Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Luan Eduardo da Costa

Rafael Donizete Basso

DESENVOLVIMENTO DE UM APLICATIVO PARA PÓS-MARKETING: NEXT MSGER

Americana – SP

2020

Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Luan Eduardo da Costa

Rafael Donizete Basso

DESENVOLVIMENTO DE UM APLICATIVO PARA PÓS-MARKETING: NEXT MSGER

Trabalho de Conclusão de Curso desenvolvido em cumprimento a exigência do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Dr. Kleber de Oliveira Andrade

Área de concentração: Engenharia de Software

Americana – SP

2020

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

C873d COSTA, Luan Eduardo da

Desenvolvimento de um aplicativo para pós-marketing: Next MSGer. / Luan Eduardo da Costa, Rafael Donizete Basso. – Americana, 2020.

222f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Dr. Kleber de Oliveira Andrade

1 Desenvolvimento de software 2 Dispositivos móveis - aplicativos I. BASSO, Rafael Donizete II. ANDRADE, Kleber de Oliveira III. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.3.05
681.518

Luan Eduardo da Costa

Rafael Donizete Basso

DESENVOLVIMENTO DE UM APLICATIVO PARA PÓS-MARKETING: NEXT MSGER

Trabalho de Conclusão de Curso desenvolvido em cumprimento a exigência do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Dr. Kleber de Oliveira Andrade.

Área de concentração: Engenharia de Software

Americana, 03 de dezembro de 2020

Banca Examinadora:

_____

Kleber de Oliveira Andrade (Presidente)
Doutor
Fatec Americana

_____

Wagner Siqueira Cavalcante (Membro)
Mestre
Fatec Americana

_____

André de Lima (Membro)
Doutor
Fatec Americana

AGRADECIMENTOS

Nossos agradecimentos a todas as pessoas que contribuíram direta ou indiretamente para a realização deste trabalho, em especial:

Ao Prof. Kleber de Oliveira Andrade que nos auxiliou desde a idealização do projeto e durante todas as etapas de desenvolvimento.

A todos nossos familiares, que sempre nos deram o apoio necessário para continuarmos estudando e dando o máximo de nós.

RESUMO

O presente trabalho foca na análise e desenvolvimento de um aplicativo para *smartphones* que visa facilitar o processo de pós-marketing de empresas prestadoras de serviço, proporcionando cadastro de clientes, etapas de *marketing*, serviços e principalmente o envio de mensagens para os clientes cadastrados, tornando a comunicação entre empresa e cliente mais frequente, clara e melhorando a experiência da contratação de um serviço. Para o desenvolvimento do *software* foram utilizados diversos procedimentos de engenharia de *software*, como a análise de requisitos funcionais e não funcionais, diagramas UML e a metodologia SCRUM foi escolhida e aplicada no gerenciamento das atividades. A linguagem JavaScript em conjunto com o *framework* React, React Native e o Firebase foram as principais ferramentas escolhidas para a criação do sistema. Por fim o trabalho obteve êxito em atingir os objetivos propostos e, como resultado, o aplicativo foi disponibilizado para usuários do sistema operacional Android na loja de aplicativos Play Store.

Palavras Chave: pós-marketing; serviço; cliente.

ABSTRACT

This work focuses on the analysis and development of an application for smartphones that aims to facilitate the post-marketing process of service providers, providing customer registration, marketing stages, services and mainly sending messages to registered customers, making more frequent, clearer communication between company and customer and improving the experience of contracting a service. For software development, several software engineering procedures were used, such as the analysis of functional and non-functional requirements, UML diagrams and the SCRUM methodology was chosen and applied in the management of activities. The JavaScript language in conjunction with the React framework, React Native and Firebase were the main tools chosen for the creation of the system. Finally, the work was successful in achieving the proposed objectives and, as a result, the application was made available to users of the Android operating system in the Play Store application store.

Keywords: post-marketing; service; customer.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Interface*
APK	*Application Package*
BAAS	*Back-end As A Service*
CEP	Código de Endereçamento Postal
CNPJ	Cadastro Nacional de Pessoa Jurídica
CPF	Cadastro de Pessoa Física
CSS	*Cascading Style Sheets*
DB	*Database*
DER	Diagrama de Entidade e Relacionamento
FAB	*Floating Action Button*
HTTP	*HyperText Transfer Protocol*
ID	*Identifier*
IDE	*Integrated Development Environment*
JS	JavaScript
JSON	*JavaScript Object Notation*
NPM	*Node Package Manager*
OBS	Observação
OS	*Operational System*
PO	*Product Owner*
QR	*Quick Response*
RE	*Requirements Engineering*
RF	Requisito Funcional
RNF	Requisito Não Funcional
SDK	*Software Development Kit*
SMS	*Short Message Service*
SQL	*Structured Query Language*
SRC	*Source*
UI	*User Interface*
UML	*Unified Modeling Language*
UTC	*United Time Coordinated*

SUMÁRIO

1 INTRODUÇÃO

Devido a constante necessidade das empresas em manter uma carteira de clientes sólida e conseguir administrá-la no tocante ao gerenciamento de relacionamento, percebe-se que há uma grande dificuldade de se realizar processos de *marketing* pós-venda com seus clientes.

O projeto em questão trata exatamente de como melhorar essa dificuldade das empresas, de forma geral, em manter um bom relacionamento com clientes após um serviço ser finalizado. Para isto, um aplicativo foi desenvolvido, disponibilizando uma interface simples e amigável para o cadastro de clientes, serviços, etapas de *marketing* e o contato com o cliente através de mensagens enviadas por diversos canais, como *e-mail*, SMS, WhatsApp e até ligações telefônicas.

O aplicativo não está limitado a um único segmento de mercado, podendo ser levado para uso pessoal ou empresarial e a maioria de suas funcionalidades pode ser executada sem necessidade de uma conexão com a internet, o que proporciona uma mobilidade aos usuários.

Este trabalho tem como objetivo melhorar o processo de pós-marketing de prestadores de serviços. Quanto aos objetivos específicos são:

- Utilizar a metodologia SCRUM e expor os procedimentos realizados;
- Demonstrar como solução desenvolvida funciona;
- Mostrar como a ideia e os requisitos foram criados;
- Explicar as tecnologias e ferramentas utilizadas para a criação do *software*;
- Apresentar as conclusões finais e exibir as possibilidades para trabalhos futuros.

O restante do trabalho está organizado em três capítulos conforme descrição a seguir: Capítulo 2 apresenta o projeto do sistema, o Capítulo 3 mostra a modelagem de dados e contém os diagramas UML, o capítulo 4 descreve o desenvolvimento do projeto utilizando a metodologia SCRUM, o capítulo 5 mostra as telas criadas e as considerações finais, juntamente com as diversas possibilidades de trabalhos futuros são apresentadas no Capítulo 4.

2 PROJETO DO SISTEMA

O aplicativo foi desenvolvido utilizando o *framework open source* React Native criado pelo Facebook, que permite a criação de aplicativos nativos para Android e iOS utilizando JavaScript como linguagem. Como banco de dados foi utilizado o Firebase pela facilidade de integração no aplicativo e o seu arsenal de funcionalidades que acelerou o desenvolvimento da solução.

Antes mesmo de se iniciar o desenvolvimento de códigos foram realizadas pesquisas mercadológicas para determinar a viabilidade do aplicativo e priorizar tarefas para que o aplicativo tenha as funcionalidades essenciais mais rapidamente. Como resultado foram encontrados outros *softwares* minimamente semelhantes, mas que não possuíam várias das funcionalidades que desejávamos desenvolver, como envio de mensagens por vários canais de comunicação, cadastro de etapas de *marketing*, cadastro de tipos de serviço, retornos ao estabelecimento, entre outras.

2.1 Sistemas Similares

A fim de facilitar o contato com clientes após realizar uma venda de um serviço e aumentar as chances de um cliente realizar novamente uma compra, existem variados *softwares* de gestão pós-*marketing* para tais serviços. Atualmente, os mais populares que existem no mercado são:

- Eloqua (S1): *Software* gerenciado pela Oracle e bem reconhecido no mercado, tem grandes vantagens por ter facilidade de uso, no entanto é de alto preço e complexo na sua configuração inicial. Tem foco não só na realização de pós-venda, mas também em todo o sistema de gerenciamento de *marketing* digital, criação de campanhas de *marketing* entre outras coisas.
- Nova Central Pós-Vendas (S2): *Software* de fácil utilização que foca na fidelização de clientes e no *marketing* após a venda de um produto. Nele também é possível acompanhar a situação seus pedidos após a realização de um pedido.

- Meu Pós-Venda (S3): *Software* estritamente voltado para reter, engajar e resgatar clientes inativos. Tem uma boa colocação no mercado específico para montadoras de veículos.
- Art Informática (S4): *Software* com suporte disponível para todo o país. O seu módulo pós-vendas tem como funcionalidades emitir ordens de serviço e gerenciar a manutenção, suporte e acompanhamento dos contratos iniciais realizados.
- Notificações Inteligentes (S5): é uma ferramenta de automatização de eventos pelo WhatsApp. Todas as funcionalidades são geradas automaticamente e enviadas através de mensagens, imagens ou áudio via plataforma de rede social WhatsApp.
- Histórico de Atendimento ao Cliente (S6): Aplicativo voltado à consulta de registros de serviços de clientes, podendo realizar um atendimento personalizado através das informações adquiridas em relatórios.

Levando estes aspectos em consideração, foi elaborada a Tabela 1 mostrando as principais diferenças entre tais *softwares* e o aplicativo desenvolvido (S7) neste trabalho:

Tabela 1 - Comparativo de funcionalidades entre os *softwares* similares e o aplicativo desenvolvido neste trabalho.

Funcionalidade	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
Variabilidade de segmento de serviço	X	X		X	X	X	X
Cadastro de Serviço Realizado (após o final do serviço)						X	X
Cadastro de tipo de serviço				X			X
Cadastro de etapas pós-marketing	X						X
Cadastro do retorno do cliente ao estabelecimento			X				X
Cadastro de clientes	X		X	X			X
Cadastro de avaliação dos serviços	X						X
Histórico de Serviços Realizados	X	X	X	X	X	X	X
Plataformas WEB	X	X	X	X			
Plataformas *Mobile*	X	X	X	X	X	X	X
Plataforma *Mobile* nativa							X
Acompanhamento por etapas		X	X	X	X		X

Automatização de eventos através de rede sociais					X		
Custo inicial baixo		X		X		X	X
Envio de mensagens para clientes	X	X	X	X	X		X
Envio de mensagens por vários canais de comunicação							X
Envio de mensagens de aniversário do cliente							X

Fonte: Elaborado pelos autores (2020).

2.2 Levantamento de Requisitos

A engenharia de requisitos (RE – *Requirements Engineering*) é o processo de descobrir, analisar, documentar e verificar requisitos de um sistema. Um requisito pode ser definido como uma descrição dos serviços fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2007). Tradicionalmente, os requisitos são divididos em dois tipos: requisitos funcionais e requisitos não funcionais.

2.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema deve fazer, isto é, definem a funcionalidade desejada do software (SOMMERVILLE, 2007). Logo abaixo, a Tabela 2 apresenta os requisitos funcionais deste projeto.

Tabela 2 - Requisitos funcionais do projeto.

Identificação	Requisito Funcional	Prioridade
RF001	Realizar *Login.*	Essencial
RF002	Manter clientes (cadastrar, editar, visualizar e excluir).	Essencial
RF003	Manter tipos de serviço (cadastrar, visualizar e excluir).	Essencial
RF004	Manter etapas de pós-marketing (cadastrar, editar, visualizar e excluir).	Essencial
RF005	Manter serviços (cadastrar, editar, visualizar e excluir).	Essencial
RF006	Manter avaliação do serviço feita pelo cliente (cadastrar, editar, visualizar e excluir).	Desejável
RF007	Manter retornos do cliente ao estabelecimento da empresa por causa de um serviço (cadastrar, editar, visualizar e excluir).	Desejável
RF008	Manter motivos de retorno (cadastrar, visualizar e excluir).	Desejável
RF009	Realizar comunicação com clientes após a realização de serviços via WhatsApp, *e-mail*, SMS e ligação telefônica.	Essencial
RF010	Enviar mensagens de aniversário para clientes.	Importante
RF011	Manter usuário (alterar senha, nome e *e-mail*, visualizar perfil próprio e recuperar conta quando esquece a senha).	Essencial
RF012	Manter empresa (cadastrar, visualizar e editar)	Essencial
RF013	Manter configuração de mensagem de aniversário (cadastrar, visualizar, editar, excluir).	Importante
RF014	Deslogar da conta.	Essencial

Fonte: Elaborado pelos autores (2020).

2.2.2 Requisitos Não Funcionais

"Os requisitos não funcionais são aqueles não diretamente relacionados às funções específicas fornecidas pelo sistema" (SOMMERVILLE, 2007). A Tabela 3 apresenta os requisitos não funcionais deste projeto.

Tabela 3 - Requisitos não funcionais do projeto.

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	Mostra uma mensagem de fácil compreensão ao usuário digitar uma informação não válida.	Usabilidade	Desejável
RNF002	O sistema oferece confiança por ter uma interface de fácil compreensão devido às informações consistentes.	Confiabilidade	Essencial
RNF003	O sistema gera resposta rápida ao usuário, em média 3 segundos.	Desempenho	Importante
RNF004	O tráfego de dados entre o aplicativo e o servidor é criptografado para proteger os dados do usuário.	Segurança	Essencial
RNF005	Existe uma gama de segmentos no mercado de trabalho a quem o sistema pode atender.	Distribuição	Importante
RNF006	O sistema oferece facilidade de uso com botões e caixa de textos padronizados para uma interface de fácil entendimento.	Padrões	Desejável
RNF007	O sistema foi desenvolvido para 2 tipos de plataforma *mobile* (Android e IOS).	*Hardware* e *Software*	Importante
RNF008	É possível continuar a usar o aplicativo para inserir, editar, excluir e visualizar informações mesmo sem conexão com a internet.	Confiabilidade	Importante
RNF009	Um usuário de uma empresa não consegue modificar as informações cadastradas de outra empresa nem inserir novos registros.	Segurança	Importante

Fonte: Elaborado pelos autores (2020).

2.3 Recursos e Ferramentas

Esta seção contempla as ferramentas de programação e os recursos necessários para o desenvolvimento do sistema:

- **Firebase:** O Firebase é uma plataforma de desenvolvimento de aplicativos móveis, ela permite criar aplicativos rapidamente sem se preocupar com o gerenciamento da infraestrutura, sendo assim a plataforma oferece análises, banco de dados, mensagens e relatórios de erros. As APIs do Firebase estão incluídas em um único SDK, ou seja, é possível expandir os aplicativos para diversas plataformas e linguagens com facilidade. Hoje a Firebase é mantida pela Google (FIREBASE, 2020).

- **Git:** O Git se trata de um sistema de controle de versão distribuído em código aberto, ou seja, gratuito e que também pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo. Isso foi feito para funcionar no *kernel* do Linux, sendo assim consegue lidar com grandes repositórios. Ele é escrito em linguagem C e sua manutenção é atualmente supervisionada por Junio Hamano (GIT, 2020).
- **GitHub:** O GitHub é uma plataforma de desenvolvimento que possibilita a hospedagem de código fonte com controle de versão utilizando o Git (GITHUB, 2020).
- **JavaScript:** O JavaScript (JS) é uma linguagem de programação interpretada e baseada em objetos com funções de primeira classe, é uma linguagem altamente utilizada em páginas *web* e também em outros ambientes como Node.js, Apache CouchDB e Adobe Acrobat. O JavaScript foi criado por Bredan Eich. O padrão do JavaScript é o ECMAScript, facilitando o suporte para todos os navegadores (MDN WEB DOCS, 2020).
- **Trello:** É uma ferramenta de colaboração que organiza seus projetos em quadros, o Trello também informa o que está sendo trabalhado, quem está trabalhando em cada quadro e em que momento do processo o quadro está. É uma ferramenta gratuita com opções de assinaturas para mais recursos. Ele está disponível através da *web* e por aplicativos iOS e Android (TRELLO,2020).
- **Visual Studio Code:** Editor de código fonte leve, distribuído gratuitamente pela Microsoft que possibilita a programação em diversas linguagens diferentes de programação, assim como a utilização de extensões e ferramentas *open source* para agilizar o desenvolvimento. (VISUAL STUDIO CODE, 2020).
- **React:** O React ou muitas vezes denominado React.js é uma biblioteca JavaScript, ele foi criado e é mantido pelo Facebook. O React tem como principal intuito a construção de interfaces de usuário, ou seja, ele propõe que a criação de UIs interativas seja feita com maior facilidade. Ele é baseado em componentes encapsulados escritos em JavaScript que gerenciam o seu próprio estado (REACT, 2020).

- **React Native**: Constitui-se de um *framework* baseado em React que possibilita o desenvolvimento de aplicações *mobile* realmente nativos, ou seja, que não comprometem a experiência dos usuários tanto para os sistemas Android quanto para iOS. O React Native é escrito em JavaScript, ou seja, ele é uma biblioteca JavaScript. O React Native foi desenvolvido e é mantido pelo Facebook (REACT NATIVE, 2020).
- **Styled-components:** É uma Biblioteca para aplicações React e React Native que possibilita a utilização estilizada de componentes utilizando a sintaxe do CSS. Ela é uma das soluções de “CSS em JS” existentes, ou seja, permite a escrita de CSS em arquivos JS. Seus benefícios são: CSS com escopo (no caso da *web*), maior clareza do código, possibilidade de estender componentes e acessar as propriedades de um componente no arquivo de estilo através da interpolação de *strings* (*STYLED-COMPONENTS*, 2020).
- **Eslint:** Biblioteca que faz a checagem estática de códigos JavaScript, ou seja, mostra possíveis erros de programação e garante que toda a base de código seja escrita com um mesmo estilo (padronização de código), facilitando o entendimento do código pelos integrantes da equipe.
- **Prettier:** Formatador de código que possibilita uma rápida configuração dos padrões de código e integração com o Eslint.
- **I18next:** Biblioteca para facilitar a internacionalização de aplicações que permite a escrita de todos os textos utilizados no app em arquivos JSON e fornece ferramentas para trocar a linguagem de forma simples.
- **Moment:** Biblioteca JavaScript que abrange todas as manipulações de datas, facilitando a exibição de diferentes formatos de datas dependendo da linguagem do usuário ou do *layout*, assim como cálculos de datas. Sua integração com o i18next permite, além disso, a formatação a partir de interpolação de *strings* (MOMENT.JS, 2020).
- **Prop-types:** Biblioteca para aplicações React e React Native que auxilia no processo de tipagem das propriedades de componentes melhorando, desta forma, o *intellisense* da IDE, além de facilitar a leitura de um componente pelos demais membros da equipe.

- **react-native-gesture-handler:** Biblioteca para aplicativos que usam React Native que permite uma melhor manipulação de eventos *touchscreen*, garantindo animações com 60 frames por segundo.
- **Redux:** O Redux é uma biblioteca escrita com JavaScript que fornece suporte ao gerenciamento de estado e do fluxo de execução de tarefas das aplicações. Ele existe para várias linguagens e tem outras bibliotecas complementares para vários *frameworks*, ajudando desenvolvedores a escreverem códigos que se comportam consistentemente, são flexíveis, depuráveis e com funcionalidades poderosas, como o fazer e desfazer persistência de estado (REDUX, 2020).
- **react-navigation:** Biblioteca que engloba toda a lógica de rotas e navegação entre telas no React Native. Com a sua ajuda é possível construir aplicações robustas em pouco tempo, sem se preocupar em criar seu próprio sistema de navegação e sem tocar em código nativo para Android ou IOS (REACT-NAVIGATION, 2020).
- **Reactotron:** Aplicação desktop para computadores Windows e MacOS que inspeciona aplicações React e React Native. Com ele é possível monitorar requisições HTTP, o estado global da aplicação (*store*) gerenciada pelo Redux, além de exibir melhor *logs* e erros do sistema (INFINITE RED, 2020).
- **Yarn:** O Yarn é um gerenciador de pacotes que permite usar e compartilhar código com outros desenvolvedores. Ele é semelhante em funcionalidade com o NPM (Node Package Manager), mas além de ser *open source* ele fornece funcionalidades a mais, como a possibilidade de criar e manipular *workspaces*. Outra característica interessante é que o Yarn, ao contrário do NPM, ele faz cache dos pacotes baixados no computador, então as adições subsequentes da mesma versão do pacote presente na cache serão finalizadas mais rapidamente (YARN, 2020).
- **NodeJs:** Node.js é um interpretador de JavaScript assíncrono com código aberto orientado a eventos, criado por Ryan Dahl em 2009, focado em migrar a programação do JavaScript do cliente (*front-end*) para os servidores, criando aplicações de alta escalabilidade (como um servidor

web), manipulando milhares de conexões/eventos simultâneas em tempo real numa única máquina física (NODE.JS, 2020).

- **Figma:** Aplicação de *design* de interfaces. Ele fornece as ferramentas necessárias para criar *layouts*, vetores, imagens e até gerar código CSS a partir do *layout* desenhado. O *software* tem versões para Windows, MacOS, Linux e funciona também em navegadores.

3 MODELAGEM

Na fase da modelagem é feita a documentação do aplicativo, se trata de diagramas que facilitam na compreensão do projeto de forma padronizada.

A documentação deste trabalho utilizará a linguagem de modelagem *Unified Modeling Language* (UML) para modelar os diagramas de casos de uso, de sequência, estado e atividades.

3.1 Casos De Uso

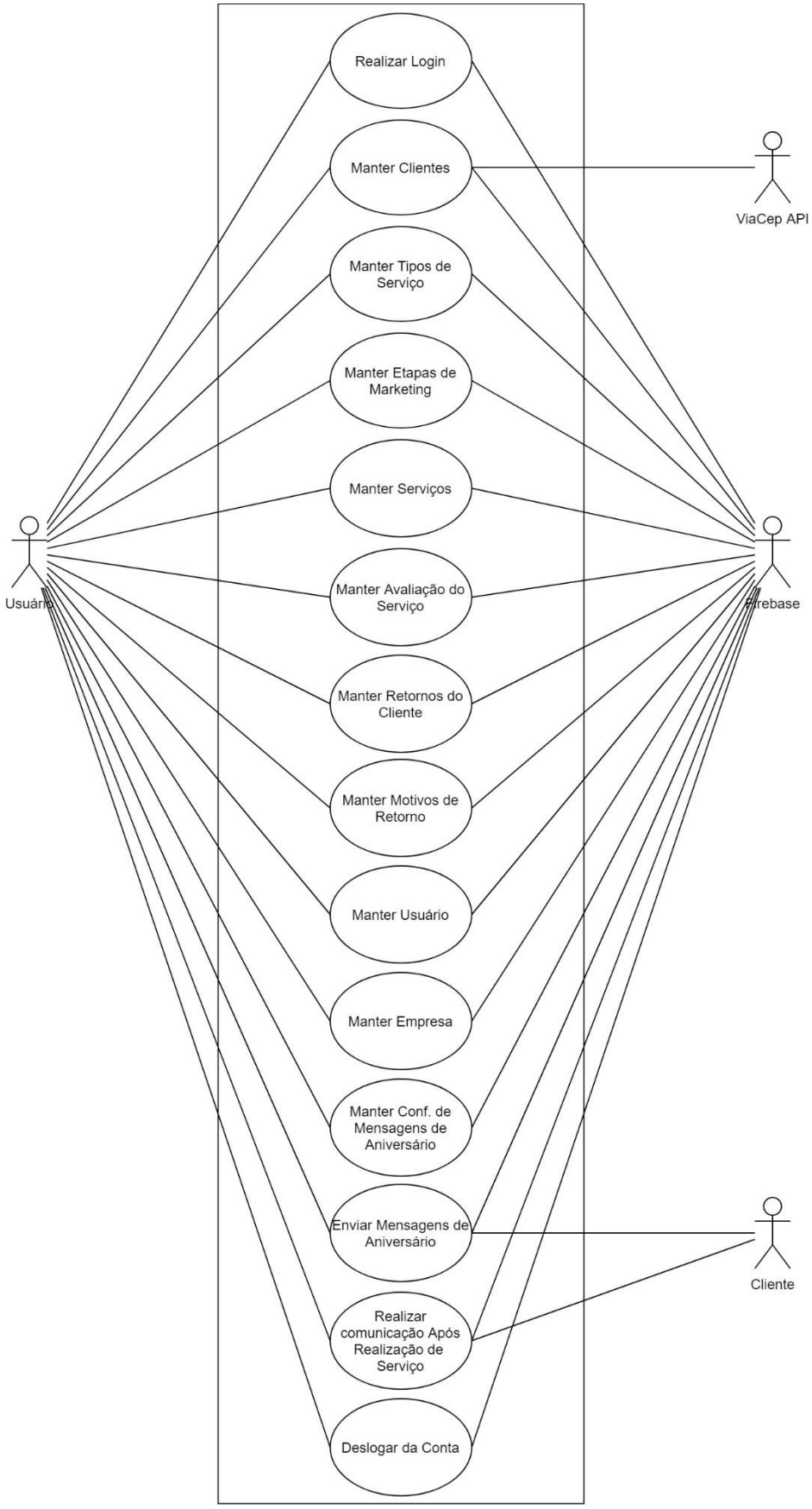
Um diagrama de caso de uso descreve um cenário de funcionalidades do ponto de vista do usuário, catalogando os requisitos funcionais do sistema. Dentro dele são retratados os atores (representado pelos bonecos), as funcionalidades (representadas pelos balões com a ação escrita por dentro) e as relações (representadas pelas linhas).

Os atores que interagem com o sistema são: o Usuário, o Firebase, o Cliente e a API do ViaCep.

- **Usuário:** é o ator que representa os utilizadores deste aplicativo. Um ator pode, por exemplo, cadastrar um serviço, cadastrar ou cliente ou cadastrar uma etapa, entre outras coisas.
- **Firebase:** representa o banco de dados em tempo real em que o sistema armazena todas as informações.
- **Cliente:** representa os clientes da empresa que usa o aplicativo. São eles que contratam os serviços e recebem as mensagens de pós-marketing.
- **ViaCep API:** API que permite a busca de um endereço pelo CEP.

A Figura 1 apresenta os casos de uso de todo o sistema.

Figura 1 - Diagrama de casos de uso.



Fonte: Elaborado pelos autores (2020).

3.2 Documentação dos Casos de Uso

Cada funcionalidade dos diagramas de casos de uso será descrita da Tabela 4 à Tabela 17 inclusive.

Tabela 4 - Caso de uso "Realizar *Login*".

Nome do caso de uso	Realizar *login.*
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos necessários para um usuário fazer *login* no sistema.
Ações do ator	**Ações do Sistema**
1. O usuário digita o *e-mail* da conta.	
2. O usuário digita a senha da conta.	
3. O usuário toca no botão "entrar".	
	4. O sistema verifica se foram preenchidos o *e-mail* e a senha.
	5. O sistema envia a requisição de autenticação para o Firebase.
	6. O Firebase devolve o resultado da requisição e o sistema age de acordo com ele, mostrando o erro ou redirecionando o usuário para a tela inicial se o *login* tenha sido efetuado com sucesso.

