO, Henrique. **Aplicações Móveis: Nativas ou Web?**. DevMedia, 2014. Disponível em: <<https://www.devmedia.com.br/aplicacoes-moveis-nativas-ou-web/30392>>. Acesso em: 12 de abril de 2020.

IANAKIARA, Dio. **10 motivos para usar Redux em 2018**. Getty/IO Blog, 2018. Disponível em: <<https://blog.getty.io/10-motivos-para-usar-redux-em-2018-96664e124425>>. Acesso em 13 de abril de 2020.

KROGER, Hélio. **Entendendo React e Redux de uma vez por todas**. Medium, 2017. Disponível em: <https://medium.com/@hliojnior_34681/entenda-react-e-redux-de-uma-vez-por-todas-c761bc3194ca>. Acesso em: 12 de abril de 2020.

LIMA, Miguel; SANTOS, Cairo. **Repositório para a disponibilização de documentos do Trabalho de Conclusão de Curso - Cairo e Miguel**. Disponível em: <<https://github.com/MiguelAleixo/service-advisor>>. Acesso em: 18 de agosto de 2020.

MATTHEW, Neil; STONES, Richard. **Beginning Databases with PostgreSQL From Novice to Professional**. 2. ed. Apress, 2005.

MICHELAZZO, Paulino. **Dicas para documentação de softwares**. Michelazzo, 2006. Disponível em: <<http://www.michelazzo.com.br/dicas-para-documentacao-de-softwares/>>. Acesso em: 29 de junho de 2020.

MIRANDA, William. **Modelagem de Dados: O que é e para que serve para um DBA**. Portal GSTI, 2017. Disponível em: <<https://www.portalgsti.com.br/2017/02/modelagem-de-dados-o-que-e-e-para-que-se-rve-para-um-dba.html>> Acesso em: 15 de agosto de 2020.

NOGUEIRA, Alana. **O que ninguém te conta sobre empreender em TI**. Blog do Movidesk, 2019. Disponível em: <<https://conteudo.movidesk.com/empreender-em-ti/>>. Acesso em 20 de junho de 2020.

OLIVEIRA, Wallace. **Notação BPMN, a mais usada para modelar processos**. HEFLO, 2018. Disponível em: <<https://www.heflo.com/pt-br/bpm/notacao-bpmn/>> . Acesso em: 17 de junho de 2020.

POSTGRESQL. **What is PostgreSQL?**. PostgreSQL, 2020. Disponível em: <<https://www.postgresql.org/about/>>. Acesso em: 12 de abril de 2020.

SANTOS, Guilherme. **Node.js – O que é, por que usar e primeiros passos**. Medium, 2016. Disponível em: <<https://medium.com/thdesenvolvedores/node-js-o-que-%C3%A9-por-que-usar-e-primeiros-passos-1118f771b889>>. Acesso em: 12 de abril de 2020.

SCUDERO, Erick. **TOP 10 principais SGBDs do mercado global!**. Bencode, 2016. Disponível em: <<https://bencode.com.br/principais-sgbds/>>. Acesso em: 15 de agosto de 2020.

SOARES, Jhonathan. **Lista de bancos de dados mais utilizados no mercado - 2019/2020**. Código Simples, 2020. Disponível em: <<https://codigosimples.net/2020/01/13/lista-de-banco-de-dados-mais-utilizados-no-mercado-2019-2020/>>. Acesso em: 15 de agosto de 2020.

TESSIS, Thiago. **Node.js: quem já utiliza a tecnologia (e aprova)?**. Umblar, 2017. Disponível em: <<https://blog.umbler.com/br/node-js-quem-ja-utiliza-a-tecnologia-e-aprova/>>. Acesso em: 12 de abril de 2020.

VIEIRA, Rodrigo. **UML — Diagrama de Casos de Uso**. Medium, 2015. Disponível em: <<https://medium.com/operacionalti/uml-diagrama-de-casos-de-uso-29f4358ce4d5>>. Acesso em: 18 de junho de 2020.

YEEPLY. **Usabilidade em aplicativos móveis: O que é e porque é necessária?**. YeePLY, 2016. Disponível em: <<https://pt.yeePLY.com/blog/usabilidade-em-aplicativos-moveis/>>. Acesso em: 12 de abril de 2020.