Validações	Para que o *login* seja efetuado, o usuário deve digitar o seu *e-mail* e senha corretamente.

Fonte: Elaborado pelos autores (2020).

Tabela 5 - Caso de uso “Manter Clientes”.

Nome do caso de uso	Manter Clientes.
Atores envolvidos	Usuário, Firebase, ViaCep API.
Objetivo	Este caso de uso descreve o que o usuário pode realizar no sistema para manter clientes e os passos necessários.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Novo Cliente” no menu lateral de navegação do aplicativo.	
	2. O sistema abre a tela de cadastro de clientes.
3. O usuário preenche o CEP.	
	4. O sistema faz uma requisição para a API do ViaCep passando o CEP digitado como parâmetro para obter o endereço do novo cliente e depois preenche os outros campos de endereço com os dados encontrados.
5. O usuário preenche as informações do novo cliente nos campos apropriados e toca no botão “Cadastrar Cliente”.	
	6. O sistema faz a requisição para o Firebase para inserir o novo cliente no banco de dados

	e em seguida limpa os campos de digitação e seleção para possibilitar um novo cadastro.
7. O usuário toca no item “Clientes” no menu lateral de navegação do aplicativo.	
	8. O sistema abre a tela de listagem de clientes e faz uma requisição para o Firebase para trazer todos os clientes cadastrados da empresa.
9. O usuário toca em um item da lista que está mostrando um cliente.	
	10. O sistema abre a tela de detalhes de cliente e faz uma requisição para o Firebase para trazer todos os dados do cliente selecionado. Os dados recebidos são então exibidos em tela.
11. O usuário visualiza os dados, então desce até a parte inferior da tela e toca no botão “Editar Cliente”.	
	12. O sistema abre a tela de edição de cliente e preenche os campos de digitação e seleção com os dados do cliente que se está editando.
13. O usuário altera as informações desejadas e toca no botão “Salvar Cliente”.	
	14. O sistema faz uma requisição para o Firebase para editar os dados do cliente no banco de dados.
15. Na tela de detalhes de cliente o usuário toca no botão “Excluir Cliente”.	
	16. O sistema faz uma requisição para o Firebase para excluir o usuário do banco de

	dados e logo em seguida redireciona o usuário para a tela de listagem de clientes.
Validações	Para o cadastro e edição de cliente todos os dados obrigatórios devem ser preenchidos, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020)

Tabela 6 - Caso de uso “Manter Tipos de Serviço”.

Nome do caso de uso	Manter Tipos de Serviço.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos necessários para um usuário manter tipos de serviço.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Novo Tipo de Serviço” no menu lateral de navegação.	
	2. O sistema abre a tela de cadastro de um novo tipo de serviço.
3. O usuário preenche o nome e a descrição do tipo de serviço e toca no botão “Salvar”.	
	4. O sistema faz uma requisição para ao Firebase para cadastrar o novo tipo de serviço no banco de dados.

5. O usuário toca no item “Tipos de Serviço” no menu lateral de navegação.	
	6. O sistema abre a tela de listagem de tipos de serviço e faz uma requisição para o Firebase para trazer todos os tipos de serviço cadastrados da empresa.
7. O usuário visualiza todos os dados do tipo de serviço e então toca no botão com o ícone de lixeira na direita do item que representa um tipo de serviço.	
	8. O sistema faz uma requisição de exclusão para o Firebase para excluir o tipo de serviço do banco de dados.
Validações	Para o cadastro de um novo tipo de serviço o usuário precisa preencher o nome apenas.

Fonte: Elaborado pelos autores (2020).

Tabela 7 – Caso de uso “Manter Etapas de *Marketing*”.

Nome do caso de uso	Manter Etapas de *Marketing.*
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para se manter etapas de *marketing*.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Nova Etapa de *Marketing*” no menu lateral de navegação.	

	2. O sistema abre a tela de cadastro de uma nova etapa de *marketing*.
3. O usuário preenche os dados e toca no botão “Cadastrar Etapa”.	
	4. O sistema faz uma requisição para ao Firebase para cadastrar a etapa de *marketing* no banco de dados e depois limpa os campos para possibilitar um novo cadastro.
5. O usuário toca no item “Etapas de *Marketing*” no menu lateral de navegação.	
	6. O sistema abre a tela de listagem de etapas de *marketing* e faz uma requisição para o Firebase para buscar todas as etapas cadastradas da empresa.
7. O usuário visualiza a listagem e toca em um item, que representa uma etapa de *marketing*.	
	8. O sistema abre a tela de detalhes de etapa de *marketing* e faz uma requisição para o Firebase para trazer todos os dados da etapa selecionada.
9. O usuário visualiza os dados, desce até a parte inferior da tela e toca no botão “Editar”.	
	10. O sistema abre a tela de edição de etapa de *marketing* com os campos preenchidos com os dados da etapa.
11. O usuário edita os dados desejados e toca em “Editar Etapa”.	
	12. O sistema faz uma requisição para ao Firebase para editar os dados da etapa no

	banco de dados e retorna o usuário para a tela de detalhes de etapa de *marketing*.
14. O usuário toca no botão “Excluir”.	
	15. O sistema abre uma caixa de diálogo de confirmação.
16. O usuário toca em “Excluir”.	
	17. O sistema faz uma requisição para ao Firebase para excluir a etapa do banco de dados e depois redireciona o usuário para a tela de listagem de etapas.
Validações	Para o cadastro e edição de etapa de *marketing* todos os campos obrigatórios devem ser preenchidos, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 8 – Caso de uso “Manter Serviços”.

Nome do caso de uso	Manter Serviços.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos necessários para o usuário manter serviços.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Novo Serviço” no menu lateral de navegação.	

	2. O sistema abre a tela de cadastro de serviço.
3. O usuário preenche os campos de seleção e toca no botão “Cadastrar”.	
	4. O sistema faz uma requisição para o Firebase para salvar o novo serviço no banco de dados.
5. O usuário toca no item “Serviços” no menu lateral de navegação.	
	6. O sistema abre a tela de listagem de serviços e faz uma requisição para o Firebase para trazer todos os serviços cadastrados da empresa.
7. O usuário toca em um item da lista que representa um serviço.	
	8. O sistema abre a tela de detalhes de serviço com a aba de detalhes selecionada e faz uma requisição para o Firebase para trazer todos os dados do serviço selecionado.
9. O usuário visualiza as informações e desce até a parte inferior da aba de detalhes para, em seguida, tocar no botão “Editar Serviço”.	
	10. O sistema abre a tela de edição de serviço com os campos de seleção preenchidos com os dados do serviço que se está editando.
11. O usuário altera as informações desejadas e toca no botão “Salvar Modificações”.	
	12. O sistema faz uma requisição para o Firebase para editar os dados do serviço e retorna para a tela de detalhes com a aba de detalhes selecionada.

13. O usuário na tela de detalhes de serviço toca nos três pontos verticais na barra de navegação da página e depois toca no item “Excluir Serviço”.	
	14. O sistema faz uma requisição para o Firebase para excluir o serviço do banco de dados e, logo em seguida, redireciona o usuário para a tela de listagem de serviços.
Validações	Para o cadastro e edição de serviço os campos obrigatórios devem ser preenchidos, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 9 – Caso de uso “Manter Avaliação do Serviço”.

Nome do caso de uso	Manter Avaliação do Serviço.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos necessários para o usuário manter avaliações de serviços.
Ações do ator	**Ações do Sistema**
1. O usuário abre a aba de avaliação de serviço dentro da tela de detalhes de serviço.	
	2. O sistema mostra os dados da avaliação realizada (se existirem) ou uma mensagem e um botão para cadastrar a avaliação.
3. O usuário toca no botão “Avaliar Serviço”.	

	4. A tela de cadastro de avaliação de serviço é aberta.
5. O usuário preenche os dados da avaliação e toca no botão “Salvar Avaliação”.	
	6. O sistema faz uma requisição para o Firebase para salvar a avaliação no banco de dados e retorna o usuário à tela de detalhes do serviço com a aba de avaliação selecionada.
	7. O sistema exibe a avaliação cadastrada.
8. O usuário visualiza a avaliação realizada e toca no botão “Editar Avaliação”.	
	9. O sistema abre a tela de edição de avaliação de serviço, carregando os dados nos campos de seleção e digitação.
10. O usuário edita as informações desejadas e toca no botão “Salvar Edição”.	
	11. O sistema faz uma requisição para o Firebase para editar a avaliação do serviço e retorna o usuário para a tela de detalhes de serviço com a aba de avaliação selecionada.
12. O usuário toca no botão com o ícone de lixeira.	
	13. O sistema abre uma caixa de confirmação da ação de excluir.
14. O usuário toca no botão “Excluir”.	
	15. O sistema faz uma requisição para o Firebase para excluir a avaliação do serviço e

	volta a mostrar a mensagem e o botão para avaliar o serviço.
Validações	Para o cadastro e edição de avaliação de serviço devem ser preenchidos todos os campos obrigatórios, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 10 – Caso de uso "Manter Retornos do Cliente".

Nome do caso de uso	Manter Retornos do Cliente.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para manter os retornos do cliente ao estabelecimento comercial.
Ações do ator	**Ações do Sistema**
1. O usuário abre a aba de retornos dentro da tela de detalhes de um serviço.	
	2. O sistema faz uma requisição para o Firebase para buscar todos os retornos do cliente do serviço que se está vendo os detalhes.
3. O usuário toca no botão flutuante na parte inferior da tela, o qual possui um ícone da operação soma.	
	4. O sistema abre a tela de cadastro de retorno do cliente.

5. O usuário preenche os campos e toca no botão “Cadastrar Retorno”.	
	6. O sistema faz uma requisição para o Firebase para cadastrar o retorno do cliente por causa do serviço no banco de dados e redireciona o usuário para a tela de detalhes de serviço com a aba de retornos selecionada.
7. O usuário visualiza os retornos cadastrados e toca em um item da lista que representa um retorno.	
	8. O sistema abre a tela de edição com os campos de digitação e seleção preenchidos.
9. O usuário edita as informações desejadas e toca no botão “Editar Retorno”.	
	10. O sistema faz uma requisição para o Firebase para editar o retorno do cliente e redireciona o usuário para a tela de detalhes de serviço com a aba de retornos selecionada.
11. O usuário toca no botão com ícone de lixeira em um item da listagem de retornos.	
	12. O sistema abre uma caixa de confirmação.
13. O usuário seleciona a opção “Excluir”.	
	14. O sistema faz uma requisição para o Firebase para excluir o retorno do cliente do banco de dados.
Validações	Para o cadastro e edição de retorno todos os campos obrigatórios devem ser preenchidos, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 11 – Caso de uso “Manter Motivos de Retorno”.

Nome do caso de uso	Manter Motivos de Retorno.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para o usuário manter motivos de retorno do cliente.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Novo Motivo de Retorno” no menu lateral de navegação.	
	2. O sistema abre a tela de cadastro de motivo de retorno.
3. O usuário preenche os campos e toca no botão “Salvar Motivo”.	
	4. O sistema faz uma requisição para o Firebase para salvar o novo motivo de retorno no banco de dados e limpa os campos para possibilitar um novo cadastro.
5. O usuário toca no item “Motivos de Retorno” no menu lateral de navegação.	
	6. O sistema faz uma requisição para o Firebase para trazer todos os motivos de retornos cadastrados da empresa.
7. O usuário visualiza os motivos de retorno e toca no botão com o ícone de lixeira em um item.	

	8. O sistema mostra uma caixa de confirmação.
9. O usuário toca no botão “Excluir”.	
	10. O sistema faz uma requisição para o Firebase para excluir o motivo de retorno do banco de dados.
Validações	Para o cadastro de motivo de retorno todos os campos obrigatórios devem ser preenchidos, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 12 – Caso de uso “Manter Usuário”.

Nome do caso de uso	Manter Usuário.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para manter usuário.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Seu Perfil” do menu lateral de navegação.	
	2. O sistema abre a tela de perfil do usuário logado.
3. O usuário toca no botão “Dados do Perfil”.	

	4. O sistema abre a tela de dados do perfil do usuário logado.
5. O usuário visualiza os dados do seu perfil e toca no item que mostra o seu nome.	
	6. O sistema abre a tela de edição de nome de usuário e preenche o campo de nome com o nome do usuário autenticado.
7. O usuário altera o nome de usuário e toca no botão “Salvar Modificações”.	
	8. O sistema faz uma requisição para o Firebase para o Firebase para editar o nome do usuário logado no banco de dados e depois redireciona o usuário para a tela de dados do perfil.
9. O usuário toca no item que mostra o *e-mail* da conta.	
	10. O sistema abre a tela de edição de *e-mail* do usuário logado.
11. O usuário edita o *e-mail* e toca no botão “Salvar Modificações”.	
	12. O sistema mostra uma caixa de confirmação no qual o usuário deve digitar a senha da sua conta.
13. O usuário digita a senha da conta e toca no botão “Confirmar”.	
	14. O sistema faz uma requisição para o Firebase para editar o *e-mail* da conta do usuário logado e retorna o usuário para a tela de dados do perfil.

15. Na tela de perfil do usuário o usuário toca no botão “Segurança”.	
	16. O sistema abre a tela de opções de segurança.
17. O usuário seleciona a opção “Alterar a Senha da Conta”.	
	18. O sistema abre a tela de alteração de senha do usuário logado.
19. O usuário preenche os campos de senha atual, nova senha e confirmação da nova senha para, logo em seguida, tocar no botão “Alterar Senha”.	
	20. O sistema faz uma requisição para o Firebase para alterar a senha da conta do usuário e então faz o usuário voltar para a tela de *login*.
Validações	Nas telas de edição de dados do perfil e na tela de trocar a senha da conta os campos obrigatórios precisam ser preenchidos, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 13 – Caso de uso “Manter Empresa”.

Nome do caso de uso	Manter Empresa.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para manter empresa.

Ações do ator	Ações do Sistema
1. O usuário toca no botão “Cadastrar Empresa” na tela de *login*.	
	2. O sistema abre a tela de cadastro de empresa.
3. O usuário preenche os campos e toca no botão “Salvar”.	
	4. O sistema faz uma requisição para o Firebase para cadastrar a empresa e faz outra requisição para criar o usuário administrador da empresa. Depois que as requisições terminarem o usuário é redirecionado para a tela inicial do sistema.
5. O usuário toca no item “Sobre a Empresa” no menu lateral de navegação.	
	6. O sistema abre a tela de detalhes da empresa do usuário logado.
7. O usuário visualiza as informações da empresa logada e toca no botão “Editar Empresa”.	
	8. O sistema abre a tela de edição dos dados da empresa do usuário logado.
9. O usuário edita as informações desejadas e toca no botão “Salvar Modificações”.	
	10. O sistema faz uma requisição para o Firebase para editar os dados da empresa e em seguida retorna o usuário para a tela de detalhes da empresa.

Validações	Na tela de cadastro e edição é necessário o preenchimento dos campos obrigatórios, respeitando as máscaras e formatos pré-definidos.

Fonte: Elaborado pelos autores (2020).

Tabela 14 – Caso de uso "Manter Configuração de Mensagens de Aniversário".

Nome do caso de uso	Manter Configuração de Mensagens de Aniversário.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para manter a configuração de mensagens de aniversário.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item "Conf. Msgs. Aniversário" no menu lateral de navegação.	
	2. O sistema abre a tela de configuração de mensagens de aniversário.
3. O usuário altera os campos e toca no botão "Salvar Configuração".	
	4. O sistema faz uma requisição para o Firebase para alterar a configuração de mensagens de aniversário no banco de dados.
Validações	Nenhuma validação é realizada.

Fonte: Elaborado pelos autores (2020).

Tabela 15 – Caso de uso “Enviar Mensagens de Aniversário”.

Nome do caso de uso	Enviar Mensagens de Aniversário.
Atores envolvidos	Usuário, Cliente, Firebase.
Objetivo	Este caso de uso descreve os passos para enviar mensagens de aniversário para clientes.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item “Aniversariantes” no menu lateral de navegação.	
	2. O sistema abre a tela de listagem de aniversariantes e faz uma requisição para o Firebase para trazer todos os clientes aniversariantes do dia e mês atual.
3. O usuário toca no botão com três pontos verticais em um item da listagem.	
	4. O sistema mostra uma caixa de seleção de ações para o aniversariante selecionado.
5. O usuário toca em uma ação. Cada ação é um canal para o envio de mensagens ou realizar ligação telefônica, no caso da ação de ligar para o aniversariante.	
	6. O sistema abre o aplicativo para enviar a mensagem para o cliente aniversariante e faz uma requisição para o Firebase para salvar uma *flag* que indica que a mensagem foi enviada para o aniversariante.

7. O usuário envia a mensagem de aniversário para o cliente.	
Validações	Nenhuma validação é realizada.

Fonte: Elaborado pelos autores (2020).

Tabela 16 – Caso de uso "Realizar Comunicação Após Realização de Serviço".

Nome do caso de uso	Realizar Comunicação Após Realização de Serviço
Atores envolvidos	Usuário, Cliente, Firebase
Objetivo	Este caso de uso descreve os passos para se realizar a comunicação com clientes após a realização de serviços.
Ações do ator	**Ações do Sistema**
1. O usuário abre a aba "Enviar Mensagens" dentro da tela de detalhes de serviço.	
	2. O sistema mostra a aba e faz uma requisição para o Firebase para buscar todas as etapas de *marketing*. Com a requisição realizada o sistema exibe uma listagem de etapas para o envio de mensagens.
3. O usuário seleciona um botão presente dentro de um item (cada botão é um canal de comunicação com o cliente).	
	4. O sistema faz uma requisição para o Firebase para salvar no banco de dados que o canal de comunicação com o cliente da etapa de *marketing* selecionada foi utilizado e o aplicativo que será usado para a comunicação é aberto.

5. O usuário envia a mensagem para o ciente.	
Validações	Só é possível enviar mensagens se não foram encerradas as atividades. de pós-marketing do serviço e se o cliente pode receber mensagens de pós-marketing.

Fonte: Elaborado pelos autores (2020).

Tabela 17 – Caso de uso "*Deslogar* da Conta".

Nome do caso de uso	Deslogar da Conta.
Atores envolvidos	Usuário, Firebase.
Objetivo	Este caso de uso descreve os passos para que o usuário logado saia de sua conta.
Ações do ator	**Ações do Sistema**
1. O usuário toca no item "Seu Perfil" no menu lateral de navegação.	
	2. O sistema abre a tela de perfil do usuário logado.
3. O usuário toca no botão "Sair da Conta".	
	4. O sistema envia uma requisição para o Firebase para deslogar o usuário e retorna ele para a tela de *login*.
Validações	Nenhuma validação é realizada.

Fonte: Elaborado pelos autores (2020).

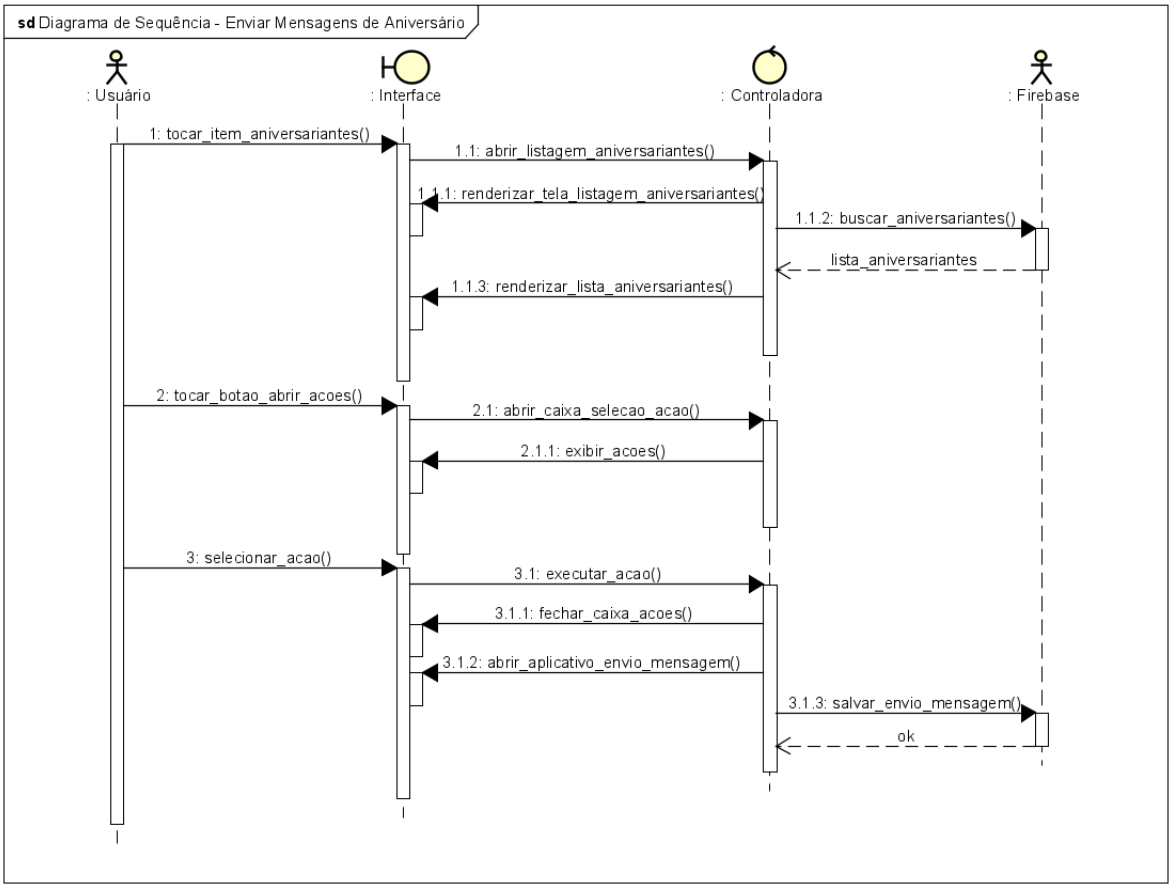
3.3 Diagramas de Sequência

O diagrama de sequência é mais um dos diagramas da UML. Ele determina a ordem dos ventos, as mensagens trocadas entre as entidades e os métodos chamados em um processo. Gilleanes (2009, p. 200) aponta que o diagrama de sequência tem uma relação muito forte com o diagrama de casos de uso, “havendo [...] um diagrama de sequência para cada diagrama de caso de uso”. Isso pode ser notado pela presença do ator nos mesmos.

Por mais que cada caso de uso possa ter seu diagrama de sequência, por via de facilitar o entendimento e proporcionar uma visão geral da sequência dos casos de uso principais da aplicação apenas os diagramas de sequência dos casos de uso de envio de mensagens para clientes serão apresentados nesse trabalho. As demais sequências, além de serem muito similares, já são brevemente, e de forma menos objetiva, explicadas na documentação dos casos de uso e no capítulo de telas do sistema.

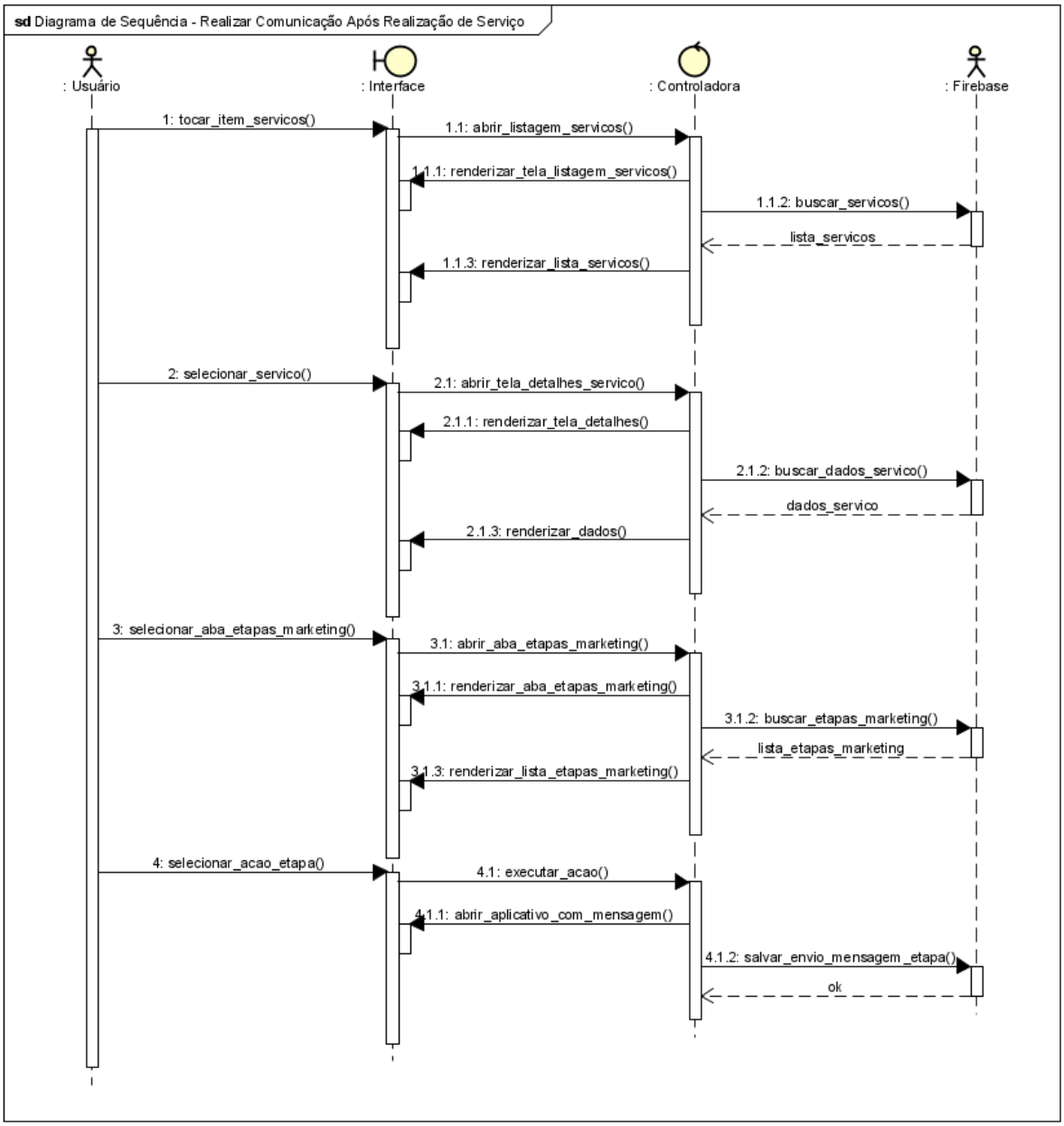
A Figura 2 abaixo ao diagrama de sequência do requisito funcional “Enviar Mensagens de Aniversário” e a Figura 3 exibe o diagrama de sequência do requisito funcional “Realizar Comunicação Após Realização de Serviço”. Para ambos os diagramas serão mostradas as ações no caso de sucesso, ou seja, quando as validações realizadas passam.

Figura 2 - Diagrama de sequência "Enviar Mensagens de Aniversário".



Fonte: Elaborado pelos autores (2020).

Figura 3 - Diagrama de sequência "Realizar Comunicação Após Realização de Serviço".



Fonte: Elaborado pelos autores (2020).

3.4 Banco de Dados

Esta seção trata sobre o banco de dados da aplicação. Nele será apresentada e explicada a escolha realizada e a estrutura da base de dados, que será explanada através do uso do dicionário de dados e do Diagrama de Entidade e Relacionamento (DER).

Um sistema de banco de dados (DBMS) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados. A coleção de dados, normalmente chamada de banco de dados, contém informações relevantes a uma empresa (SILBERSCHATZ, 2012).

3.4.1 Escolha do Banco de Dados

Após uma série discussões foi decidida a utilização das soluções disponibilizados pelo Firebase, que é um produto Google. Ele não é só um banco de dados, mas uma solução completa para complexos problemas de uma aplicação, como notificações em tempo real, autenticação, testes, análises do uso dos usuários e muito mais, por isso os desenvolvedores comumente o chamam de "*back-end as a service" (*BAAS*)*.

A vantagem do Firebase no nosso caso é o desenvolvimento acelerado, fácil criação e modificação da estrutura do banco de dados, o armazenamento de dados offline, uma vez que os dados são gravados no dispositivo do usuário, permanecendo disponíveis sem acesso à internet e são sincronizados quando o aparelho se conecta à rede, além de muitas outras vantagens.

Sua integração com o React Native a partir de bibliotecas *open source* no GitHub possibilitou a consulta dos problemas e soluções criadas pela comunidade que ajuda muito nas horas que encontramos erros ou não sabemos fazer uma demanda, além das excelentes documentações, que suprem a maioria das nossas dúvidas técnicas.

Utilizamos muito o Firestore, que é muito semelhante a soluções existentes e já consolidadas no mercado, como o MongoDB. Sua estrutura é toda baseada em documentos JSON, o que permite que alguns documentos possuam campos a mais que outros, contudo, com o devido cuidado isto se torna algo que podemos usar ao nosso favor.

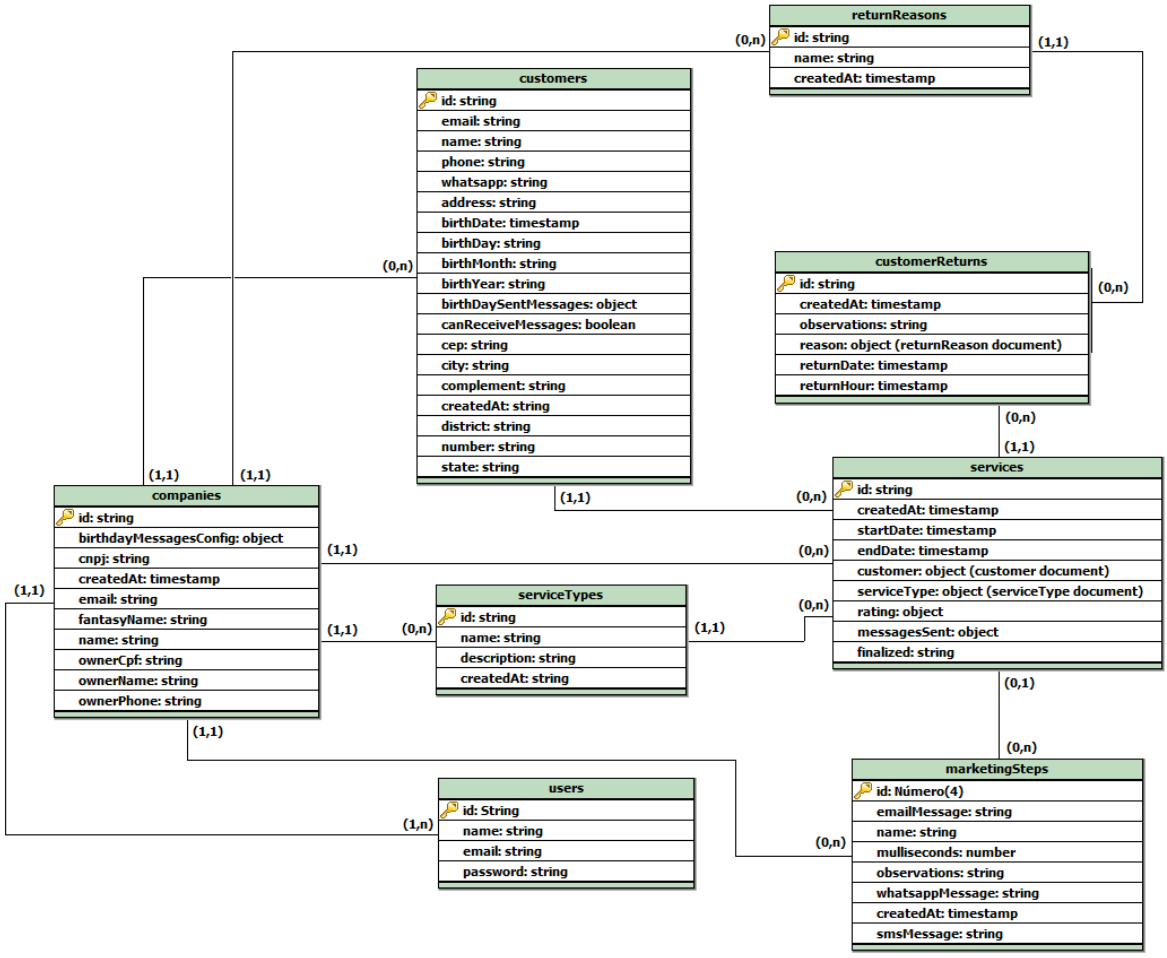
Sabemos das dificuldades técnicas de um banco NoSQL, como a dificuldade em fazer pesquisas complexas de documentos ou os próprios relacionamentos entre documentos, mas as vantagens que enxergamos foram demasiadamente maiores que as adversidades para este projeto em específico.

3.4.2 DER

Um DER nada mais é que a representação gráfica do modelo MER. Em termos conceituais é possível dizer que o DER é um modelo diagramático que descreve o modelo de dados de um sistema com alto nível de abstração. Ele é usado para representar o modelo conceitual do negócio (BALBO, 2010).

Para fornecer uma visão geral das coleções de dados um Diagrama de Entidade e Relacionamento (DER) foi elaborado, porém ele é uma versão adaptada para nosso caso em específico. Nele utilizamos os tipos aceitos em um JSON ao invés dos tipos existentes em bancos relacionais para um melhor entendimento da estrutura.

Figura 4 - Diagrama de Entidade e Relacionamento (DER).

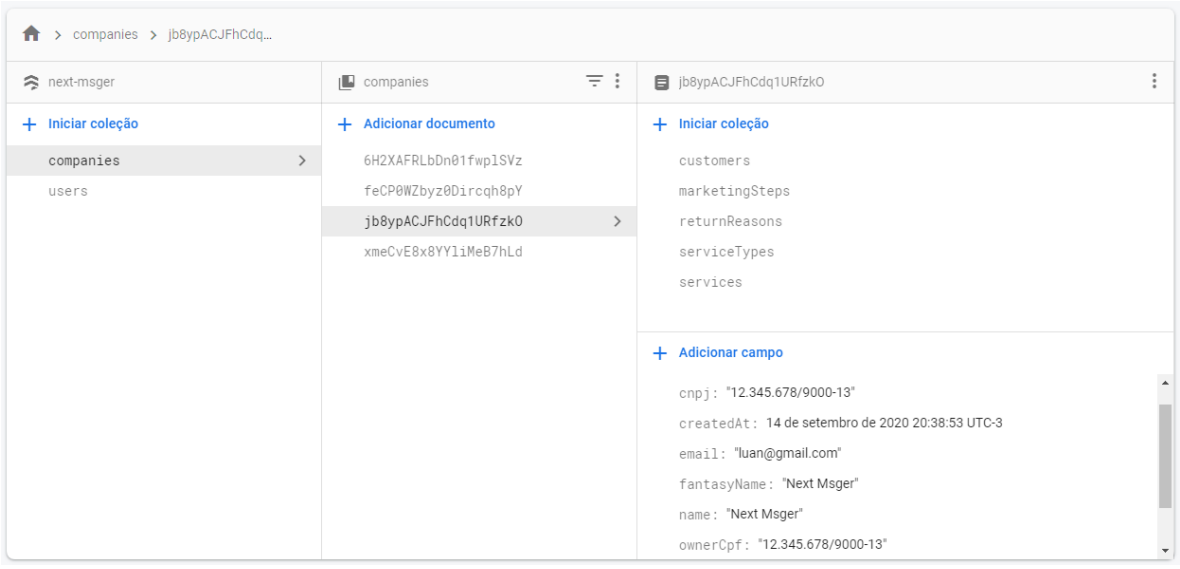


Fonte: Elaborado pelos autores (2020).

O aplicativo utiliza, em algumas telas, dados de vários documentos combinados, porém os documentos não se conhecem, eles apenas são recipientes que guardam dados, quem sabe orquestrá-los e construir as informações é o *software*, isto significa que os relacionamentos estão no nível da aplicação e não ao nível de banco de dados, como acontece nos relacionais. Há além disso, outros casos em que um documento possui todos ou quase todos os dados de outro, isto é uma técnica bem comum em estruturas não relacionais para melhorar a performance na obtenção dos dados e ter certo nível de redundância.

A Figura 5 mostra a estrutura do Firestore da aplicação. Como é possível ver, existem duas coleções de primeiro nível, ou seja, que estão na raiz e elas são a *users* e *companies*, as quais armazenam dados de usuários e de empresas, respectivamente. Dentro de um documento de uma empresa existem outras *collections,* as chamadas *subcollections,* que são coleções como qualquer outra, a única diferença é que estão dentro de um documento. Esta estrutura nos permite armazenar tudo o que for referente a uma empresa em suas coleções próprias, tornando a obtenção dos dados de empresas mais fácil.

Figura 5 - Visão geral das coleções do sistema no Firestore.



Fonte: Elaborado pelos autores (2020).

3.4.3 Dicionário de Dados

O dicionário de dados representa em forma de tabelas o banco de dados da aplicação. Com ele é possível visualizar e entender completamente o que cada documento JSON dentro de uma collections realmente armazena, o que é mais difícil olhando apenas o DER pela falta de uma descrição.

Como no Firestore não temos relacionamentos a nível de banco de dados o dicionário não terá nenhuma coluna indicando se o campo é primary key ou foreign key. Além disso, é possível que um campo armazene vetores e outros objetos, que são tipos não existentes em bancos de dados SQL, mas que podem ser utilizados em uma árvore JSON, por isso da Figura 6 a Figura 12 são apresentadas as estruturas destes objetos presentes em alguns documentos do banco de dados.

Tabela 18 – Coleção *customers* (dicionário de dados).

Nome da Coleção	*customers*	
Descrição	Armazena os clientes da empresa.	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
email	*string*	*E-mail* do cliente.
name	*string*	Nome do cliente.
phone	*string*	Número de telefone fixo do cliente.
whatsapp	*string*	Número de celular/WhatsApp do usuário.
canReceiveMessages	*boolean*	Se *true* o cliente pode receber mensagens de pós-marketing. Se *false* ele não recebe nenhuma mensagem.
createdAt	*timestamp*	Data de criação do registro em formato UTC.
city	*string*	Cidade que o cliente mora.

complement	*string*	Complemento ao endereço.
cep	*string*	Número do CEP do cliente.
district	*string*	Bairro que o cliente mora.
number	*string*	Número da residência do cliente.
address	*string*	Rua que o cliente mora.
state	*string*	Estado que o cliente mora.
birthDate	*timestamp*	Data de nascimento.
birthDay	*string*	Dia do nascimento.
birthMonth	*string*	Mês de nascimento.
birthYear	*string*	Ano de nascimento.
birthDaySentMessages	*object*	Armazena *flags* para cada ano pra saber se a mensagem de aniversário foi enviada para o cliente.

Fonte: Elaborado pelos autores (2020).

Tabela 19 - Coleção *marketingSteps* (dicionário de dados).

Nome da Coleção	*marketingSteps*	
Descrição	Armazena todas as etapas de *marketing.*	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
createdAt	*timestamp*	Data de criação do registro em formato UTC.
emailMessage	*string*	Mensagem para enviar por *e-mail* para os clientes.

name	*string*	Nome da etapa.
milliseconds	*number*	Quantidade de tempo, em milissegundos, após o término do serviço que é ideal enviar as mensagens de pós-marketing.
observations	*string*	Observações da etapa.
whatsappMessage	*string*	Mensagem para enviar por WhatsApp para os clientes.
smsMessage	*string*	Mensagem para enviar por SMS para os clientes.

Fonte: Elaborado pelos autores (2020).

Tabela 20 - Coleção *returnReasons* (dicionário de dados).

Nome da Coleção	*returnReasons*	
Descrição	Armazena os motivos pelos quais um cliente pode retornar fisicamente ao estabelecimento da empresa.	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
name	*string*	Nome do motivo de retorno.
createdAt	*timestamp*	Data de criação do registro em formato UTC.

Fonte: Elaborado pelos autores (2020).

Tabela 21 - Coleção *serviceTypes* (dicionário de dados).

Nome da Coleção	*serviceTypes*	
Descrição	Armazena os tipos de serviço que a empresa realiza.	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
name	*string*	Nome do tipo de serviço.
description	*string*	Descrição do tipo de serviço.
createdAt	*timestamp*	Data de criação do registro em formato UTC.

Fonte: Elaborado pelos autores (2020).

Tabela 22 - Coleção *users* (dicionário de dados).

Nome da Coleção	*users*	
Descrição	Armazena os usuários da aplicação. OBS: O ID dos documentos são os mesmos que os dos usuários dos provedores de autenticação (*E-mail*, Google, Facebook...).	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
name	*string*	Nome do usuário.
companyId	*string*	ID da empresa que o usuário pertence.

Fonte: Elaborado pelos autores (2020).

Tabela 23 - Coleção *services* (dicionário de dados).

Nome da Coleção	*services*	
Descrição	Armazena os serviços cadastrados.	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
createdAt	*timestamp*	Data de criação do registro em formato UTC.

customer	*object*	Armazena o cliente do serviço. Ele possui todos os atributos de um documento da *collection* "*customers*".
endDate	*timestamp*	Data em que o serviço foi finalizado.
messagesSent	*object*	Armazena quais mensagens de quais etapas de pós-marketing já foram enviadas. O id do documento é o mesmo que o das etapas de *marketing* e os atributos são *flags* que se existirem e tiverem como "*true*" significa que a mensagem já foi enviada.
rating	*object*	Armazena a avaliação do serviço feita pelo cliente. É armazenada uma nota de 1 a 5 e um comentário.
serviceType	*object*	Armazena o tipo do serviço. Ele possui todos os atributos de um documento da *collection* "*serviceTypes*".
startDate	*timestamp*	Data em que o serviço foi iniciado.

Fonte: Elaborado pelos autores (2020).

Tabela 24 - Coleção *customerReturns* (dicionário de dados).

Nome da Coleção	*customerReturns*	
Descrição	Armazena os retornos do cliente ao estabelecimento.	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
reason	*object*	Armazena a cópia de um documento da *collection* "*returnReasons*".
observations	*string*	Observações do retorno do cliente.
createdAt	*timestamp*	Data de criação do registro em formato UTC.
returnDate	*timestamp*	Data que o cliente retornou ao estabelecimento.
returnHour	*timestamp*	Hora que o cliente retornou ao estabelecimento.

Fonte: Elaborado pelos autores (2020).

Tabela 25 - Coleção *companies* (dicionário de dados).

Nome da Coleção	*companies*	
Descrição	Armazena as empresas cadastradas.	
Estrutura dos Documentos		
Nome	**Tipo**	**Descrição**
id	*string*	Identificador único. Uma *string* com 28 caracteres contendo letras e números.
name	*string*	Razão social da empresa.
fantasyName	*string*	Nome fantasia da empresa.
email	*string*	*E-mail* da empresa.
cnpj	*string*	CNPJ da empresa.
ownerCpf	*string*	CPF do dono da empresa.
ownerName	*string*	Nome do dono da empresa.
ownerPhone	*string*	Telefone ou celular do dono da empresa.
createdAt	*timestamp*	Data de criação do registro em formato UTC.
birthdayMessagesConfig	*object*	Configurações de mensagens de aniversário da empresa.

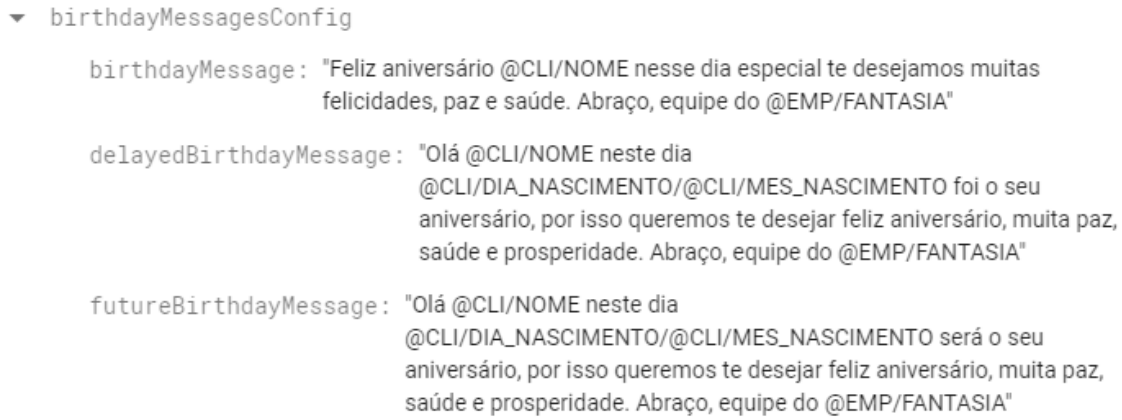
Fonte: Elaborado pelos autores (2020).

3.4.4 Estrutura dos objetos dentro de documentos

No dicionário de dados não é possível documentar a estrutura de objetos dentro de documentos, seja ele um objeto com uma estrutura fixa ou os que possuem uma estrutura variável, isto é, as chaves que armazenam valores podem variar. Também não é possível documentar quando existem objetos dentro de objetos em um documento.

A Figura 6 representa o objeto que armazena a configuração de mensagens de aniversário dentro do documento que armazena os dados da empresa, pertencente a coleção de empresas.

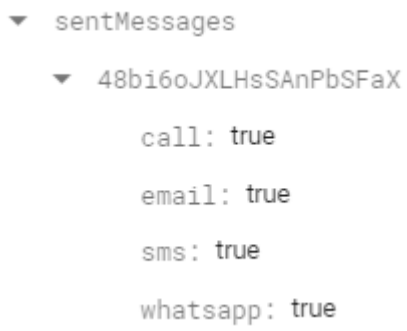
Figura 6 - Objeto que armazena a configuração de mensagens de aniversário da empresa.



Fonte: Elaborado pelos autores (2020).

A Figura 7 representa o objeto que armazena os canais de comunicação utilizados para falar com o cliente em uma etapa de *marketing*. Ele fica localizado dentro dos documentos da coleção *services* e é um objeto que armazena outros objetos, sendo que a chave dos objetos no seu interior é o identificador da etapa de *marketing*. O conteúdo do objeto são *flags*, que quando estão com o valor *true* significa que o canal foi utilizado.

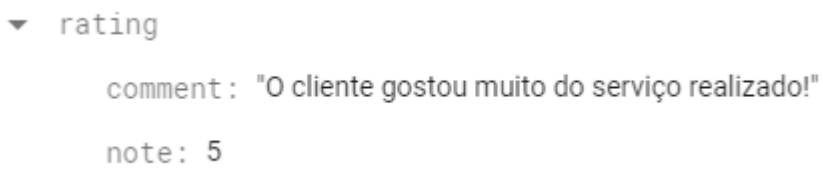
Figura 7 - Objeto que armazena os canais utilizados para falar com o cliente em uma etapa de *marketing.*



Fonte: Elaborado pelos autores (2020).

A Figura 8 representa o objeto que armazena a avaliação do serviço feita pelo cliente. Ele fica localizado nos documentos da coleção *services* do banco de dados. Sua estrutura é bem simples e fixa, armazenando a nota atribuída e as observações realizadas.

Figura 8 - Objeto que armazena a avaliação do serviço feita por um cliente.



Fonte: Elaborado pelos autores (2020).

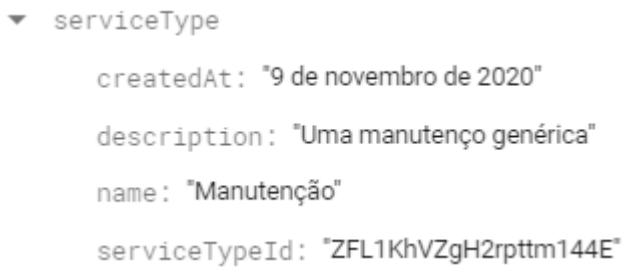
A Figura 9 representa o objeto que armazena exatamente tudo o que um documento de cliente possui. Ele fica localizado dentro dos documentos da coleção *services* para que se possa saber que cliente pagou pelo serviço. Aqui é possível ver a estratégia de bancos não relacionais de réplica de dados (o que é muito utilizado).

Figura 9 – Objeto réplica de um documento de cliente dentro de um documento de serviço.

Fonte: Elaborado pelos autores (2020).

A Figura 10 representa o objeto que é exatamente igual a um documento de tipo de serviço. Ele fica localizado dentro dos documentos da coleção *services* e guarda qual foi o tipo do serviço realizado. Nele novamente é utilizada a técnica de réplica de dados.

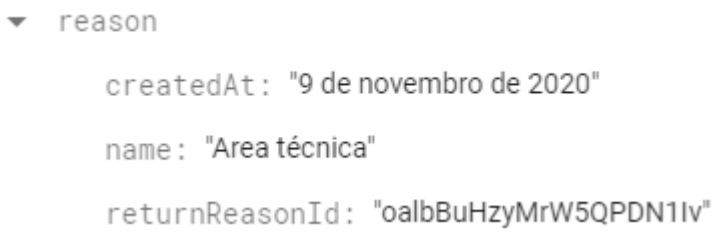
Figura 10 - Objeto réplica de um documento de tipo de serviço dentro de um documento de serviço.



Fonte: Elaborado pelos autores (2020).

A Figura 11 representa o objeto que armazena o motivo de um retorno do cliente. Ele fica localizado dentro da coleção de retornos de cliente (que fica dentro de documentos de serviço) e é uma cópia idêntica do documento da coleção de motivos de retorno que foi usado para cadastrar o retorno.

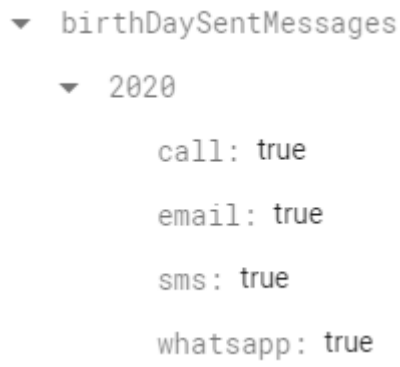
Figura 11 - Objeto que armazena o motivo de um retorno do cliente.



Fonte: Elaborado pelos autores (2020).

A Figura 12 representa o objeto que armazena os canais utilizados para enviar a mensagem de aniversário de um cliente. Este objeto possui outros objetos que armazenam uma *flag* para cada canal utilizado e as chaves utilizadas é o ano em que as mensagens foram enviadas. Além disso ele fica localizado nos documentos da coleção *customers*, assim facilita saber se o cliente recebeu os parabéns em algum ano.

Figura 12 - Objeto que armazena os canais utilizados para enviar a mensagem de aniversário para o cliente em algum ano.



Fonte: Elaborado pelos autores (2020).

4 DESENVOLVIMENTO

O SCRUM é uma estrutura metodológica que é usada para implementar o desenvolvimento ágil em projetos de *software*, negócios, entre outros. Esta metodologia pode ajudar na organização de equipe e ter mais trabalho feito em menos tempo garantindo, desta forma, uma maior eficácia e eficiência.

Os projetos SCRUM são divididos em diversos ciclos de desenvolvimento chamados *sprints*. Elas são etapas que duram determinado espaço de tempo nas quais várias atividades são executadas. Estas atividades são as funcionalidades contidas em uma lista chamada de *Product Backlog*.

Em cada *sprint* a *Sprint Planning Meeting* é realizada, ou seja, uma reunião de planejamento na qual itens do *Product Backlog* são priorizados e a equipe seleciona quais deles ela irá desenvolver. Ao final do planejamento as tarefas a serem feitas na *sprint* são movidas do *Product Backlog* para o *Sprint Backlog*.

Em todos os dias durante o desenrolar de uma *sprint* acontece uma breve reunião (normalmente de manhã), chamada de *Daily Scrum*, cujo objetivo é manter os membros da equipe atualizados sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.

Quando se chega ao final de uma *sprint*, a equipe *apresenta* as funcionalidades desenvolvidas na chamada *Review Meeting*, a *Sprint* Retrospective é feita e a equipe parte para o planejamento da próxima *sprint*.

Na metodologia SCRUM podemos encontrar três principais funções para os membros da equipe que desenvolverá o projeto. A primeira função é o *Product Owner* (PO), ele é responsável por definir os itens que farão parte do *Product Backlog* e as respectivas prioridades, o PO também é o membro que tem o conhecimento da regra de negócio. Também está presente na equipe o *Scrum Master*, esse cargo é tipicamente exercido por um gerente ou um líder técnico, mas pode ser qualquer pessoa da equipe, a principal função do *Scrum Master* é assegurar que a equipe não se comprometa excessivamente durante a *sprint* e assegurar que a equipe respeite as práticas da SCRUM. A última função é o *Scrum Team*, que é a equipe de desenvolvimento. Nela não existe uma divisão através de papeis (como programadores, analistas, designers, entre outros), todos trabalham para terminar o projeto entregando o que foi prometido na *sprint*.

Neste projeto o grupo está dividido da seguinte forma: o membro Luan Eduardo tem as funções de *Product Owner*, já que o mesmo foi o idealizador da aplicação, portanto conhece todas as regras de negócio e também é o *Scrum Master*, já que possui mais experiência em desenvolvimento e o conhecimento técnico necessário para auxiliar o grupo. O restante do grupo, Rafael Donizete, é membro pertencente ao *Scrum Team*, realizando as tarefas selecionadas em cada *sprint*.

4.1 Etapas de Desenvolvimento

O projeto foi segmentado em diversas *sprints* de desenvolvimento. Em cada uma delas certa quantidade de itens do *Product Backlog* foi desenvolvida pelos integrantes da equipe.

Na primeira *sprint*, a prioridade foi a criação e configuração do projeto, integração com o Firebase, junto com o desenvolvimento de algumas telas e componentes essenciais para o aplicativo funcionar. Inicialmente ocorreu a implementação de bibliotecas para padronização e formatação do código fonte, junto com algumas outras para que o sistema tenha suporte a múltiplas linguagens e para facilitar a estilização de telas. Tendo o ambiente configurado o foco foi na implementação do Firebase, para que a partir disto as telas que necessitassem da manipulação de informações de um banco de dados pudessem ser construídas.

Na segunda *sprint* várias telas essenciais foram desenvolvidas, como o cadastro e detalhes de serviço a tela de perfil do usuário e várias outras. Também foram criados alguns componentes que seriam utilizados nelas, como o selecionador de data e hora, por exemplo.

A terceira *sprint* teve foco em arrumar detalhes das telas já criadas, já que foram identificados pontos a melhorar e foram criadas as telas de manutenção da conta de usuário, como a alteração de nome, *e-mail* e a troca de senha, que são essenciais em um sistema.

A quarta *sprint* reuniu as atividades necessárias para a publicação do app na Play Store, refatorações e melhorias para publicar e mudanças nas configurações do banco de dados. Foi nesta *Sprint* que o app foi publicado e os usuários ganharam a possibilidade de baixa-lo gratuitamente.

Na quinta e penúltima *sprint* várias telas de edição de dados foram feitas, para permitir alterações nos dados. Também foram adicionados novos campos no cadastro de cliente, como os campos de endereço e data de aniversário e novas *flags* que indicam algum estado de um documento, como a *flag* que indica que um serviço teve todos os processos de marketing encerrados.

A sexta e última *sprint* concentrou funcionalidades muito importantes para a aplicação, como o envio de mensagens de aniversário para clientes e a possibilidade de adicionar macros nas mensagens das etapas de pós marketing e nas mensagens de aniversário. Estas alterações permitem que sejam criadas mensagens específicas para cada cliente ou que sejam adicionados dados em uma mensagem quando ela for enviada, como a data atual, por exemplo.

4.1.1 Entrega 1

No dia 17 de fevereiro de 2020 o grupo iniciou o planejamento da primeira *sprint (Sprint Planning Meeting),* que possui 21 dias de duração (prazo máximo 19 de março de 2020). Nesta reunião as atividades foram definidas e um nível de dificuldade foi estipulada para cada uma delas. No geral essas atividades estão relacionadas à criação e configuração do projeto, a integração com os serviços do Firebase, junto com o desenvolvimento de algumas telas e componentes essenciais. A Tabela 26 abaixo apresenta detalhadamente as atividades, seu tempo de realização em dias e sua respectiva pontuação.

Tabela 26 - Planejamento realizado para primeira entrega.

Atividade	**Tempo (em dias)**	**Pontos**
CRIAÇÃO DO PROJETO: Criar projeto.	0,2	1
CRIAÇÃO DO PROJETO: Criar controle de versão Git.	0,2	1
CRIAÇÃO DO PROJETO: Criar projeto no GitHub.	0,2	1

CRIAÇÃO DO PROJETO: Fazer um *push* inicial do repositório para o GitHub.	0,2	1
CRIAÇÃO DO PROJETO: Criar a pasta SRC e dentro colocar: *pages, components, assets, styles, store, hooks, helpers, config.*	0,2	1
CRIAÇÃO DO PROJETO: Instalar React Navigation, Datepicker, AsyncStorage e demais pacotes essenciais.	0,2	1
CRIAÇÃO DO PROJETO: Configurar *react-native-gesture-handler* e *datepicker* no projeto Android (configurar os arquivos *gradle* e *java*).	0,2	1
CRIAÇÃO DO PROJETO: Configurar colors e styles (tanto nos arquivos .xml quanto nos .js).	0,2	1
CRIAÇÃO DO PROJETO: Configurar o Reactotron.	0,2	1
CRIAÇÃO DO PROJETO: Configurar o Redux.	0,2	1
CONFIGURAR PROJETO: Configuração Babel.	0,2	1
CONFIGURAR PROJETO: Configuração Prettier.	0,2	1
CONFIGURAR PROJETO: Configurar Eslint.	0,2	1
DESIGN: Criar logo.	0,2	1
CONFIGURAR i18next: Criar a instância configurada do i18next.	0,2	1
CONFIGURAR i18next: Criar arquivos de linguagem e importar na configuração do i18next.	0,2	1
CONFIGURAR i18next: Carregar os *locales* na página *App/index.js*.	0,2	1
SPLASHSCREEN: Criar um componente de *splashscreen.*	0,2	1
SPLASHSCREEN: Colocar a *window background* do Android como uma imagem e usar o logo da aplicação nela.	0,2	1

CONFIGURAÇÃO DAS ROTAS: *DrawerNavigator.*	0,3	2
CONFIGURAÇÃO DAS ROTAS: Criar o *header* do *drawer.*	0,3	2
CONFIGURAÇÃO DAS ROTAS: *MainNavigator.*	0,2	1
CONFIGURAÇÃO DAS ROTAS: *SwitchNavigator.*	0,2	1
ADD FIREBASE AO PROJETO: Criar projeto Firebase.	0,4	3
ADD FIREBASE AO PROJETO: Instalar a biblioteca react-native-firebase.	0,3	2
ADD FIREBASE AO PROJETO: Adicionar a dependência do Firestore ao projeto.	0,3	2
ADD FIREBASE AO PROJETO: Configurar a biblioteca para conectar com o projeto criado no site do Firebase.	0,3	2
ADD FIREBASE AO PROJETO: Descobrir como obter o usuário logado na aplicação.	0,2	1
ADD FIREBASE AO PROJETO: Fazer a autenticação do usuário (*login*).	0,3	2
ADD FIREBASE AO PROJETO: Criar a *collection* de *customers.*	0,2	1
ADD FIREBASE AO PROJETO: Cadastrar um usuário manualmente no console do Firebase.	0,2	1
COMPONENTES ESSENCIAIS: *InputError.*	0,2	1
COMPONENTES ESSENCIAIS: FAB (*Floating Action Button*)	0,2	1
COMPONENTES ESSENCIAIS: Icon com tamanho e cor padrão para utilizar mais facilmente em todo o app.	0,2	1
COMPONENTES ESSENCIAIS: Instalar o *TextInputMask* no app e fazer a estilização padrão com o *styled-components*.	0,2	1

COMPONENTES ESSENCIAIS: *SimpleListItemText.*	0,2	1
COMPONENTES ESSENCIAIS: *SimpleListItem.*	0,2	1
COMPONENTES ESSENCIAIS: *RefreshControl* com cores padrão se ndo a *accent color.*	0,2	1
COMPONENTES ESSENCIAIS: *MessagePanel.*	0,2	1
COMPONENTES ESSENCIAIS: *TouchableIcon.*	0,2	1
COMPONENTES ESSENCIAIS: *NavigationHeader.*	0,3	2
COMPONENTES ESSENCIAIS: *Spinner.*	0,2	1
COMPONENTES ESSENCIAIS: *Input.*	0,4	3
COMPONENTES ESSENCIAIS: *Label.*	0,2	1
COMPONENTES ESSENCIAIS: *Button.*	0,2	1
COMPONENTES ESSENCIAIS: *Touchable* (.*android* e .ios).	0,2	1
LOGIN: Instalar *styled-components.*	0,2	1
LOGIN: Criar *label* dos *inputs.*	0,2	1
LOGIN: Mostrar mensagem de erro se o *login* falhar.	0,2	1
LOGIN: Redirecionar o usuário para a *home* se o *login* for efetuado co m sucesso.	0,2	1
LOGIN: Fazer o *login* - Consultar no banco de dados.	0,2	1

LOGIN: Criar *action* do *input* de senha para mostrar a senha.	0,3	2
LOGIN: Criar *card* que contém o botão e os *inputs.*	0,2	1
LOGIN: Criar o *input* de *e-mail* e senha.	0,2	1
LOGIN: Criar botão de *login.*	0,2	1
LOGIN: Colocar imagem de *background.*	0,2	1
LOGIN: Se clicar no botão *login* sem preencher alguma informação da r foco no campo que está vazio.	0,2	1
CADASTRO DE CLIENTES: *E-mail*	0,2	1
CADASTRO DE CLIENTES: Celular/Whatsapp (usar máscara).	0,2	1
CADASTRO DE CLIENTES: Telefone fixo (usar máscara).	0,2	1
CADASTRO DE CLIENTES: Nome completo.	0,2	1
LISTAGEM DE CLIENTES: Criar *hook* para filtrar itens em um array p or *keys* presentes nos objetos (itens) desse *array.*	0,5	4
LISTAGEM DE CLIENTES: Criar *hook* de *callback* quando o usuário p ara de digitar.	0,3	2
LISTAGEM DE CLIENTES: Campo de digitação para pesquisa.	0,2	1
LISTAGEM DE CLIENTES: FAB com o ícone *plus* para abrir a tela de cadastro de clientes.	0,2	1
LISTAGEM DE CLIENTES: Quando tocar em um item da lista (um clie nte) abrir a tela de detalhes de cliente enviando o ID do cliente como p arâmetro.	0,3	2
LISTAGEM DE CLIENTES: Mostrar mensagem quando não tiver nenh um cliente cadastrado.	0,2	1

CADASTRO DE TIPOS DE SERVIÇO: Criar botão adicionar serviço.	0,2	1
CADASTRO DE TIPOS DE SERVIÇO: Criar campo de nome do tipo d e serviço e validar se está vazio para salvar o tipo do serviço.	0,2	1
LISTAGEM DE TIPOS DE SERVIÇO: Cada item da lista deve ter um b otão para excluir esse tipo de serviço.	0,2	1
LISTAGEM DE TIPOS DE SERVIÇO: Criar a lista que mostra o nome do tipo de serviço e a data de cadastro.	0,2	1
LISTAGEM DE TIPOS DE SERVIÇO: Fazer campo de pesquisa por n ome e data de cadastro que fica em cima da lista.	0,2	1
LISTAGEM DE TIPOS DE SERVIÇO: Criar FAB que navega para a tel a de cadastro.	0,2	1
DETALHES CLIENTE: Mostrar os dados do cliente.	0,2	1
DETALHES CLIENTE: Criar opção de excluir o cliente.	0,2	1
CADASTRO DE ETAPA DE PÓS-MARKETING: Nome da etapa.	0,2	1
CADASTRO DE ETAPA DE PÓS-MARKETING: Observações da etapa.	0,3	2
CADASTRO DE ETAPA DE PÓS-MARKETING: Mensagem para enviar por *e-mail.*	0,3	2
CADASTRO DE ETAPA DE PÓS-MARKETING: Mensagem para enviar pelo whatsapp.	0,3	2
CADASTRO DE ETAPA DE PÓS-MARKETING: Quantidade de dias para que o serviço entre nesta etap a.	0,3	2
LISTAGEM DE ETAPAS DE PÓS MARKETING: Ao tocar um item abri r a página de detalhes da etapa.	0,2	1
LISTAGEM DE ETAPAS DE PÓS MARKETING: Criar FAB para ir par a o cadastro de etapas.	0,2	1
LISTAGEM DE ETAPAS DE PÓS-MARKETING: Criar lista que mostra o nome, a quantidade de dias e as observações.	0,2	1

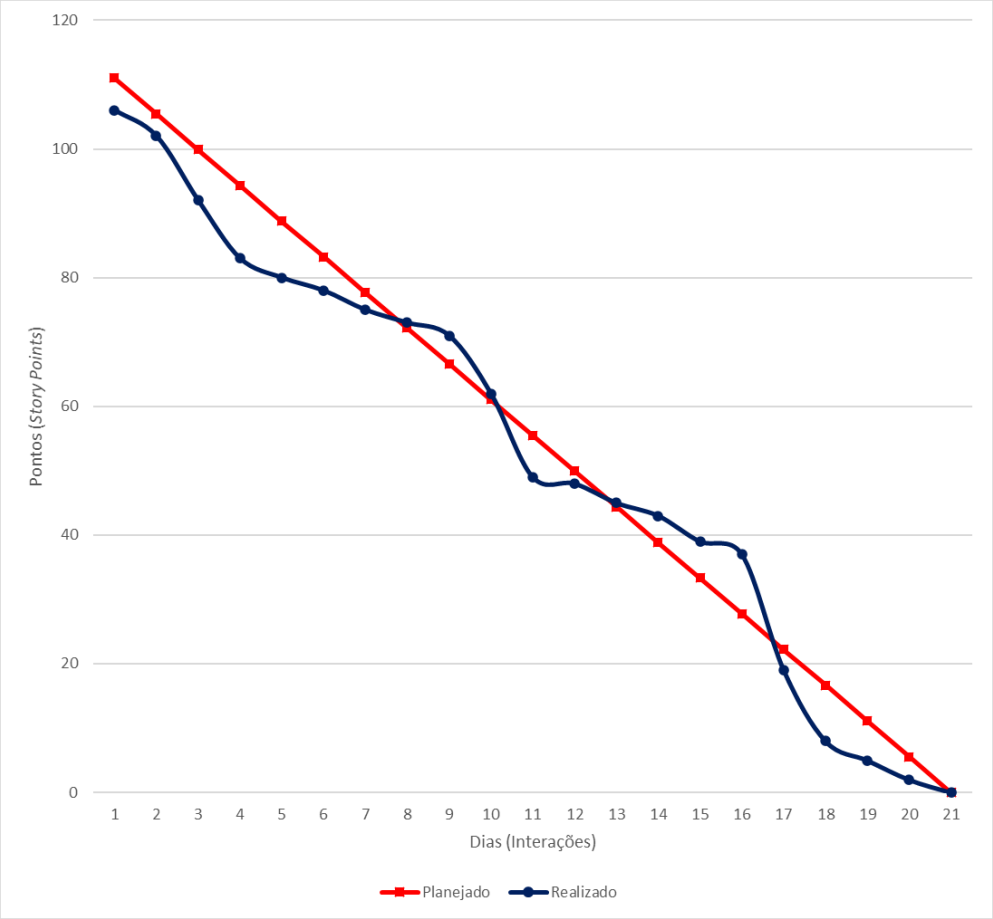
LISTAGEM DE ETAPAS DE PÓS-MARKETING: Criar campo de pesquisa.	0,2	1
DETALHES DE ETAPA DE PÓS-MARKETING: Mostrar os dados da etapa.	0,2	1
DETALHES DE ETAPA DE PÓS-MARKETING: Criar botão para excluir etapa.	0,2	1
PERFIL DO USUÁRIO: Subscrever a mudanças do documento do usuário dentro da coleção de usuários para pegar os dados do usuário logado e gravar na memória (usando *Redux*).	1,7	4
Total	**21**	**111**

Fonte: Elaborado pelos autores (2020).

Ao decorrer dos 21 dias de desenvolvimento, foram realizadas baixas na pontuação conforme as atividades eram concluídas. A Figura 13 apresenta o gráfico de Burndown da primeira entrega, que destaca o que foi planejado e o que foi realizado.

Com estas informações é notável o bom desempenho obtido quanto ao desenvolvimento do aplicativo e as atividades relacionadas ao mesmo. Não houveram atrasos nem problemas graves que prejudicassem a realização das tarefas planejadas. A comunicação sempre foi amigável e por isto não houveram desentendimentos ou conflitos.

Figura 13 - Gráfico de *Burndown* da entrega 1.



Fonte: Elaborado pelos autores (2020).

No dia 18 de março de 2020 ocorreu a reunião para a realização da revisão do entregável. Nesta reunião todos os membros estavam presentes. Durante a reunião, a equipe fez uma autoavaliação procurando detectar os principais pontos de acertos e erros, e juntos, propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

- O que deu certo: Houve uma boa comunicação entre os integrantes. Todos estavam cientes das dificuldades enfrentadas. Além disso o desenvolvimento do aplicativo ocorreu como planejado e a organização dos cartões do Trello foram coisas que funcionaram perfeitamente;
- O que deu errado: Houve um pequeno mal-entendido dos integrantes quanto a regra de negócio e o funcionamento inicial da aplicação, gerando assim um retrabalho no desenvolvimento dos diagramas de caso de uso.

Houve também pequenos problemas com a utilização dos recursos quanto os arquivos de documentação;

- Ações de Melhorias: Reunir o grupo remotamente mais vezes por WhatsApp, Discord ou Skype para alinhar as tarefas, o que cada um está fazendo e se estão encontrando alguma dificuldade (e se caso estejam, realizar o desbloqueio deste membro do time), facilitar a utilização da documentação em nuvem para e edição simultânea da mesma.

4.1.2 Entrega 2

No dia 14 de maio de 2020 o grupo se reuniu e realizou o planejamento da segunda *sprint*. Nesta reunião o grupo selecionou atividades mais complexas, que estão diretamente ligadas com a função principal do aplicativo, uma vez que a estrutura do projeto já foi desenvolvida na primeira *sprint*. O grupo notou que após a estrutura do projeto ter sido criada, o processo de desenvolvimento se tornou semelhante a montagem de um quebra cabeça, no qual as peças já estão prontas e só é necessário encaixá-las nos lugares adequados.

Tabela 27 - Planejamento realizado para segunda entrega.

Atividade	**Tempo (em dias)**	**Pontos**
RETORNO DO CLIENTE: Criar campo de motivo do retorno.	1	10
RETORNO DO CLIENTE: Criar campo de hora do retorno (sugerir hora atual).	1	2
RETORNO DO CLIENTE: Criar campo de data do retorno (sugerir dia atual).	1	2
RETORNO DO CLIENTE: Criar campo de observações.	1	2
RETORNO DO CLIENTE: Criar botão para abrir a tela de cadastro de motivo de retorno.	1	1
DETALHES DO SERVIÇO: Botão para cadastrar o retorno do cliente ao estabelecimento.	1	1
DETALHES DO SERVIÇO: Botão para editar e excluir a avaliação.	1	2

DETALHES DO SERVIÇO: Botão para cadastrar avaliação.	1	1
DETALHES DO SERVIÇO: Mostrar o cliente	1	1
DETALHES DO SERVIÇO: Mostrar datas de inclusão, data do serviço e as outras informações.	1	1
LISTAGEM DE SERVIÇOS: Mostrar o nome do cliente, nome do tipo de serviço, data de início e data de término do serviço.	1	1
LISTAGEM DE SERVIÇOS: Criar campo de pesquisa.	1	1
PERFIL DO USUÁRIO: Criar a página de perfil do usuário para mostrar seus dados.	1	4
PERFIL DO USUÁRIO: Criar botão para sair da conta.	1	2
COMPONENTES ESSENCIAIS: Criar *modal* que exibe uma lista grande, que pode ter paginação e tem pesquisa.	1	10
CADASTRO DE SERVIÇO: Escolher data de finalização do serviço.	1	2
CADASTRO DE SERVIÇO: Criar campo para escolher a data início do serviço.	1	2
CADASTRO DE SERVIÇO: Selecionar cliente (pode ser um *modal* ou abrir uma outra tela).	1	5
COMPONENTES ESSENCIAIS: *Select* (um componente genérico para selecionar algo e mostrar um texto simples).	1	4
COMPONENTES ESSENCIAIS: Criar *datetimepicker* usando o componente da comunidade do React Native.	1	5
COMPONENTES ESSENCIAIS: Um componente para mostrar a avaliação do usuário (componente que mostra a quantidade de estrelas selecionadas).	1	2
Total	**21**	**61**

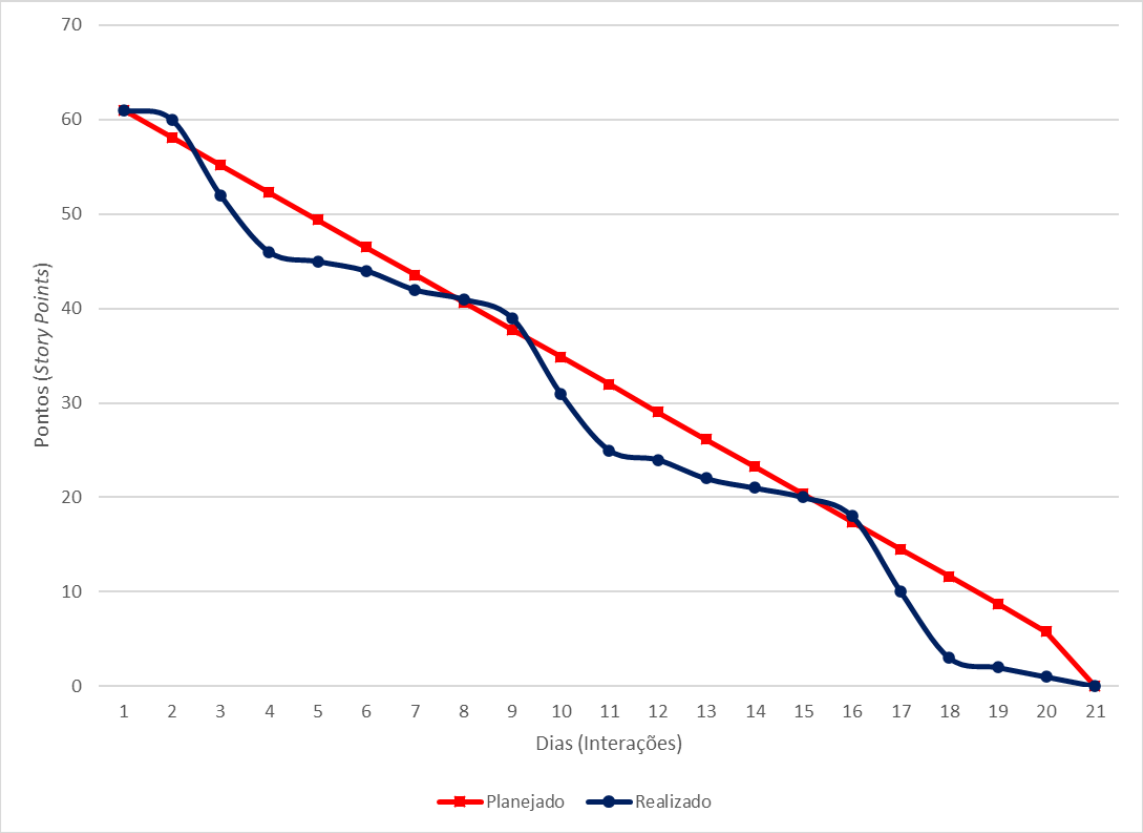
Fonte: Elaborado pelos autores (2020).

Ao decorrer da segunda *sprint* de desenvolvimento a equipe realizou todas as atividades planejadas estando em dia com o cronograma. A Figura 14 abaixo apresenta o gráfico de *Burndown* da segunda entrega, destacando o planejamento de baixas nos pontos e a baixas realizadas pela equipe.

Não houveram atrasos nem problemas graves que prejudicassem a realização das tarefas planejadas. Toda a equipe manteve uma boa comunicação

e sempre trabalhou nos dias que podiam para adiantar o desenvolvimento, principalmente nos finais de semana, nos quais foram feitas mais baixas.

Figura 14 - Gráfico de *Burndown* da entrega 2.



Fonte: Elaborado pelos autores (2020).

No dia 03 de maio de 2020 a equipe inteira se reuniu para realizar a revisão do entregável e durante ela a equipe fez uma autoavaliação procurando detectar os principais pontos de acertos e erros, e juntos, propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

- O que deu certo: Comunicação amigável e frequente, boa organização dos cartões no Trello e um bom planejamento da *sprint*;
- O que deu errado: Problemas de um integrante em seu computador para executar o aplicativo em modo de desenvolvimento;
- Ações de Melhorias: Disponibilizar ao integrante com problemas em seu computador uma forma de ajudar no desenvolvimento do aplicativo.

4.1.3 Entrega 3

No dia 4 de junho de 2020 o grupo se reuniu e realizou o planejamento da terceira *Sprint*. Nesta reunião o grupo selecionou as atividades que são essenciais para a versão mínima e viável do aplicativo, ao ponto que quando finalizadas o aplicativo já fizesse o que ele promete, que é o envio de mensagens de pós-marketing. Além disso outras atividades muito importantes para dar aos usuários uma visão geral da sua atividade foram selecionadas, como as tarefas relacionadas a tela de *Dashboard*. A Tabela 28 abaixo corresponde ao planejamento feito para esta terceira entrega.

Tabela 28 – Planejamento realizado para a terceira entrega.

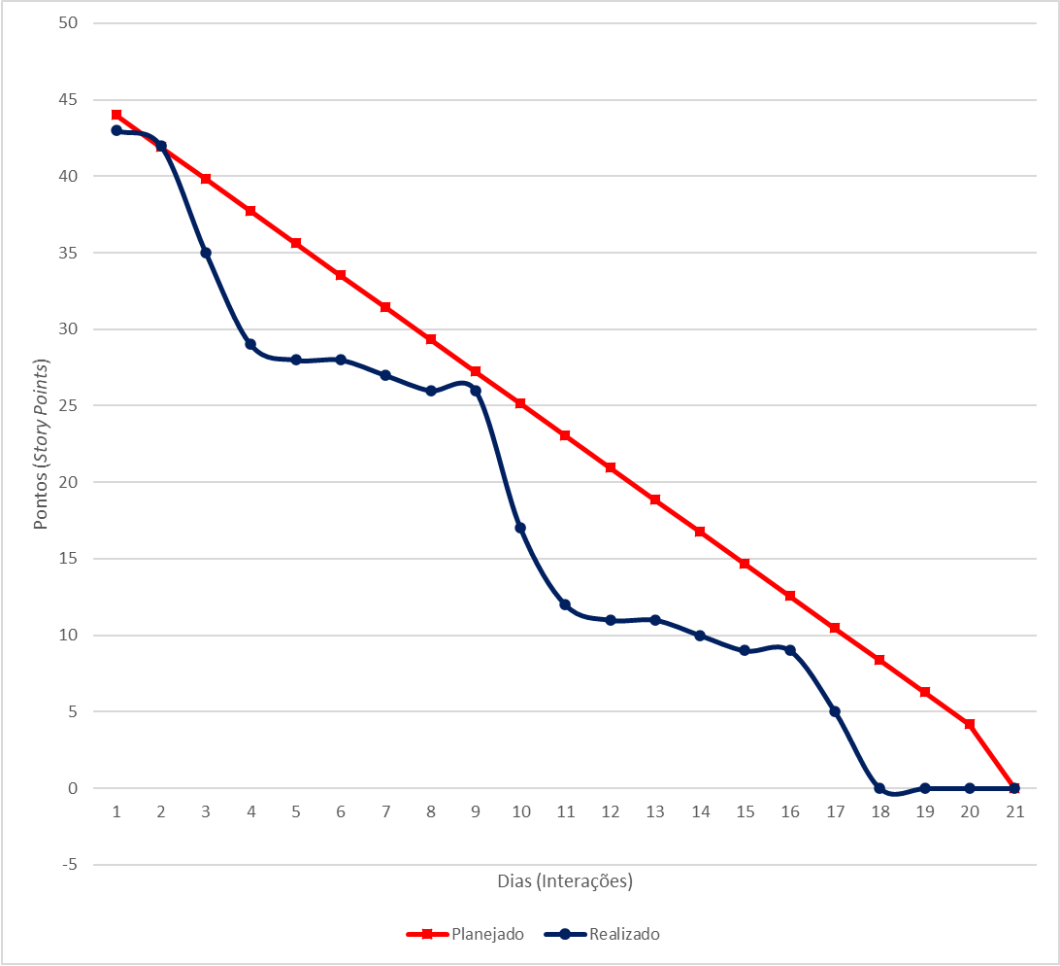
Atividade	**Tempo (em dias)**	**Pontos**
DASHBOARD: Gráfico de volume de serviços: semanal/mensal/anual/todo o período. Com filtro por tipo de serviço.	3	5
DASHBOARD: Aba de visão geral que mostra estatísticas gerais da empresa.	3	5
PERFIL DO USUÁRIO: Criar página para o usuário ver e alterar as informações segurança conta do app (como a sua senha).	1	5
PERFIL DO USUÁRIO: Criar página para o usuário ver seus dados pessoais e conseguir editá-los.	1	2
DASHBOARD: Criar estrutura da página usando abas.	2	5
CADASTRO DE ETAPA DE PÓS-MARKETING: Adicionar campo para salvar mensagem SMS.	1	1
DETALHES DO SERVIÇO: Colorir o botão de enviar mensagem se a mensagem já foi enviada.	1	2
DETALHES DO SERVIÇO: Diminuir a opacidade dos botões se ainda não se passaram a quantidade de tempo da etapa de pós-marketing.	3	2
DETALHES DO SERVIÇO: Abrir o WhatsApp, os apps de *e-mail* e o app de SMS ao clicar em seus respectivos botões na aba de enviar mensagem.	3	10
DETALHES DO SERVIÇO: Mostrar botões para enviar as mensagens para o cliente do serviço.	1	5
HOMEPAGE: Mostrar breves estatísticas do mês atual e colocar botão para abrir o *Dashboard.*	1	1

COMPONENTES ESSENCIAIS: *Checkbox.*	1	1
Total	**21**	**44**

Fonte: Elaborado pelos autores (2020).

Ao decorrer do desenvolvimento da terceira entrega a equipe seguiu fielmente o cronograma para não se perder nos prazos estipulados previamente. A Figura 15 abaixo apresenta o gráfico de *Burndown* da terceira entrega, que irá mostrar assertivamente como foi executado o que foi planejado.

Figura 15 - Gráfico de *Burndown* da terceira entrega.



Fonte: Elaborado pelos autores (2020).

No dia 24 de junho de 2020 a equipe realizou a revisão do entregável e fez uma autoavaliação procurando detectar os principais pontos de acertos e erros para

propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, foi elaborado o seguinte resumo sobre algumas questões:

- O que deu certo: Melhoria da comunicação remota (WhatsApp e Discord) e Organização dos cartões do Trello;
- O que deu errado: Confusão sobre diagramas UML na elaboração da documentação do *software*;
- Ações de Melhorias: Pesquisar bem o que se deseja documentar antes de começar a fazer.

4.1.4 Entrega 4

No dia 4 de julho de 2020 o grupo se reuniu e realizou o planejamento da quarta *Sprint*. Nesta reunião o grupo selecionou atividades necessárias para a publicação do aplicativo na Play Store, como a reformulação do logo, criação dos *screenshots* que ficam na página de detalhes, configuração do *analytics* e *crashlytics*, entre outras. A Tabela 29 corresponde ao planejamento feito para esta entrega.

Tabela 29 - Planejamento realizado para a quarta entrega.

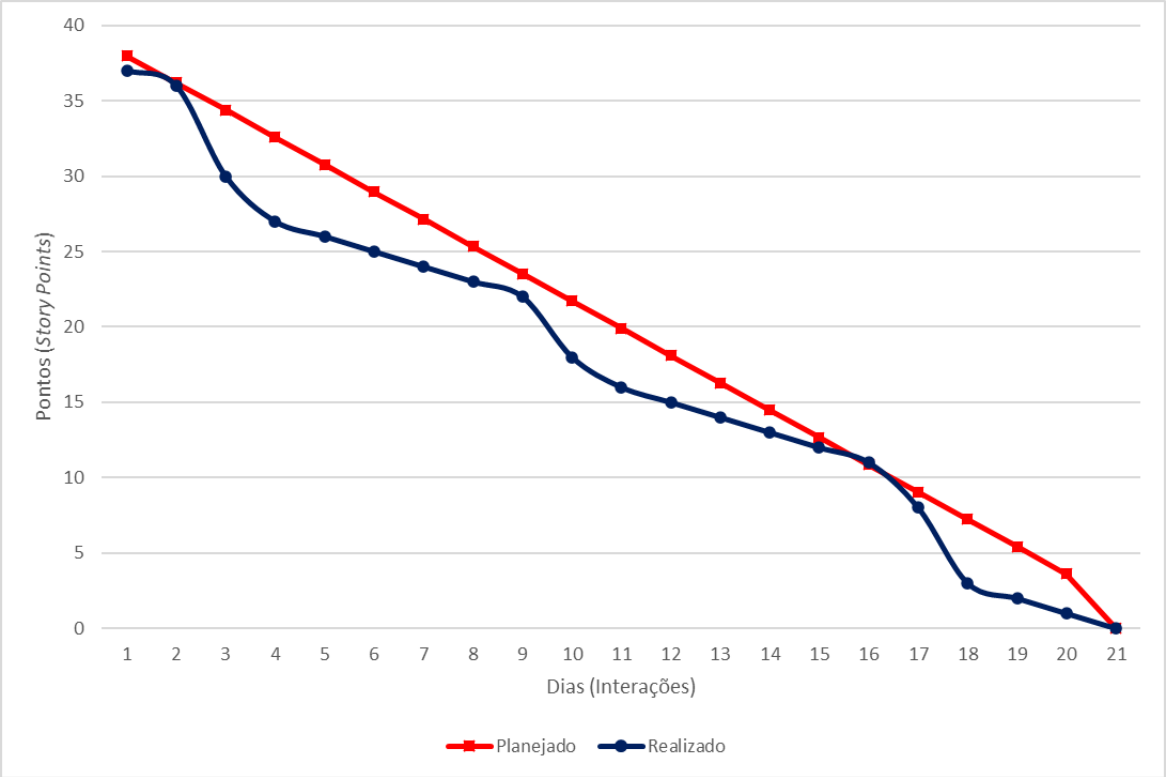
Atividade	**Tempo (em dias)**	**Pontos**
PUBLICAÇÃO: Criar app, imagens, textos da loja e site com política de privacidade.	2	3
FIREBASE: Adicionar regras de leitura e gravação.	1	1
MONETIZAÇÃO: Adicionar anúncios no aplicativo, nas telas de listagem e cadastro.	2	5
REFATORAÇÃO E MELHORIAS: Copiar os dados de tipo de retorno dentro dos documentos de retorno do cliente.	1	1
REFATORAÇÃO E MELHORIAS: Colocar todas as *collections* menos a de *users* dentro da *collection* de *companies.*	1	1
REFATORAÇÃO E MELHORIAS: Copiar os dados de cliente e tipo de serviço nos documentos de serviço.	1	2
CADASTRO DE USUÁRIOS: Criar cadastro de conta e de empresa para efetuar *login.*	3	15
LOGIN: Esqueci minha senha.	1	1
REFATORAÇÃO E MELHORIAS: Esconder *Dashboard* e estatísticas da *Home.*	1	1
VISUAL: Alterar logo do app Android na *splashscreen*, ícone no *androidManifest* e na pasta *assets.*	1	1
REFATORAÇÃO E MELHORIAS: Separar os APKs por arquitetura e habilitar o Proguard.	1	1

VISUAL: Criar *screenshots* da Play Store.	1	1
VISUAL: Criar novo logo.	1	1
VISUAL: Criar *banner* Play Store.	1	1
FIREBASE: Arrumar *templates* de *e-mail* para redefinição de senha, *e-mail* e outros.	1	1
FIREBASE: *Setup* Firebase *crashlytics.*	1	1
FIREBASE: *Setup* Firebase *analytics.*	1	1
Total	**21**	**38**

Fonte: Elaborado pelos autores (2020).

Ao decorrer do desenvolvimento da quarta entrega a equipe seguiu fielmente o cronograma para não se perder nos prazos definidos. A Figura 16 abaixo apresenta o gráfico de *Burndown* da quarta entrega, que irá mostrar claramente como foi executado o que foi planejado.

Figura 16 - Gráfico de *Burndown* da quarta entrega.



Fonte: Elaborado pelos autores (2020).

No dia 24 de julho de 2020 a equipe realizou a revisão desta entrega e fez uma autoavaliação procurando detectar os principais pontos de acertos e erros para propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, foi elaborado o seguinte resumo sobre algumas questões:

- O que deu certo: A publicação do aplicativo foi um sucesso;
- O que deu errado: Por causa do tempo sem mexer no projeto alguns detalhes tiveram que ser alinhados novamente;
- Ações de Melhorias: Não ficar longos tempos sem mexer no projeto.

4.1.5 Entrega 5

No dia 27 de julho de 2020 o grupo se reuniu e realizou o planejamento da quinta *Sprint*. Nesta reunião o grupo selecionou várias atividades de refatoração, criação de novos campos em cadastros e a criação de telas de edições de dados para melhorar ainda mais o produto. A Tabela 30 corresponde ao planejamento feito para esta entrega.

Tabela 30 - Planejamento realizado para a quinta entrega.

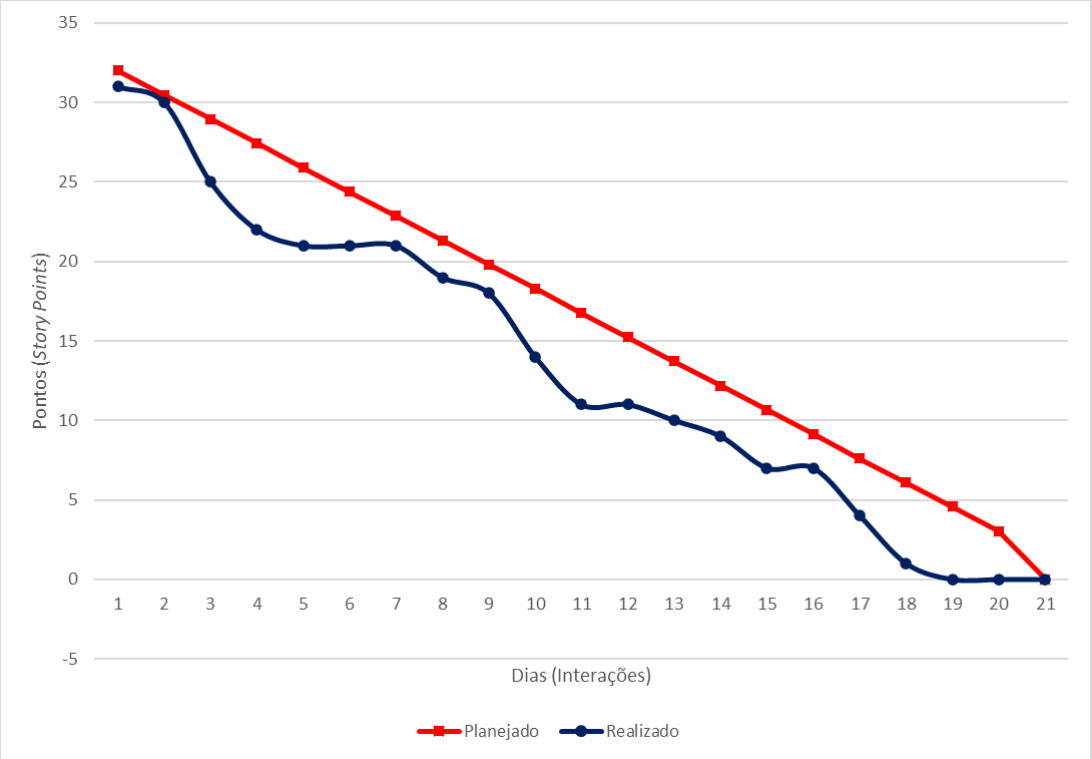
Atividade	**Tempo (em dias)**	**Pontos**
EDIÇÕES DE DADOS: editar dados do cliente.	2	5
CADASTRO DE CLIENTES: Mostrar a *flag* que enviar mensagem e os dados do endereço na página de detalhes.	1	1
CADASTRO DE CLIENTES: Usar *flag* do cliente na hora de mandar mensagem de pós-marketing.	1	1
CADASTRO DE CLIENTES: Consumir API de busca de CEP.	2	3
CADASTRO DE CLIENTES: Endereço do cliente. CEP, endereço, bairro, número, cidade, estado.	2	4
CADASTRO DE CLIENTES: *Checkbox* para marcar se o usuário deseja receber mensagens de pós-marketing.	1	1
CADASTRO DE CLIENTES: Escolher Data de nascimento.	1	1
ABA ENVIO DE MENSAGENS: Desmarcar canal como enviado na aba de enviar mensagens do serviço.	2	3

SERVIÇO: Excluir serviço. Quando fizer isso excluir os retornos de cliente ou mover eles para o documento de serviço, aí não precisa excluir eles.	2	3
SERVIÇO: Marcar serviço que os processos de *marketing* acabaram (bloquear todas modificações).	1	1
EDIÇÕES DE DADOS: editar dados de etapa de *marketing*.	1	1
REFATORAÇÃO E MELHORIAS: Resolver o erro de espaçamento do componente *Input.*	1	1
MOTIVO RETORNO: Listagem de motivos de retorno no *drawer* com botão para excluir. Não fazer a edição.	1	1
REFATORAÇÃO E MELHORIAS: Arrumar as datas que usam o objeto *timestamp* do Firebase.	1	1
EDIÇÕES DE DADOS: editar serviço.	2	5
Total	**21**	**32**

Fonte: Elaborado pelos autores (2020).

Ao decorrer do desenvolvimento da quinta entrega a equipe seguiu fielmente o cronograma para não se perder nos prazos definidos. A Figura 17 abaixo apresenta o gráfico de *Burndown* da quinta entrega, que irá mostrar claramente como foi executado o que foi planejado.

Figura 17 - Gráfico de *Burndown* da quinta entrega.



Fonte: Elaborado pelos autores (2020).

No dia 16 de agosto de 2020 a equipe realizou a revisão desta entrega e fez uma autoavaliação procurando detectar os principais pontos de acertos e erros para propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, foi elaborado o seguinte resumo sobre algumas questões:

- O que deu certo: As melhorias foram publicadas em uma atualização do app com sucesso;
- O que deu errado: Não conseguimos identificar nenhum problema nesta entrega, então nada deu errado.

4.1.6 Entrega 6

No dia 17 de agosto de 2020 o grupo se reuniu e realizou o planejamento da sexta *sprint*. Nesta reunião o grupo selecionou várias atividades muito importantes para o envio de mensagens de pós-marketing do app, como o envio de mensagens de aniversário para clientes e a possibilidade de adicionar macros nas mensagens a serem enviadas, além de uma tela que mostra os dados da empresa do usuário

logado e a possibilidade de editar estas informações. A Tabela 31 corresponde ao planejamento feito para esta entrega.

Tabela 31 - Planejamento realizado para a sexta entrega.

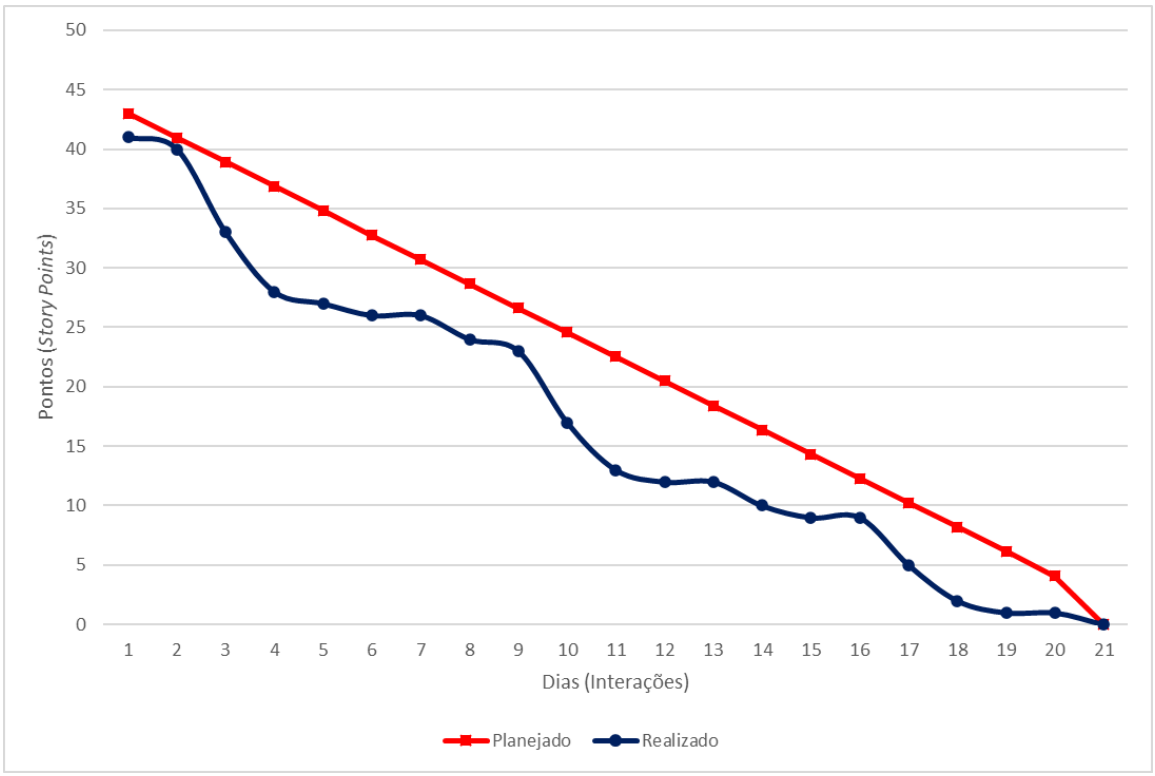
Atividade	**Tempo (em dias)**	**Pontos**
EMPRESA: Perfil da empresa, que mostra os dados da empresa.	1	1
EMPRESA: Editar dados da empresa.	1	1
BUG: Quando exclui em uma tela e ela tem a regra do "*doc.exists*" a mensagem de erro da função *showErrorAlert* é exibida.	1	1
ANIVERSÁRIO CLIENTE: Colocar macro nas mensagens de aniversário.	2	5
CADASTRO DE ETAPA DE PÓS-MARKETING: Pré-visualização de macros para ver como vai ficar a mensagem final enviada para o cliente.	2	5
CADASTRO DE ETAPA DE PÓS-MARKETING: Permitir inserir macros que substituam o macro por alguma informação do serviço ou do cliente.	3	10
CADASTRO DE ETAPA DE PÓS-MARKETING: *builder* de tempo para enviar a mensagem de pós-marketing. Gerar tempo em milissegundos.	3	10
ANIVERSÁRIO CLIENTE: Mostrar quais mensagens já foram enviadas para um cliente.	1	1
ANIVERSÁRIO CLIENTE: Listagem de aniversariantes com filtro de período (data inicial e final). Adicionar filtro de aniversários atrasados (mensagens não enviadas).	2	4
ANIVERSÁRIO CLIENTE: Configuração de mensagens de aniversário.	1	1
TIPO DE SERVIÇO: Descrição do tipo de serviço.	1	1
ABA ENVIO DE MSGS: Abrir detalhes etapa de pós-marketing a partir da aba de enviar mensagens do serviço.	1	1
ABA DE VISÃO GERAL: Botão para abrir os detalhes do cliente.	1	1
DETALHES CLIENTE: mostrar aviso quando entrar nos detalhes do cliente e o cliente não existir mais.	1	1

Fonte: Elaborado pelos autores (2020).

Ao decorrer do desenvolvimento da sexta entrega a equipe seguiu fielmente o cronograma para não se perder nos prazos definidos. A Figura 18 abaixo

apresenta o gráfico de *Burndown* da sexta entrega, que irá mostrar claramente como foi executado o que foi planejado.

Figura 18 - Gráfico de *Burndown* da sexta entrega.



Fonte: Elaborado pelos autores (2020).

No dia 06 de setembro de 2020 a equipe realizou a revisão desta entrega e fez uma autoavaliação procurando detectar os principais pontos de acertos e erros para propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, foi elaborado o seguinte resumo sobre algumas questões:

- O que deu certo: As novas e importantes funcionalidades criadas foram colocadas em produção sem maiores problemas;
- O que deu errado: Não foram identificados problemas nesta *sprint*.

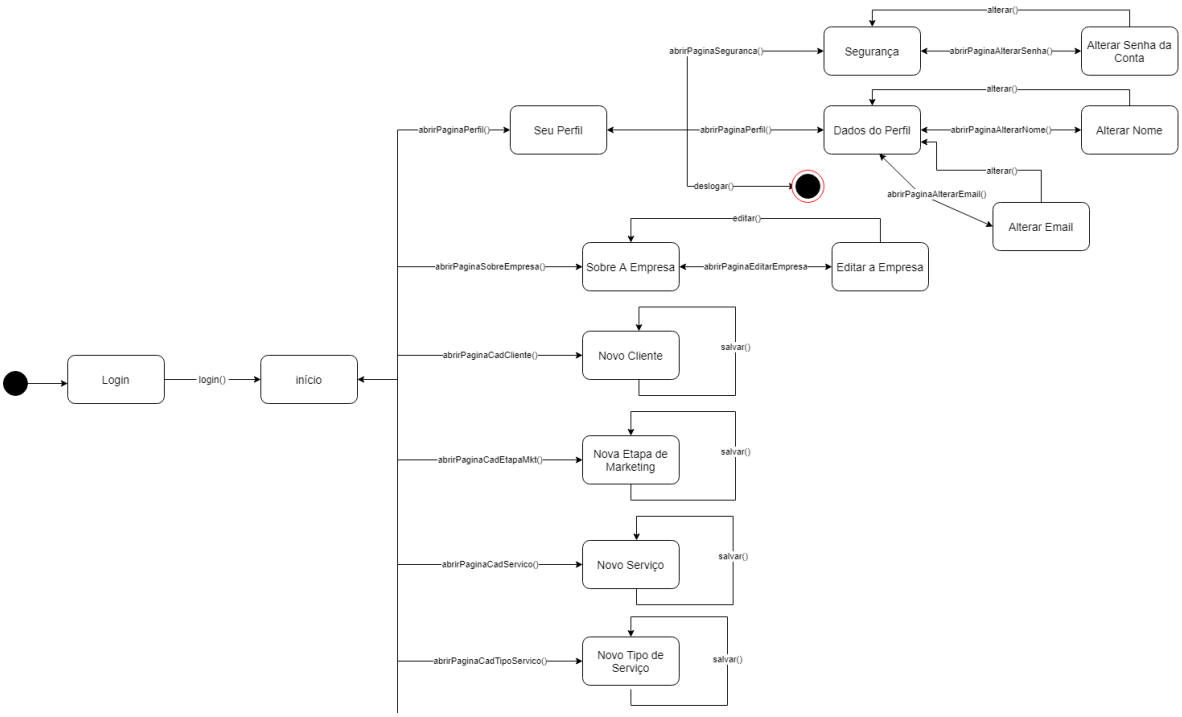
5 TELAS DO APLICATIVO

Este capítulo tem o objetivo de apresentar e explicar cada uma das diferentes telas ou o conjunto de telas que executam determinada ação.

É fundamental que seja criada uma interface amigável ao usuário, pois é ela que determina como as pessoas operam e controlam o sistema. Quando uma interface é bem projetada, ela é facilmente compreensível e controlável.

A Figura 19 corresponde a primeira parte do diagrama de estados, que é uma representação geral de todas as telas presentes no sistema. A figura foi separada em duas partes para ficar visível neste trabalho.

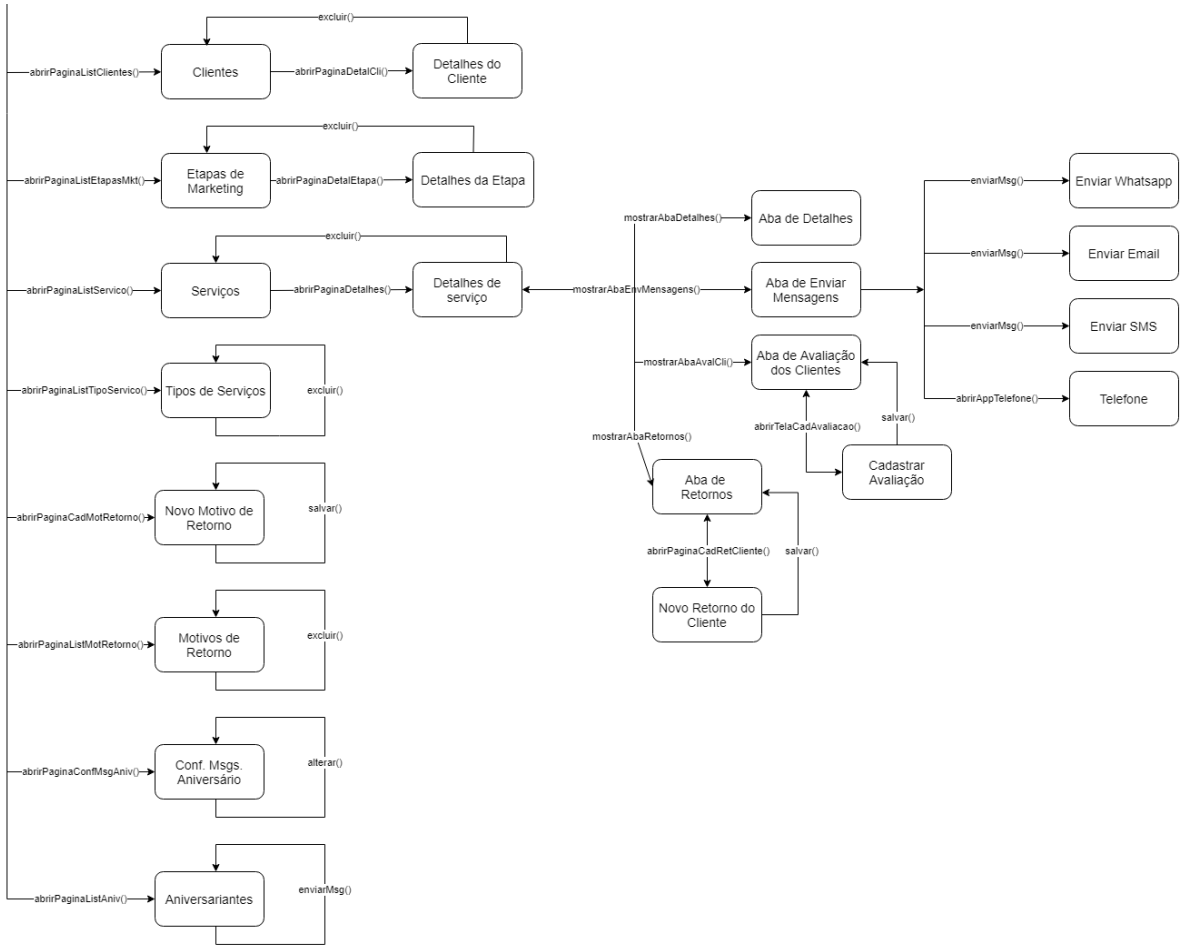
Figura 19 - Diagrama de estados (mapa do sistema) parte 1.



Fonte: Elaborado pelos autores (2020).

A Figura 20 corresponde a segunda parte do diagrama de estados que possui o restante das telas.

Figura 20 - Diagrama de estados (mapa do sistema) parte 2.



Fonte: Elaborado pelos autores (2020).

5.1 Capturas de telas

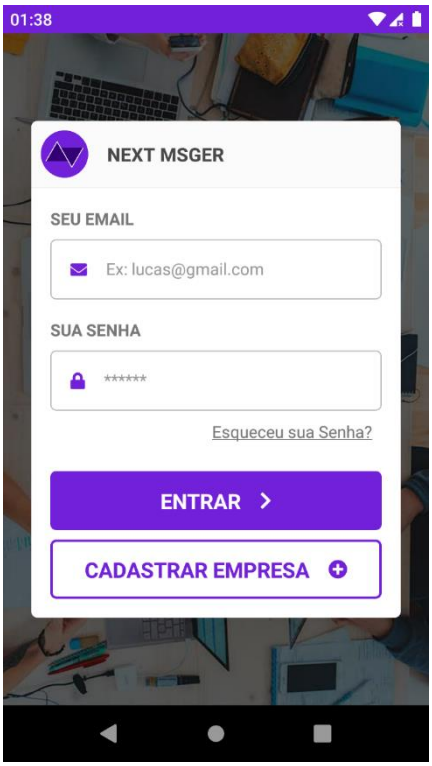
Diversas telas foram construídas no aplicativo e são acessíveis ao usuário. A maioria das telas seguem o mesmo padrão visual e comportamentos semelhantes, desta forma, entendendo uma as outras poderão ser entendidas bem mais facilmente.

As capturas foram tiradas em um emulador do Nexus 5 rodando a versão 9 do Android e usando a compilação de desenvolvimento do aplicativo, por causa disso, os anúncios que são exibidos no app são anúncios de teste disponibilizados pelo Google Admob.

Não serão mostradas muitas telas de edição de dados, já que várias delas são visualmente idênticas as telas de cadastro e só executam uma lógica diferente para ao invés de cadastrar, editar os dados.

A Figura 21 representa a tela de *login* do sistema, que é a porta de entrada para a utilização do app. Nela os usuários colocam suas credenciais de acesso e elas são validadas, caso ele tenha inserido credenciais válidas o *login* é efetuado com sucesso, caso contrário uma mensagem de erro aparecerá. As credenciais necessárias para entrar é o *e-mail* e uma senha. Preenchidas as informações o usuário deve tocar no botão com a cor de fundo roxa visível na tela. Esta tela também possui o botão com borda roxa que leva o usuário até a tela de cadastro de uma nova empresa e o texto tocável sublinhado que leva o usuário até a página de recuperação de conta, caso ele tenha esquecido a sua senha.

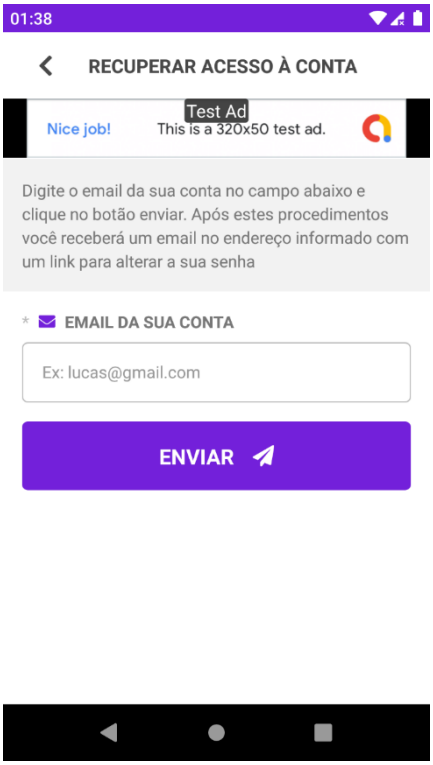
Figura 21 - Tela de *login* do aplicativo.



Fonte: Elaborado pelos autores (2020).

A Figura 22 representa a tela de recuperação de acesso à conta. Nela o usuário coloca o *e-mail* da conta da qual ele não lembra a senha e uma mensagem será enviada para o *e-mail* fornecido para que ele possa redefinir a senha de acesso. Além disso, esta tela exibe anúncios no formato de banner e uma mensagem na sua parte superior.

Figura 22 - Tela de recuperação de acesso à conta.

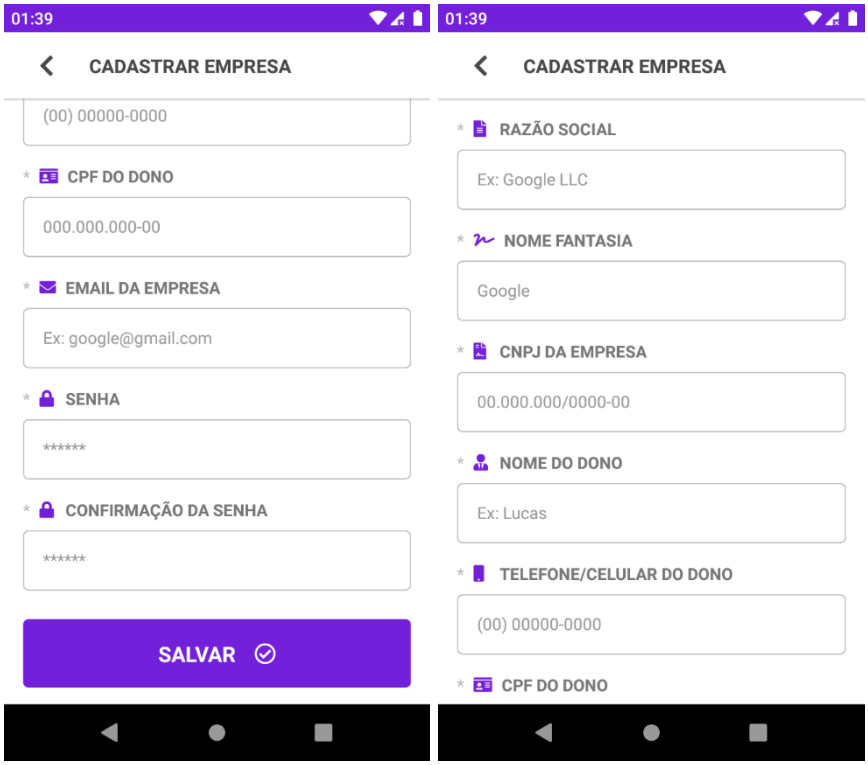


Fonte: Elaborado pelos autores (2020).

A Figura 23 representa a tela de cadastro de empresa do aplicativo. Para todo e qualquer usuário conseguir entrar no sistema ele precisa estar vinculado a uma empresa, já que os usuários pertencem as empresas. Esta tela exige alguns dados da empresa, como razão social, nome fantasia e outros. Alguns dos campos do cadastro, como o de CNPJ, possuem máscara de digitação para facilitar o seu preenchimento. Ao fim da tela existe o botão que invoca a ação de salvar a empresa. Se a empresa for cadastrada com sucesso o usuário será logado no sistema.

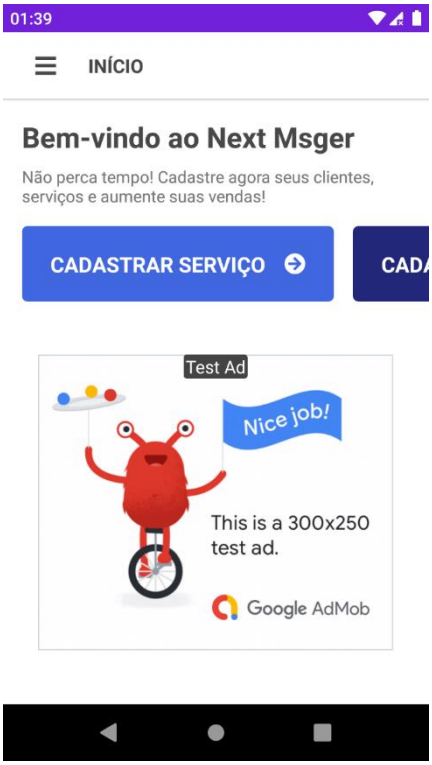
A Figura 24 representa a tela inicial do aplicativo após o *login* ter sido efetuado com sucesso na aplicação. Ela mostra uns botões de acesso rápido para as principais ações de inserção de dados da aplicação e um anúncio centralizado no formato de *banner*.

Figura 23 - Tela de cadastrar empresa.



Fonte: Elaborado pelos autores (2020).

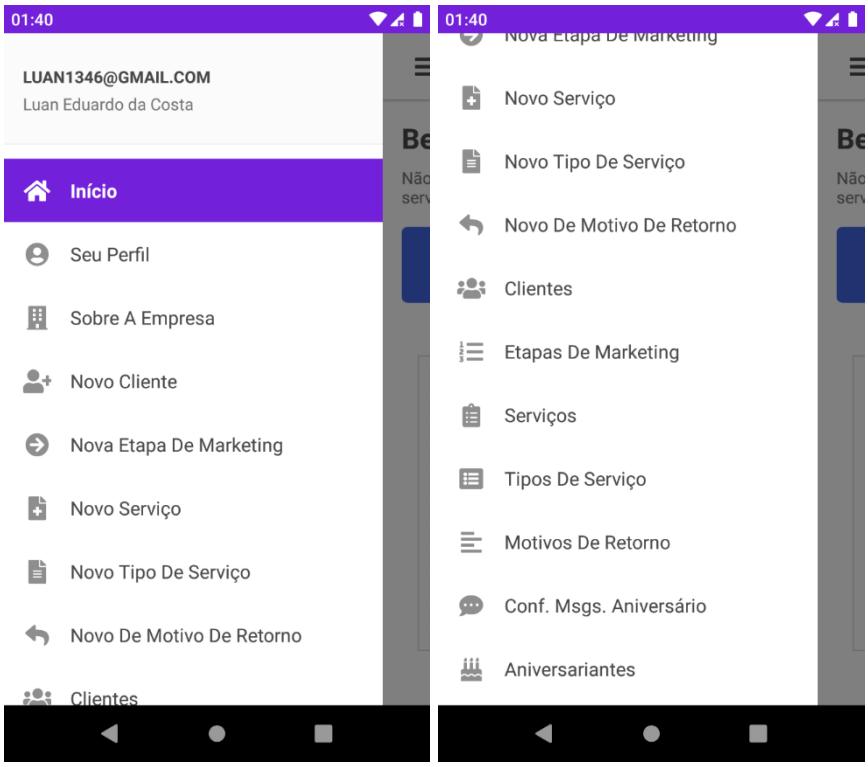
Figura 24 - Tela inicial do aplicativo quando o usuário está logado.



Fonte: Elaborado pelos autores (2020).

A Figura 25 mostra o menu lateral de navegação, também chamado de *drawer menu*. Ele é responsável por facilitar a navegação do usuário entre as principais telas do sistema.

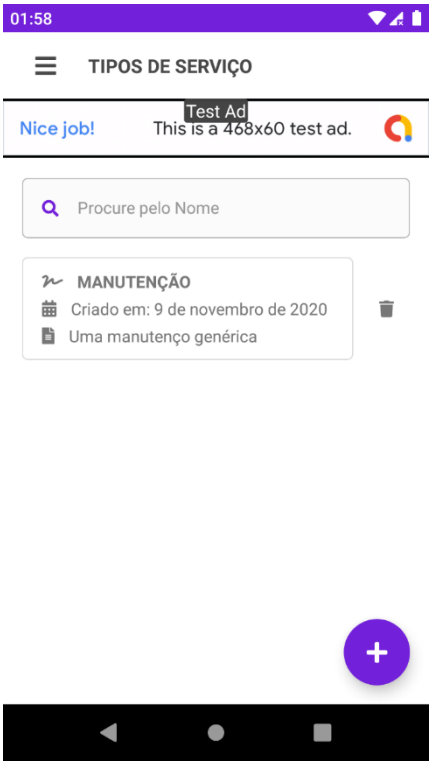
Figura 25 - Menu lateral de navegação do aplicativo.



Fonte: Elaborado pelos autores (2020).

A Figura 26 representa a listagem de tipos de serviços. Ela possui um campo para pesquisa na listagem que busca pelo nome do tipo de serviço. A tela possui um botão flutuante com o ícone de soma, esse botão leva o usuário para a tela de inserir um novo tipo de serviço. Já o ícone de lixeira é outro botão que exclui o tipo de serviço do banco de dados, mas antes de excluir ele mostra uma caixa de confirmação da operação.

Figura 26 - Tela de listagem de tipos de serviço.

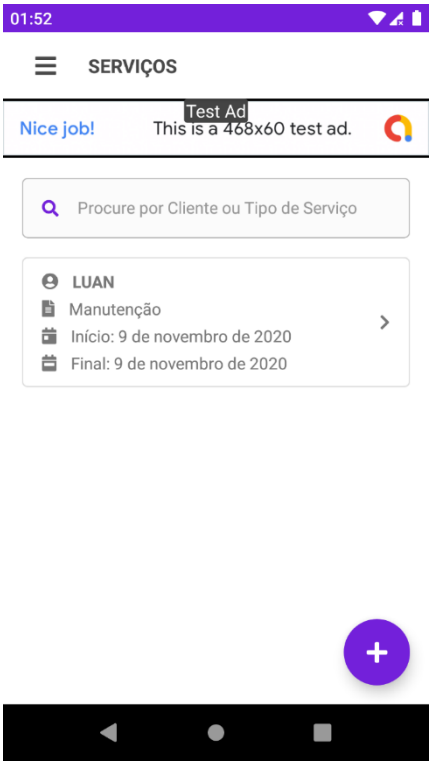


Fonte: Elaborado pelos autores (2020).

A Figura 27 representa a tela listagem de serviços. Ela é muito semelhante as outras listagens da aplicação, o que é bom, já que o usuário se acostuma com o padrão e não demora a aprender a utilizar o *app*. A barra de pesquisa busca na lista pelo nome do cliente e nome do tipo de serviço. Já os itens da lista mostram o nome do cliente, tipo do serviço, data de início do serviço e data de finalização, respectivamente. O botão flutuante para a navegação para a tela de cadastro de serviço está presente no canto inferior direito e os itens respondem ao evento de toque na tela, levando para a tela de detalhes do serviço selecionado.

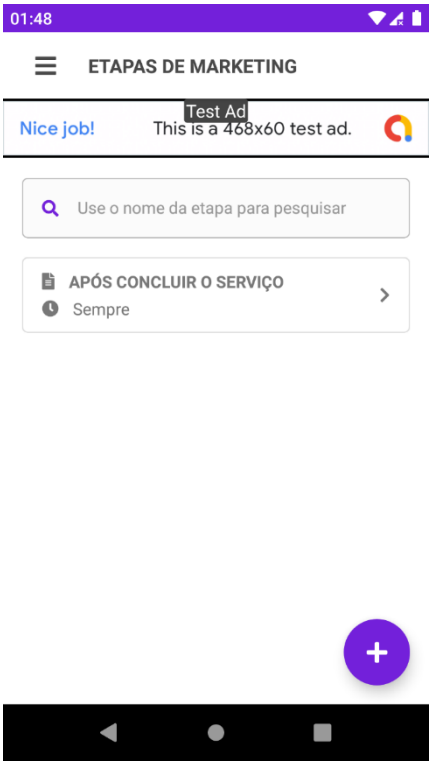
A Figura 28 representa a listagem de etapas de *marketing*. A pesquisa é feita pelo nome da etapa. Já os itens exibem o nome, quando que estará habilitado o envio de mensagens da etapa e as observações (se existirem). Novamente a tela possui o botão de ação flutuante na parte inferior que leva o usuário para a tela de cadastro de etapa de *marketing* e os itens são tocáveis, levando o usuário para a tela de detalhes da etapa quando tocados.

Figura 27 - Tela de listagem de serviços.



Fonte: Elaborado pelos autores (2020).

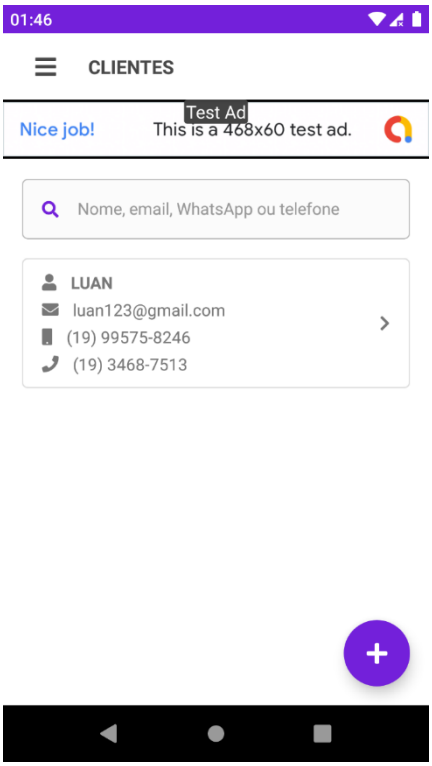
Figura 28 - Tela de listagem de etapas de marketing.



Fonte: Elaborado pelos autores (2020).

A Figura 29 representa a listagem de clientes que segue o mesmo padrão das anteriores. A diferença desta tela é que a pesquisa é feita pelo nome, *e-mail*, WhatsApp e telefone e os itens da lista mostram exatamente esses dados. Além disso os itens também respondem ao toque na tela, levando o usuário aos detalhes do cliente selecionado.

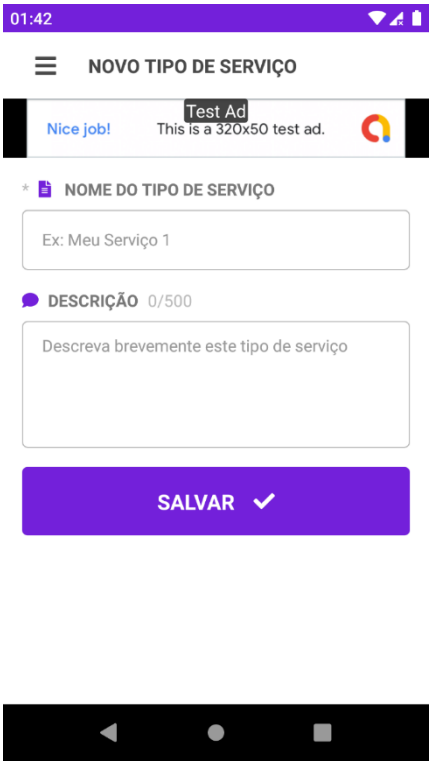
Figura 29 - Tela de listagem de clientes.



Fonte: Elaborado pelos autores (2020).

A Figura 30 representa a tela de cadastro de tipo de serviço, que possui um campo de digitação para o nome do tipo do serviço, que é de preenchimento obrigatório, um campo para descrever o tipo de serviço e um botão para salvar. Quando o botão é tocado o tipo do serviço é cadastrado no banco de dados.

Figura 30 - Tela de cadastro de tipo de serviço.

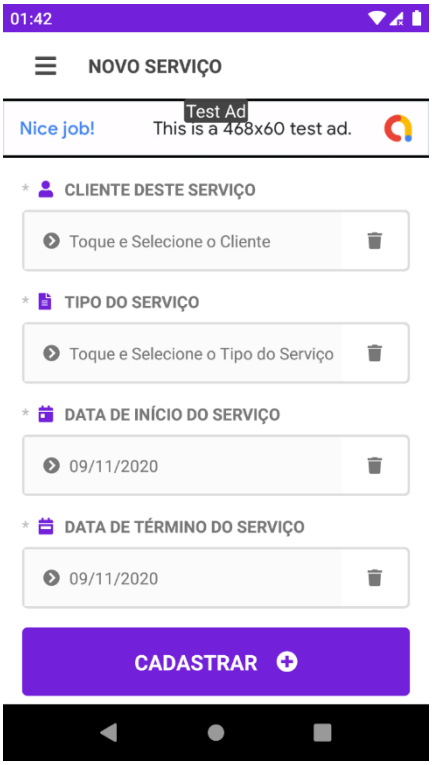


Fonte: Elaborado pelos autores (2020).

A Figura 31 corresponde a tela de cadastro de serviço. Ela, diferente das demais telas, possui apenas componentes de seleção dados via toque, então o usuário toca na parte cinza indicado pele ícone de seta e uma caixa de diálogo aparecerá. Esta caixa de diálogo para os primeiros dois componentes de seleção mostrará uma listagem de tipos de serviço e de clientes, respectivamente. Já os dois outros componentes abrirão uma caixa de seleção de data, que é um componente nativo do sistema operacional.

O ícone de lixeira corresponde a um botão que apaga o que foi selecionado da memória, no caso de uma seleção errônea. Ao fim do processo o usuário deve tocar o botão roxo para cadastrar o serviço no banco de dados.

Figura 31 - Tela de cadastro de serviço.

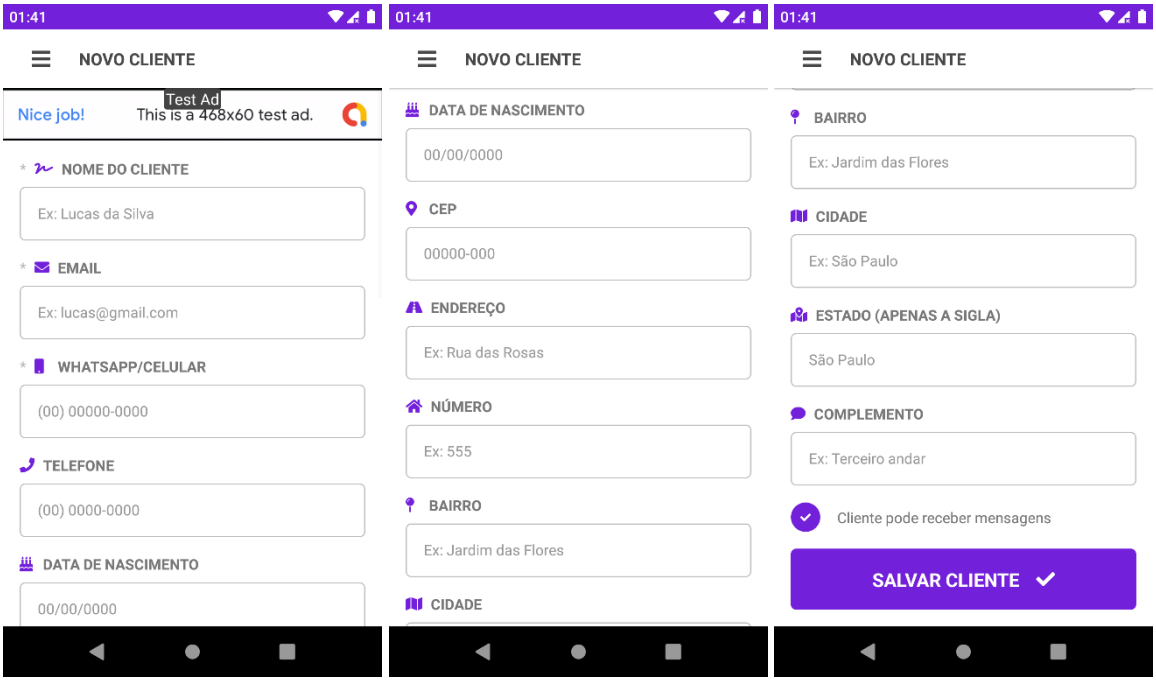


Fonte: Elaborado pelos autores (2020).

A Figura 32 corresponde ao cadastro de um novo cliente no sistema. Esta tela possui diversos campos de digitação, permitindo a inserção de nome, *e-mail*, telefones, endereço e vários outros, além de um componente *checkbox* para informar se esse cliente pode receber mensagens de pós-marketing ou não. Os três primeiros campos são obrigatórios, evidentes pelo asterisco antes do ícone na legenda do campo. Alguns dos campos possuem máscara de digitação para facilitar a entrada de dados, como é o caso dos campos de telefone e data de nascimento. Esta tela também busca endereço por CEP quando o campo de CEP perde o foco, isto é, quando o usuário acaba de digitar e seleciona outro campo para digitar.

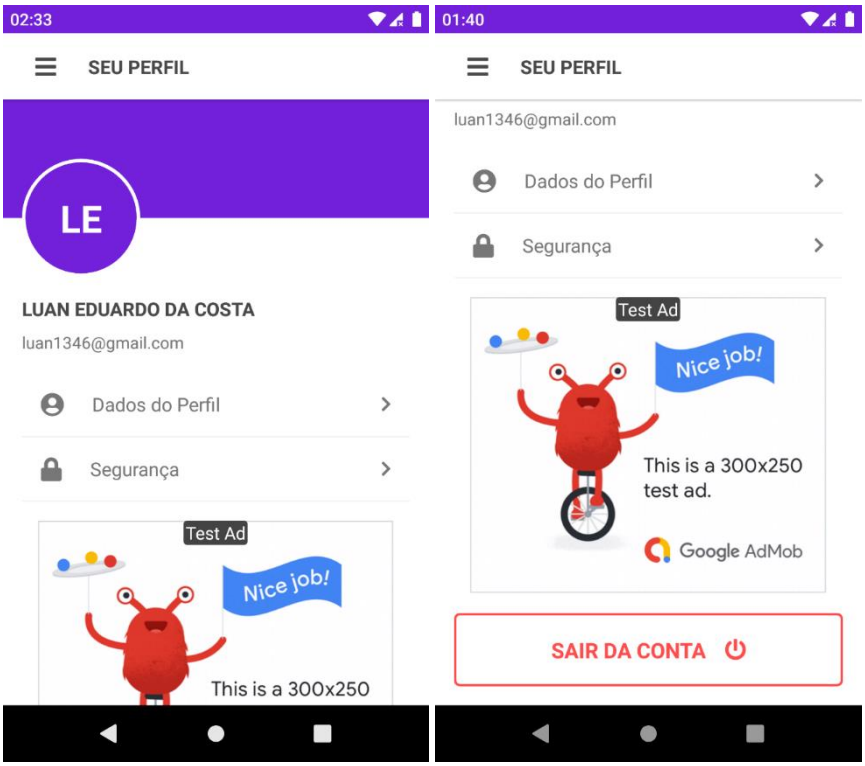
A Figura 33 corresponde a tela de perfil do usuário logado no sistema. Esta tela permite que o usuário navegue até a edição das informações referentes a sua conta, como nome, *e-mail* e senha e saia da sessão atual, voltando para o *login* assim que faz isto.

Figura 32 - Tela de cadastro de clientes.



Fonte: Elaborado pelos autores (2020).

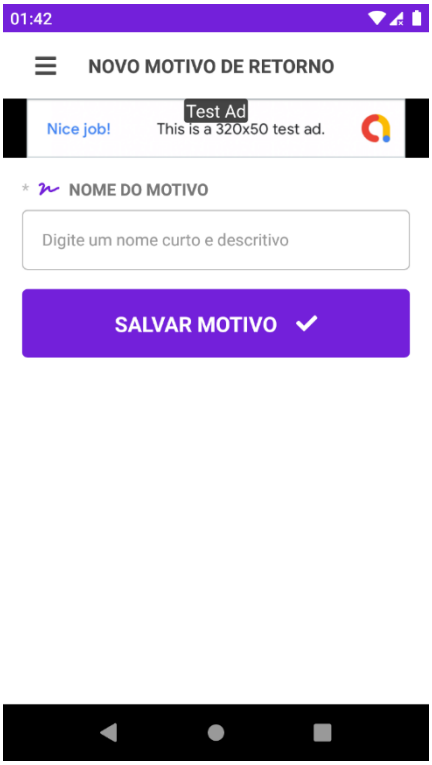
Figura 33 - Tela de perfil do usuário logado.



Fonte: Elaborado pelos autores (2020).

A Figura 34 corresponde a tela de inclusão de um novo motivo de retorno do cliente ao estabelecimento. Existe o campo de digitação do nome do motivo e um botão para salvar. Claramente o campo é obrigatório e o botão quando tocado grava o motivo no banco de dados.

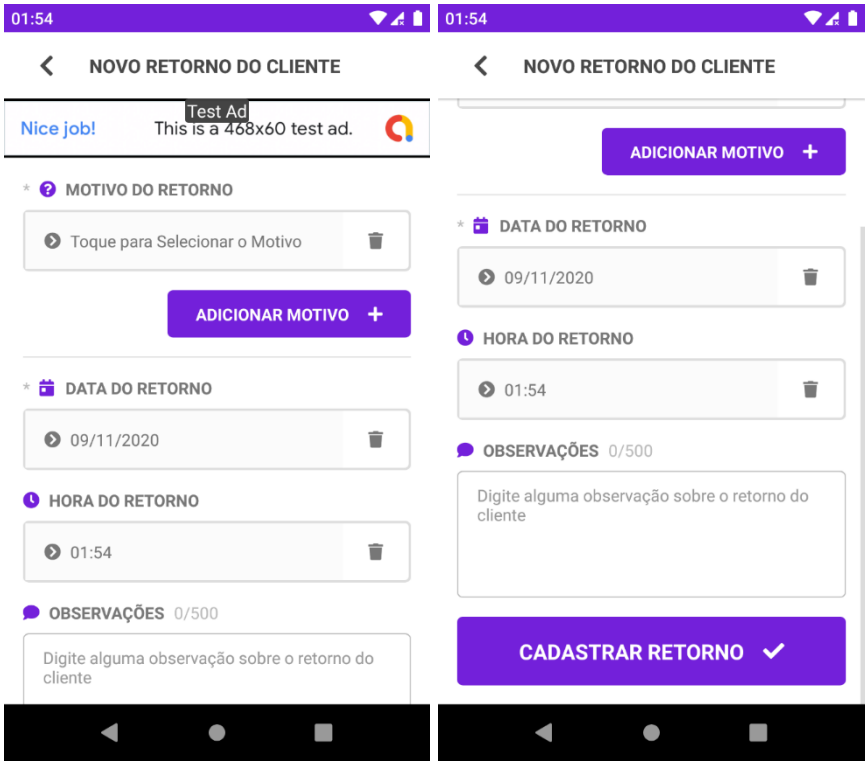
Figura 34 - Tela de cadastro de motivo de retorno ao estabelecimento.



Fonte: Elaborado pelos autores (2020).

A Figura 35 representa o cadastro de retorno do cliente ao estabelecimento. Ela contém um componente para selecionar o motivo do retorno, logo abaixo um botão para abrir a tela de adicionar um novo motivo de retorno, o campo de selecionar a data do retorno, outro para a hora do retorno, um campo de digitação de várias linhas para escrever observações sobre este retorno e o botão de salvar, que quando tocado cadastra o retorno do cliente no banco de dados.

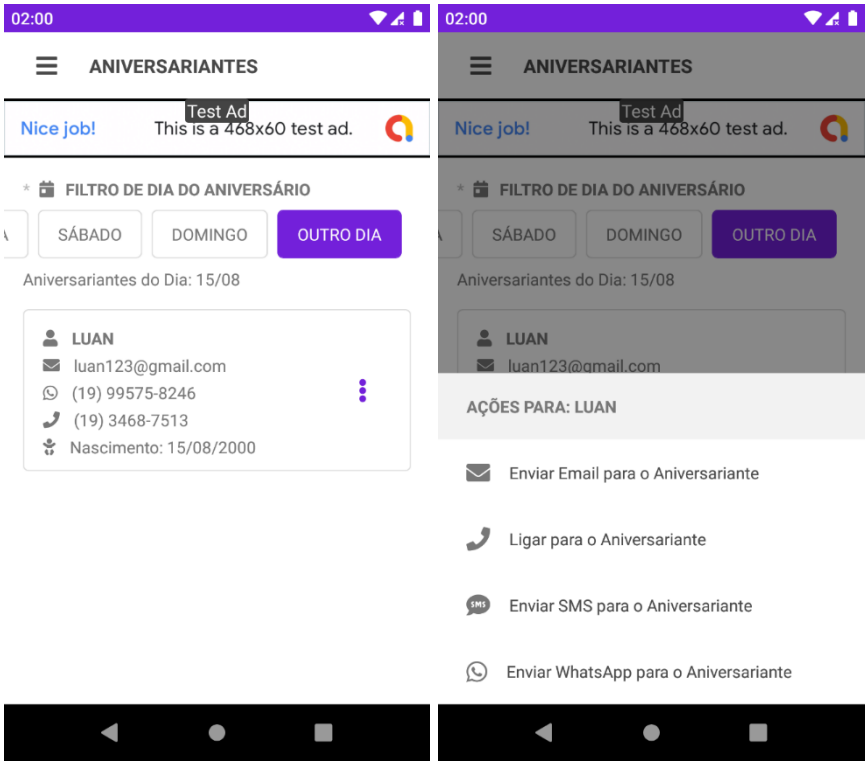
Figura 35 - Tela de cadastro de retorno do cliente ao estabelecimento.



Fonte: Elaborado pelos autores (2020).

A Figura 36 representa a listagem de aniversariantes do sistema, a qual possui um filtro de data de aniversário, que traz do banco de dados os clientes que fazem aniversário no dia e mês selecionados. Cada item da lista quando tocado abre a tela de detalhes de clientes e quando o ícone de três pontos verticais é tocado a caixa de diálogo de ações é aberta (mostrada no *print* da direita na Figura 36), a qual possui vários itens tocáveis que disparam ações.

Figura 36 - Tela de listagem de aniversariantes.



Fonte: Elaborado pelos autores (2020).

A Figura 37 mostra a tela de configuração de mensagens de aniversário. É nela que as mensagens dos aniversariantes são definidas. A primeira mensagem é enviada no dia exato do aniversário do cliente, a segunda é enviada quando se passa do dia do aniversário e a terceira é enviada quando ainda não chegou o dia do aniversário. Todas elas têm suporte para macros, o que permite a personalização da mensagem para cada cliente e o usuário não precisa se preocupar com qual mensagem mandar para o cliente porque o sistema faz isto sozinho.

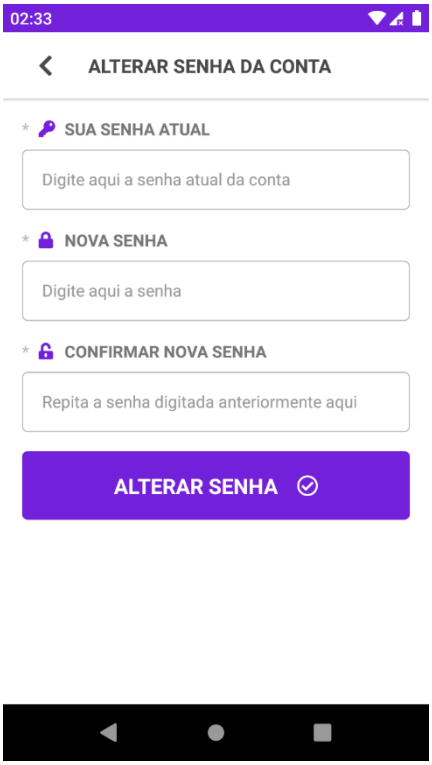
Figura 37 - Tela de configuração de mensagens de aniversário.

Fonte: Elaborado pelos autores (2020).

A Figura 38 corresponde a tela em que o usuário logado pode trocar a sua senha. Ela possui três campos, a senha atual dele, a nova senha e a confirmação da nova senha. Ao tocar no botão para alterar o usuário precisa entrar em sua conta novamente, então ele é redirecionado ao *login* e deve usar a nova senha para entrar.

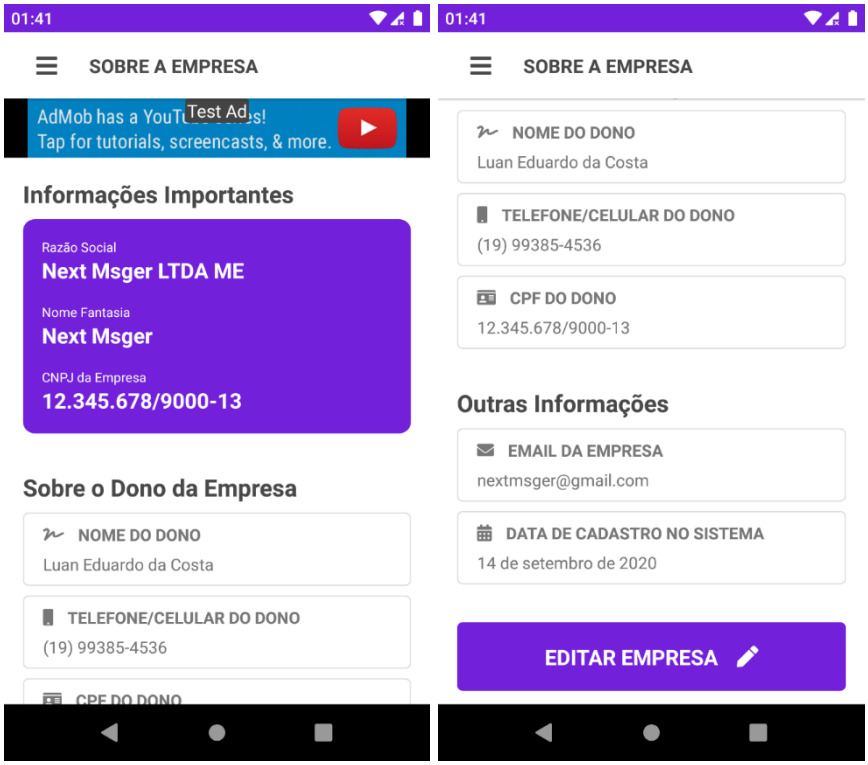
A Figura 39 corresponde a tela que mostra os dados da empresa do usuário logado. Esta é uma tela que apenas mostra dados, muito semelhante as outras telas de detalhes do app. Ao final dela existe um botão que direciona o usuário para a tela de edição dos dados da empresa, que é a mesma tela de cadastro de empresa, mas com os dados carregados nela e preparada para editar e não salvar.

Figura 38 – Tela de alteração de senha do usuário logado.



Fonte: Elaborado pelos autores (2020).

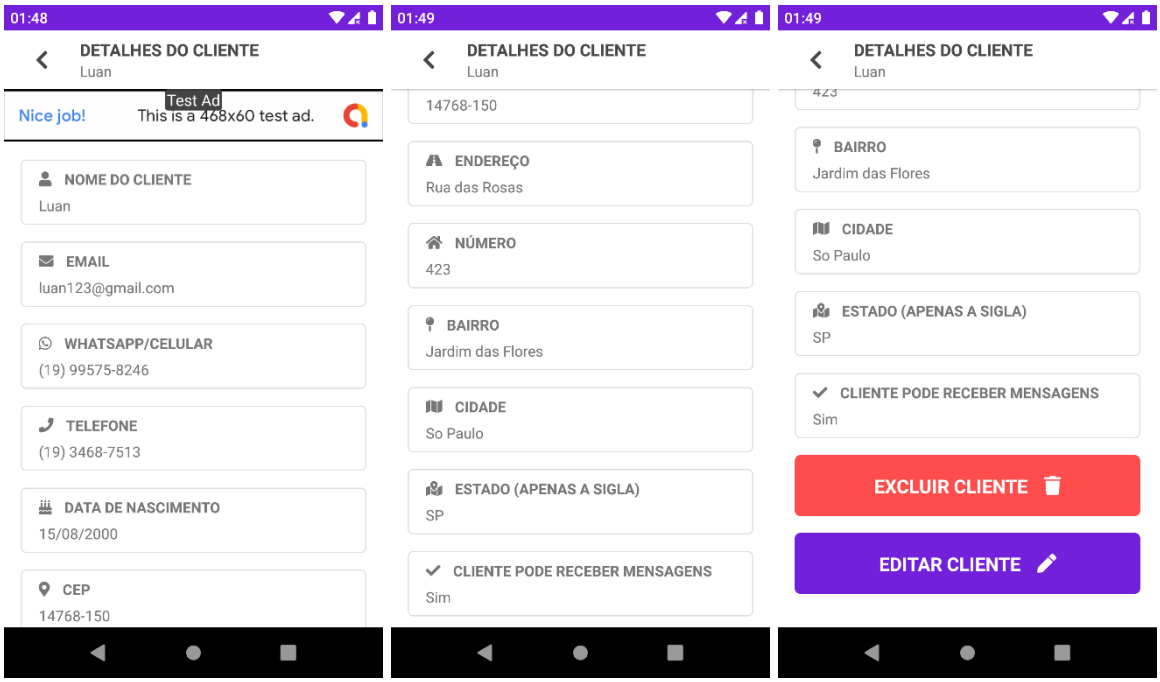
Figura 39 – Tela que mostra os dados da empresa.



Fonte: Elaborado pelos autores (2020).

A Figura 40 corresponde a tela que mostra os detalhes de um cliente cadastrado no sistema. Ela apresenta todas as informações e na sua parte inferior mostra dois botões, o vermelho serve para excluir o cliente e o roxo abre a tela de edição de cliente que é uma tela idêntica à de cadastro de cliente, mas rodando uma lógica de edição.

Figura 40 - Tela de detalhes de cliente.

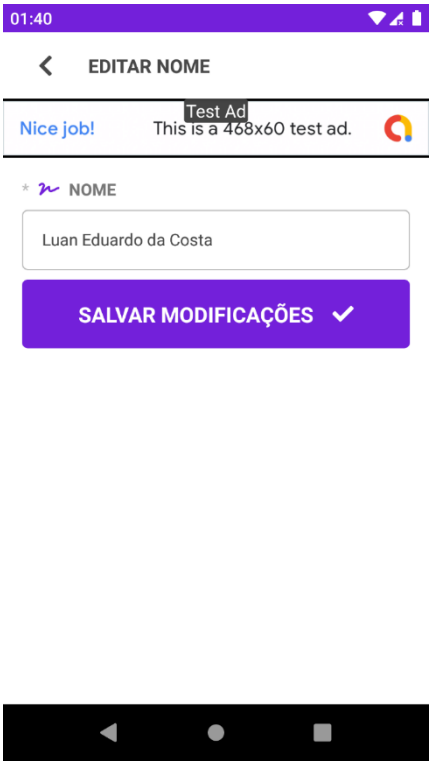


Fonte: Elaborado pelos autores (2020).

A Figura 41 mostra a tela em que o usuário logado pode editar seu nome no sistema. Além do anúncio no topo a tela tem um campo de digitação do nome e um botão para salvar a edição.

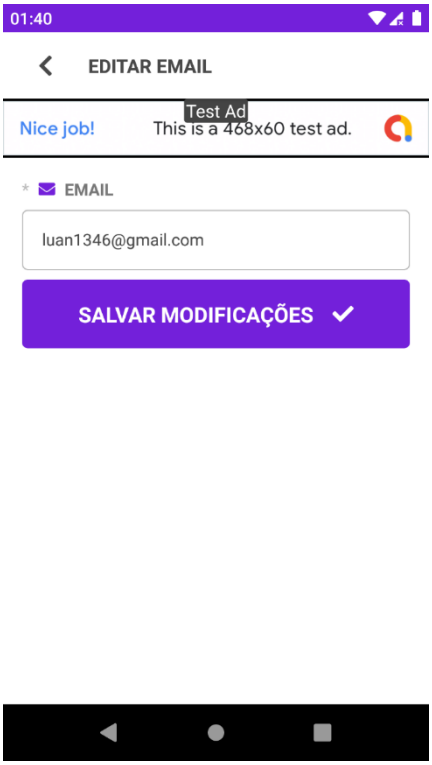
A Figura 42 mostra a tela em que o usuário logado pode editar seu *e-mail* no sistema. Além do anúncio no topo a tela tem um campo de digitação do *e-mail* e um botão para salvar a edição. O *e-mail* que será editado é o mesmo que é usado para fazer *login*.

Figura 41 - Tela de edição do nome do usuário logado.



Fonte: Elaborado pelos autores (2020).

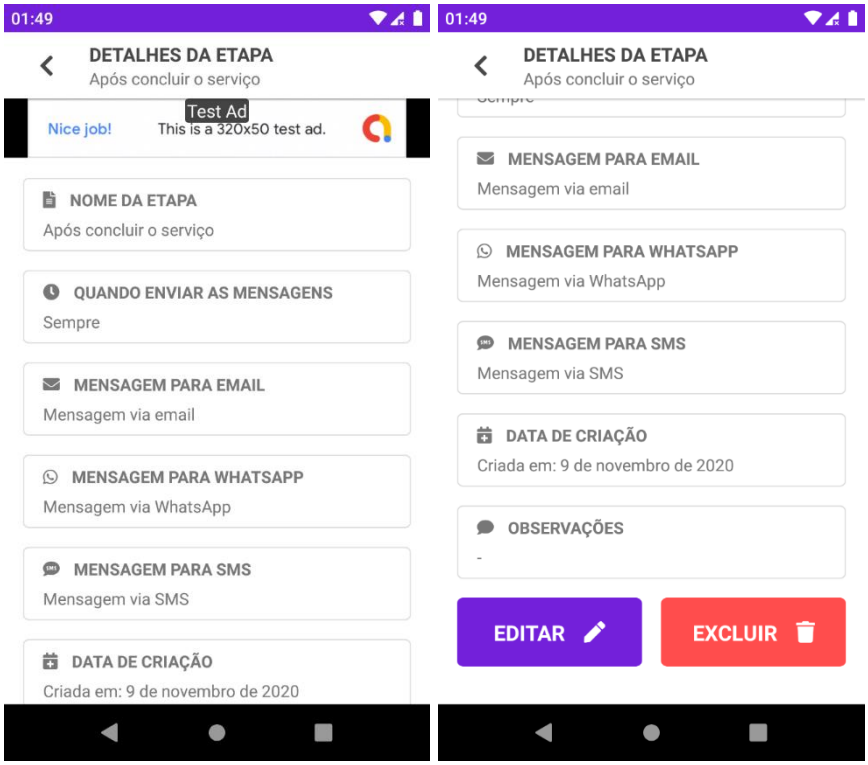
Figura 42 - Tela de edição do *e-mail* do usuário logado.



Fonte: Elaborado pelos autores (2020).

A Figura 43 mostra a tela de detalhes de etapa de *marketing*, que apenas mostra todos os dados da etapa e tem dois botões na sua parte inferior, um para editar a etapa, que abre a tela de cadastro de etapa configurada para editar e o botão de exclusão da etapa que se está visualizando.

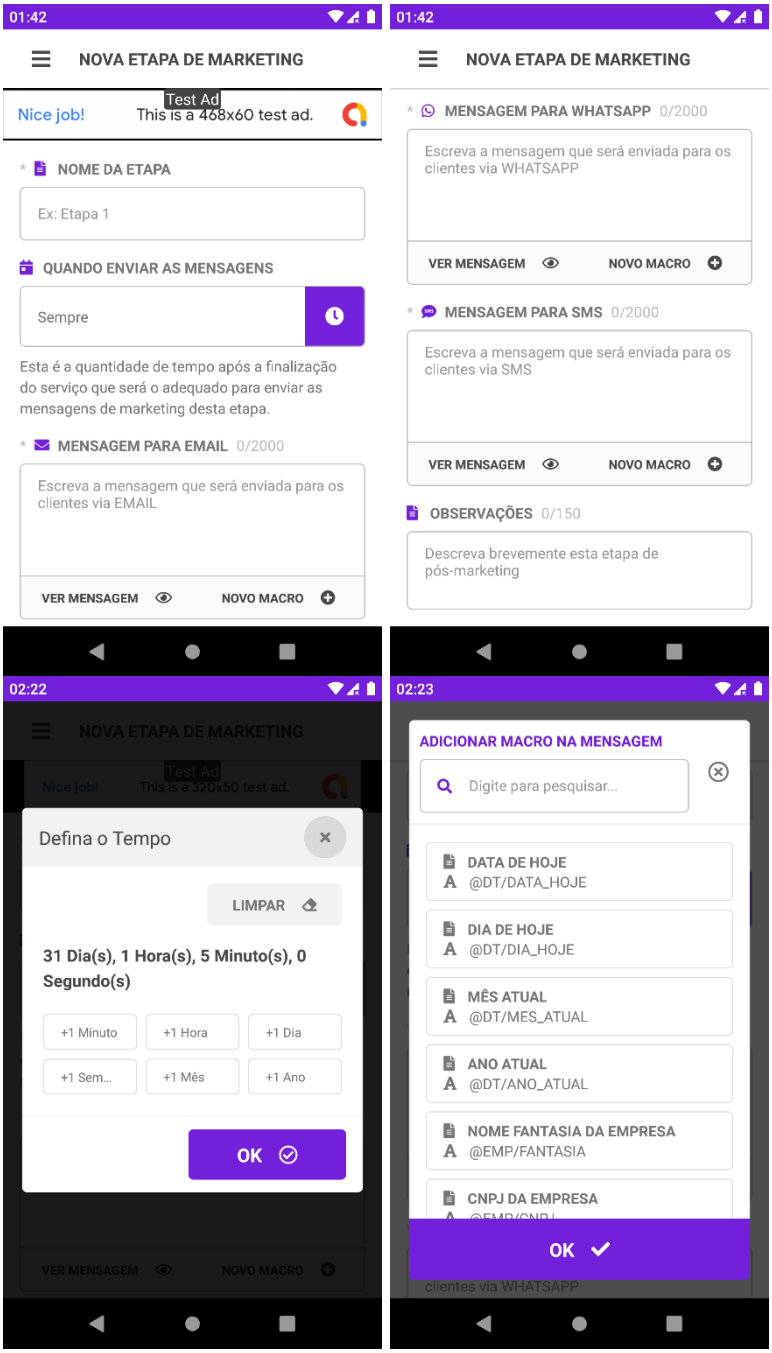
Figura 43 - Tela de detalhes de etapa de *marketing*.



Fonte: Elaborado pelos autores (2020).

A Figura 44 mostra a tela responsável por cadastrar etapas de marketing. Esta é uma das telas mais complexas de se entender e uma das que mais se demora para utilizar, porque ela tem um *builder* de tempo e muitas mensagens nas quais se pode adicionar macros. O *builder* de tempo é um componente desenvolvido pela equipe que facilita a definição de um tempo por parte do usuário, já que tudo o que ele deve fazer é tocar nos botões que somam determinada quantidade de tempo em uma variável.

Figura 44 - Tela de cadastro de etapa de *marketing*.



Fonte: Elaborado pelos autores (2020).

A Figura 45 mostra a tela na qual o usuário cadastra a avaliação de um serviço feita por um cliente. A tela tem o seletor de quantidade de estrelas, no qual cada estrela é tocável, o campo para escrever as observações feitas pelo cliente e o botão para salvar a avaliação no banco de dados.

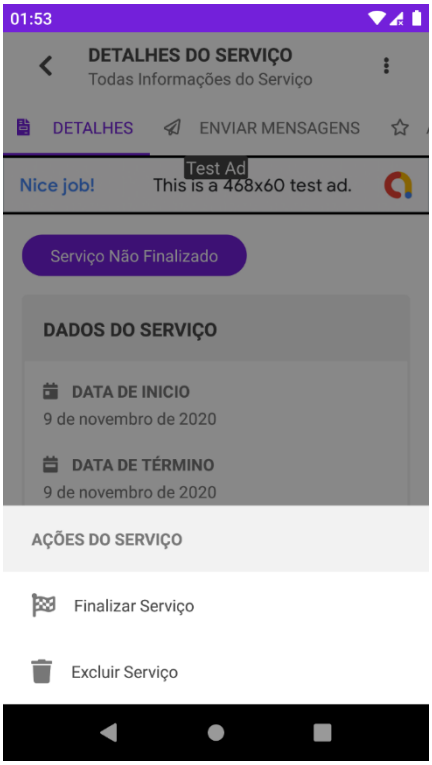
Figura 45 - Tela de cadastro de avaliação de um serviço.

Fonte: Elaborado pelos autores (2020).

A Figura 46 mostra a tela de detalhes do serviço com a caixa de diálogo de ações do serviço aparecendo. Esta caixa é mostrada quando se toca nos três pontos verticais na barra de navegação superior na qual fica o título da página. As ações mostradas são a de finalizar todos os processos de marketing para o serviço em questão e excluir permanentemente o mesmo.

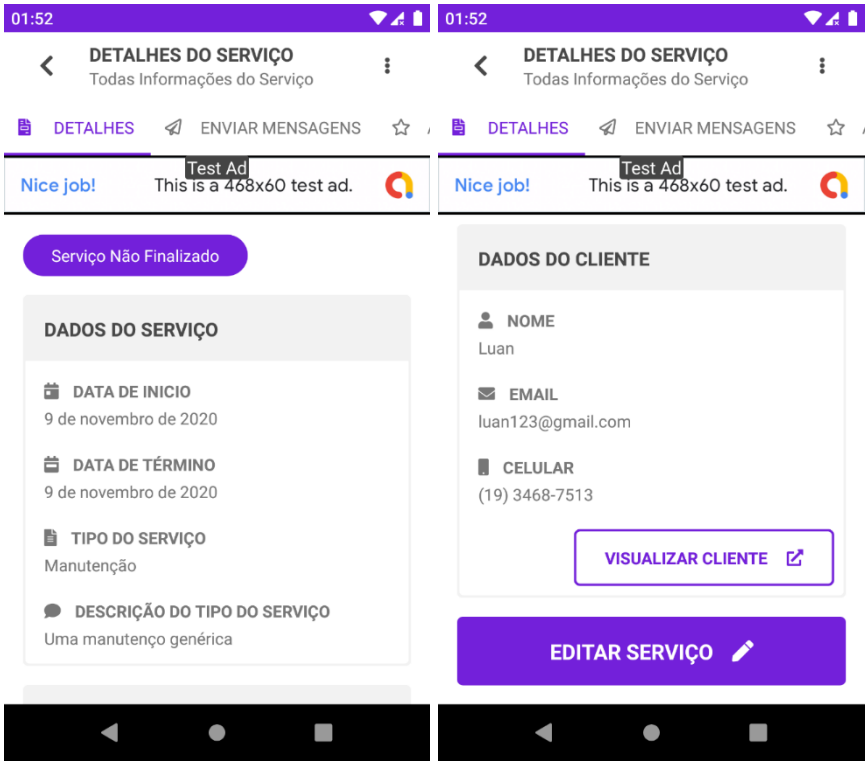
A Figura 47 mostra a tela de detalhes de serviço com a aba de detalhes selecionada. Esta aba é responsável por exibir os dados do serviço, como o cliente, tipo de serviço, data de início, término e promover a navegação para a tela de edição de serviço e para os detalhes do cliente caso o usuário queira ver mais sobre o cliente deste serviço.

Figura 46 - Tela de detalhes de serviço mostrando ações do serviço.



Fonte: Elaborado pelos autores (2020).

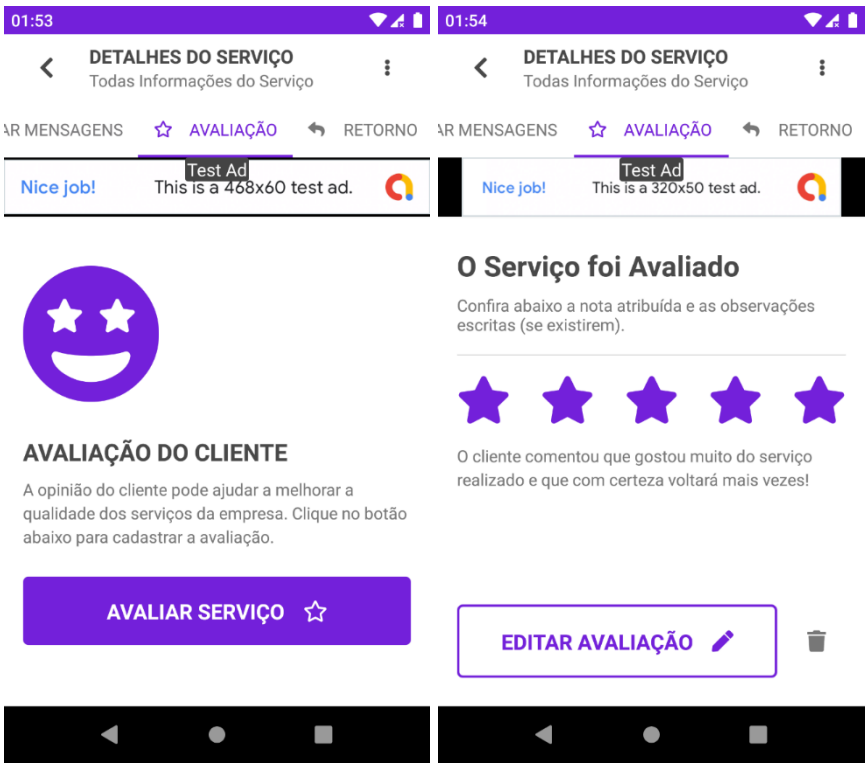
Figura 47 - Tela de detalhes de serviço com aba de detalhes selecionada.



Fonte: Elaborado pelos autores (2020).

A Figura 48 mostra a tela de detalhes de serviço com a aba de avaliação do serviço selecionada. Esta tela serve para o usuário cadastrar a avaliação do cliente sobre o serviço e ela tem dois estados: quando o serviço ainda não foi avaliado e quando ele foi. No primeiro estado um descritivo do que é a avaliação junto com um botão para abrir a tela de avaliação são mostrados. No segundo estado a quantidade de estrelas e a observação cadastrada são exibidos junto com o botão de apagar e o de editar a avaliação.

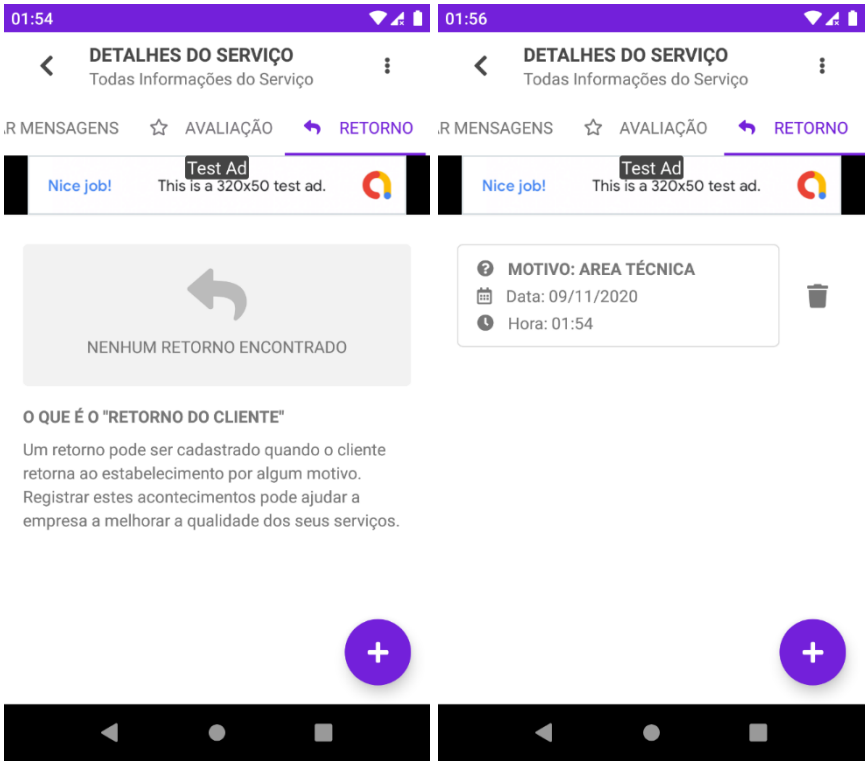
Figura 48 - Tela de detalhes de serviço com aba de avaliação selecionada.



Fonte: Elaborado pelos autores (2020).

A Figura 49 mostra a tela de detalhes de serviço com a aba de retornos do cliente selecionada. Esta tela mostra uma mensagem quando não existem retornos explicando o que a tela é. Quando existem retornos eles são exibidos em forma de lista, na qual todo item possui um botão com o ícone de lixeira para excluir o retorno e quando o item é tocado a tela de edição de retorno do cliente é aberta. Além disso o FAB abre a tela de cadastro de retorno do cliente.

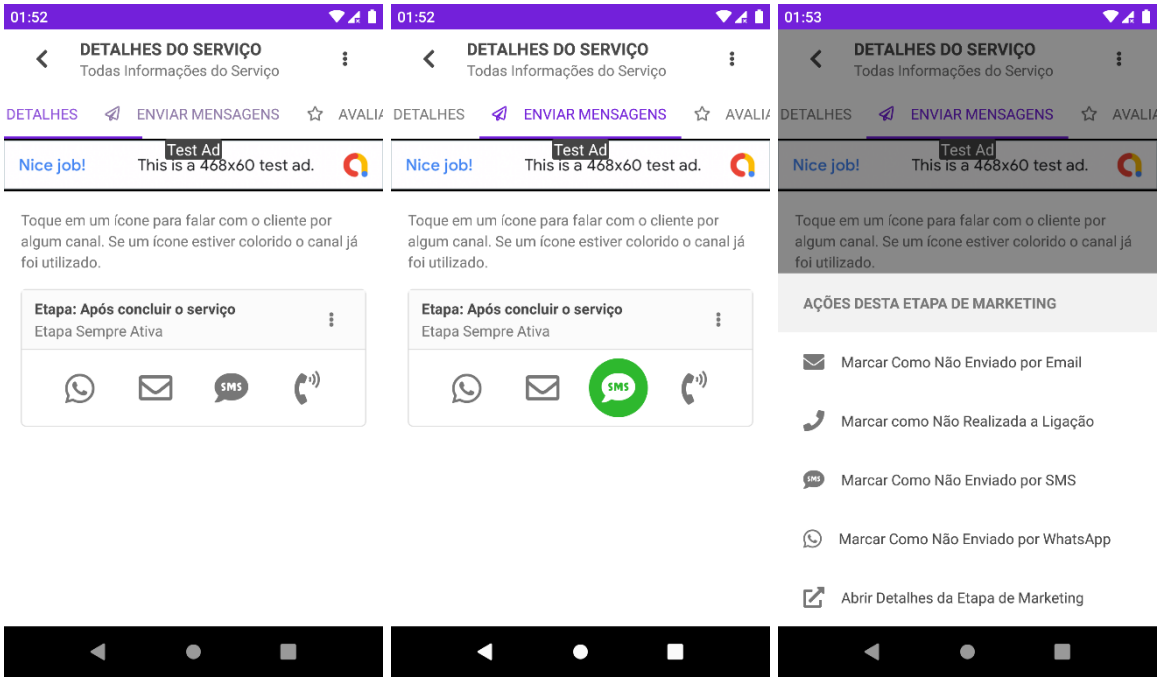
Figura 49 - Tela de detalhes de serviço com aba de retornos do cliente selecionada.



Fonte: Elaborado pelos autores (2020).

A Figura 50 mostra a tela de detalhes de serviço com a aba de enviar mensagens para o cliente do serviço selecionada. Esta é uma das telas mais importantes do app, porque é nela que as mensagens de marketing cadastradas nas etapas são enviadas para o cliente do serviço que está vendo os detalhes. Para cada etapa de marketing um item da lista vai ser exibido. Este item possui quatro botões e cada um deles é um canal de comunicação com o cliente. Exceto o botão com o ícone de telefone, que abre o aplicativo de telefone do celular os outros botões abrem os aplicativos que enviam mensagens de texto. Quando o botão com os três pontos verticais é clicado é mostrada a caixa de diálogo com as ações para aquela etapa de marketing. Estas ações são de dois tipos: desmarcar o envio por algum canal (as quatro primeiras ações) ou abrir os detalhes da etapa de marketing (a última ação).

Figura 50 - Tela de detalhes de serviço com a aba de enviar mensagens selecionada.

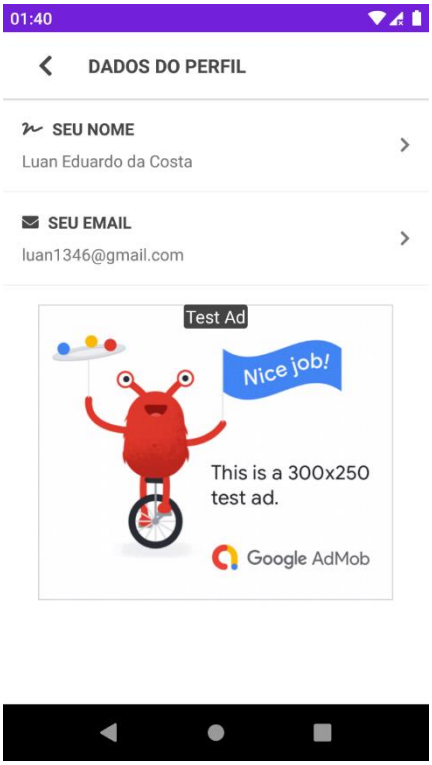


Fonte: Elaborado pelos autores (2020).

A Figura 51 mostra a tela que exibe os dados do perfil do usuário logado. Esta tela contém dois itens clicáveis que exibem o *e-mail* e nome do usuário e quando tocados abrem a tela de edição desses dados.

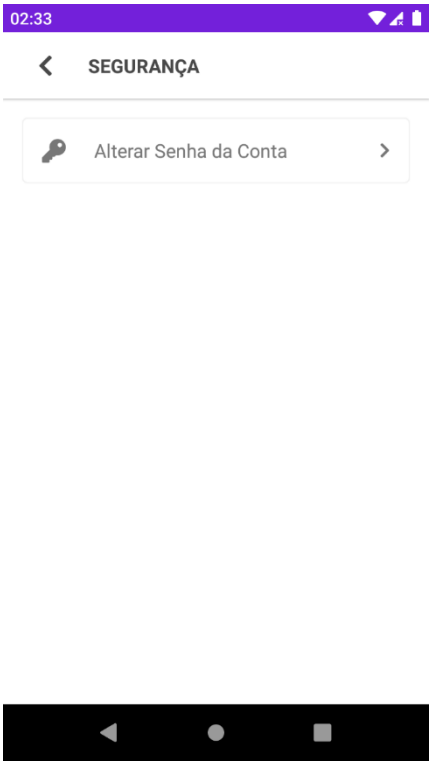
A Figura 52 mostra a tela que exibe opções de controle de segurança da conta do usuário logado. Na versão atual do app esta tela apenas possui um item tocável que leva o usuário a tela de alteração da senha, mas em próximas versões é nela que estariam a ativação da verificação em duas etapas ou a ativação do *analytics* do aplicativo, por exemplo.

Figura 51 - Tela que mostra os dados do perfil do usuário logado.



Fonte: Elaborado pelos autores (2020).

Figura 52 - Tela de opções de segurança do usuário logado.



Fonte: Elaborado pelos autores (2020).

5.2 Avaliação Heurística

Seguindo as tendências da execução de testes mais rápidos, fáceis, de menor custo e com bons resultados, Nielsen (1993) propõe a denominada Engenharia Econômica de Usabilidade *(Discount Usability Engineering*), utilizando como principal método, a Avaliação Heurística. A avaliação heurística é uma técnica de inspeção de usabilidade executada por examinadores que seguem um conjunto de princípios de usabilidade, as heurísticas, e avaliam todos os elementos de interface com o usuário, com o objetivo de encontrar falhas de usabilidade.

5.2.1 Heurísticas de Nielsen

O objetivo das heurísticas é fazer uma avaliação da qualidade da interface em relação à usabilidade, e assim, detectar precocemente problemas desta. As heurísticas são compostas por 10 princípios fundamentais de usabilidade, que são:

1. **Visibilidade de Status do Sistema:** isso significa que você precisa se certificar de que a interface sempre informe ao usuário o que está acontecendo, ou seja, todas as ações precisam de *feedback* instantâneo (retorno do que está acontecendo) para orientá-lo. Dez segundos é o limite para manter a atenção do usuário focada nas ações que realiza. (Sequência do site | barra de progressão - mostrando o estágio da operação, seja qual for, exemplo *download* 10% 25% até 100%).
2. **Relacionamento entre a interface do sistema e o mundo real:** não usar palavras de sistema (Ex. informantes – vou *startar* o processo), que não fazem sentido para o usuário. Toda a comunicação do sistema precisa ser contextualizada ao usuário, e ser coerente com o chamado modelo mental dele.
3. **Liberdade e controle do usuário:** facilite as “saídas de emergência” (se deu um problema, voltar, avançar, atualizar) para o usuário, permitindo desfazer ou refazer a ação no sistema e retornar ao ponto anterior, quando estiver perdido ou em situações inesperadas.
4. **Consistência (fala uma coisa faz outra):** fale a mesma língua o tempo todo (tenha padrões), e nunca identifique uma mesma ação com ícones ou

palavras diferentes. Trate coisas similares da mesma maneira, facilitando a identificação do usuário. Padronize!

5. **Prevenção de erros:** na tradução livre das palavras do próprio Nielsen “Ainda melhor que uma boa mensagem de erro é um design cuidadoso que possa prevenir esses erros”. Por exemplo, ações definitivas como deleções ou solicitações podem vir acompanhadas de um *checkbox* ou uma mensagem de confirmação (tem certeza de que quer apagar?).
6. **Reconhecimento ao invés de lembrança:** evite acionar a memória do usuário o tempo todo, fazendo com que cada ação precise ser revista mentalmente antes de ser executada. Permita que a interface ofereça ajuda contextual, e informações capazes de orientar as ações do usuário – ou seja – que o sistema dialogue com o usuário (saber sempre onde está, voltar, sem que ter que memorizar).
7. **Flexibilidade e eficiência de uso:** o sistema precisa ser fácil para usuários leigos, mas flexível o bastante para se tornar ágil à usuários avançados. Essa flexibilidade pode ser conseguida com a permissão de teclas de atalhos (ctrl+z), por exemplo. No caso de websites, uso de máscaras e navegação com *tab* (pular para o próximo campo/tabular documento) em formulários são bons exemplos.
8. **Estética e *design* minimalista (poluição visual):** evite que os textos e o projeto falem mais do que o usuário necessita saber. Os diálogos do sistema precisam ser simples, diretos e naturais, presentes apenas nos momentos em que são necessários.
9. **Ajude os usuários a reconhecer, diagnosticar e sanar erros:** as mensagens de erro do sistema devem possuir uma redação simples e clara, que ao invés de intimidar o usuário com o erro indique uma saída construtiva ou possível solução (erro 404 – *page not found* – deve-se simplificar – não encontrei a página, como se estivesse conversando com a pessoa).
10. **Ajuda e documentação:** um bom *design* deveria evitar ao máximo a necessidade de ajuda na utilização do sistema. Ainda assim, um bom conjunto de documentação e ajuda deve ser utilizado para orientar o usuário em caso de dúvida. Deve ser visível, facilmente acessada, oferecendo uma ferramenta de busca na ajuda. O ideal é que um *software* seja tão fácil de usar (intuitivo) que não necessite de ajuda ou documentação. Se for

necessária a ajuda deve estar facilmente acessível on-line (ex. Microsoft Word – linha – o *help* do Microsoft Word acaba não servindo para nada).

5.2.2 Avaliação das Heurísticas de Nielsen

Depois da aplicação da Avaliação Heurística, é importante fazer uma análise dos problemas levantados. Deve-se analisar e categorizar as informações obtidas a fim de priorizá-las para que posteriormente possam ser alocados recursos para desenvolver as soluções.

Dentro deste contexto, deve-se realizar a estimativa da seriedade dos problemas levantados, ou seja, o grau de severidade através de notas em aspectos preestabelecidos. Os participantes devem também avaliar os problemas levantados pelos outros avaliadores e ordenar as estimativas de seriedade.

O grau de severidade é definido a partir do impacto que um problema tem para o mercado consumidor do produto. Para determinar valores para os Fatores de severidade, deve-se usar uma escala de 0 (fraco) a 4 (fortíssimo) para medi-los, sendo considerado:

- Frequência: É um problema comum ou raramente experimentado?
- Impacto: Será fácil ou difícil para os usuários o superar?
- Persistência: sempre ou às vezes?
- Impacto de Mercado: popularidade do produto.

Após determinado os fatores de severidade, realiza-se a Estimativa de Severidade, que deve ser apontada como:

- 0 = Não concordo que seja um problema de usabilidade;
- 1 = Problema apenas estético: não precisa ser reparado, a menos que
- haja tempo extra no projeto;
- 2 = Pequeno problema de usabilidade: deve ser resolvido, com baixa prioridade;
- 3 = Grande problema de usabilidade: é importante repará-lo. Deve ser resolvido com alta prioridade;

- 4 = Catástrofe de usabilidade: é imperativo repará-lo antes do lançamento do produto.

Neste contexto, os itens 3 e 4 devem ser resolvidos com prioridade e com urgência. O questionário utilizado para avaliar o aplicativo NextMsger contém dez questões que estão agrupadas em suas respectivas heurísticas. Ao final do processo, este questionário foi tabulado e as informações analisadas a fim de que potenciais problemas de usabilidade pudessem ser encontrados. O questionário pode ser encontrado no Apêndice A.

5.2.3 Discussão dos Resultados da Avaliação Heurística

Os resultados mostraram que a primeira heurística, que é a de visibilidade de estado do sistema (determinada como H1), foi muito bem atendida, visto que 100% dos usuários ficaram satisfeitos com o *feedback* de qualidade proporcionado pelo sistema durante a navegação entre telas.

A segunda heurística que avalia a similaridade entre o sistema e o mundo real (determinada como H2), foi avaliada positivamente por 100% dos usuários, mostrando que o sistema utiliza expressões de fácil entendimento e organiza as informações seguindo um fluxo lógico e natural.

A terceira heurística, que avalia o controle e liberdade do usuário (determinada como H3), mostrou que 50% dos usuários não estão satisfeitos com o controle que tem sobre as ações realizadas no sistema e a personalização do mesmo. Esta porcentagem não chega a indicar um problema grave, mas deve impactar os resultados na conclusão da avaliação heurística e tomada de decisões no desenvolvimento do projeto.

Já para a quarta heurística, que trata da consistência e padrões do sistema (determinada como H4), 100% dos usuários ficaram satisfeitos. Apenas com os resultados obtidos até agora já é possível afirmar que a maioria dos usuários não tem grandes confusões no sistema e entendem os resultados de suas ações.

A quinta heurística, que trata da prevenção de erros (determinada como H5), aponta que 100% dos usuários afirmaram que os ícones do sistema os auxiliam a evitar erros. Os dados confirmam, desta forma, que há uma clareza visual na interface, que utiliza muito dos ícones para ilustrar as ações e tipos de informações.

A sexta heurística (determinada como H6), que analisa os aspectos de reconhecimento no lugar de memorização aponta que 100% dos usuários acreditam que o sistema possui instruções e opções sempre visíveis quando apropriadas para uso.

A sétima heurística avaliou a flexibilidade e a eficiência de uso do sistema (determinada como H7). Esta etapa apontou que 75% dos usuários estão bem satisfeitos a quantidade de coisas que é possível personalizar no sistema. Este percentual indica que tem alguns usuários que desejam mudar mais detalhes visuais na aplicação, mas eles não são a maioria, portanto é algo que pode ser considerado para melhorar a experiência dos usuários, mas que não é necessário de se fazer agora.

A oitava heurística, a de desenho estético e minimalista (determinada como H8), provou que o sistema não aparenta mostrar nenhuma informação que não seja necessária ao usuário, visto que, 100% dos usuários se sentiram satisfeitos com as informações mostradas pelo sistema.

A penúltima e nona heurística, trata do auxílio no reconhecimento, diagnóstico e recuperação de erros (determinada como H9). Ela apontou que 33,33% dos usuários acreditam que o sistema utiliza uma linguagem simplificada nos erros que mostra ao usuário, mas que muitas vezes não identifica o que exatamente falhou quando é o caso de erros ao salvar, editar ou outra ação. Além disso, uma parcela destes usuários, no entanto, ressaltou que apesar de as mensagens serem facilmente compreendidas, elas não oferecem sugestões de solução, o que em certas situações pode representar um grande problema para os usuários. Esta heurística representa um dos maiores problemas do sistema e que deve ser corrigida de forma rápida.

A última heurística, que analisa a ajuda e a documentação da aplicação (determinada como H10), mostrou, por unanimidade, que o sistema não possui opção de ajuda. Nela 100% dos usuários apontaram esta grave falha que deve corrigida de forma urgente para dar aos usuários um manual de uso claro e fácil sempre que precisarem.

Com todas os resultados da avaliação em mãos é possível verificar que a heurística H10 foi o maior problema encontrado no sistema: a falta de uma opção de ajuda para os usuários acessarem sempre que precisarem. Outra problemática destacada foi a ausência de uma identificação do motivo de um erro e a

apresentação de formas de se solucionar este problema. Melhorias envolvendo estas heurísticas devem ser feitas com uma prioridade alta, já que melhorarão significativamente a usabilidade e resiliência a erros do sistema, contribuindo, além disso, para melhorar a experiência do usuário de forma geral.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo final o desenvolvimento de um aplicativo para *smartphones* que facilita os processos de pós-marketing de empresas prestadoras de serviço, garantindo uma melhor comunicação com seus clientes, aumentando, desta forma, a experiência na contratação de um serviço e a chance de fidelização dos mesmos.

Através das análises de requisitos funcionais, não funcionais e *softwares* similares, foram identificadas várias atividades a serem desenvolvidas ao longo dos meses. Para gerir estas atividades foi selecionada e aplicada a metodologia ágil SCRUM e um conjunto de tarefas foram priorizadas para que uma versão mínima e, ao mesmo tempo, viável pudesse ser criada o mais rapidamente possível.

Não houveram grandes dificuldades no desenvolvimento do projeto, dado que os membros do grupo já tinham um bom conhecimento e experiência com as ferramentas utilizadas e sabiam o que tinha que ser feito para tirar do papel um produto minimamente viável dentro do prazo estipulado. Porém, as únicas partes que foram definidas, mas que não dariam para serem implementadas no aplicativo dentro do prazo são a monetização com assinatura de planos e o envio automático de mensagens para clientes, visto que são tarefas complexas e a segunda depende completamente da primeira, já que o envio automático vem acompanhado de custos.

Como objetivo futuro, a equipe planeja realizar a implementação do aplicativo na Apple Store para que ele também possa ser utilizado em celulares iOS e o desenvolvimento de um *website*, que atenderá as pessoas que não possuem tanta familiaridade com aplicativos de celular ou que preferem a utilização em um computador. Além disso uma série de melhorias de interface podem ser implementadas para deixar a experiência do usuário mais agradável, fluída, produtiva e mais funcionalidades podem ser adicionadas, deixando o sistema ainda mais robusto, como o gerenciamento de usuários da empresa, delegação de permissões por usuário, o envio automático de mensagens, entre outras.

6.1 Download do aplicativo

O aplicativo está disponível para download na loja de aplicativos oficial dos dispositivos Android, a Play Store. Com a leitura do código QR (*Quick Response*) apresentado na Figura 53 é possível acessar a página de *download* da aplicação.

Figura 53 – Código QR para baixar o aplicativo na Play Store

Fonte: Elaborado pelos autores (2020).

REFERÊNCIAS

BABEL. Disponível em: <https://babeljs.io/docs/en/>. Acesso em: 12 de mar. 2020.

FIREBASE. Disponível em: <https://firebase.google.com/?hl=pt-br>. Acesso em: 28 de fev. 2020.

GIT. Disponível em: <https://git-scm.com/about>. Acesso em: 27 de fev. 2020.

GITHUB. Disponível em: <https://github.com/>. Acesso em: 16 de mar. 2020.

JAVASCRIPT. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 28 de fev. 2020.

LITTLEFIELD, Andrew. GUIA DA METODOLOGIA ÁGIL E SCRUM PARA INICIANTES. Disponível em <https://blog.trello.com/br/scrum-metodologia-agil/>. Acesso em: 13 de mar. 2020.

MÓDULO DE PÓS VENDA. Art Informática. Disponível em: <https://www.crmvendamais.com/posvendas.aspx/>. Acesso em: 28 de fev. 2020.

MEU PÓS-VENDA. Disponível em: <https://www.meuposvenda.com/>. Acesso em: Acesso em: 28 de fev. 2020.

MOMENT. Disponível em: <https://momentjs.com/docs/>. Acesso em: 12 de mar. 2020

NODE.JS. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 12 de mar. 2020

NOVA CENTRAL. Tray Aplicativos. Disponível em: <https://aplicativos.tray.com.br/aplicativo/nova-central/>. Acesso em: 28 de fev. 2020.

ORACLE ELOQUA: INTELLIGENT CAMPAIGN ORCHESTRATION. Oracle. Disponível em: < https://www.oracle.com/marketingcloud/products/marketing-automation/>. Acesso em: 28 de fev. 2020.

REACT. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 26 de fev. 2020.

REACT NATIVE. Disponível em: <https://reactnative.dev/>. Acesso em: 26 de fev. 2020.

REACTOTRON. Disponível em: <https://infinite.red/reactotron/>. Acesso em: 12 de mar. 2020.

REACT-NAVIGATION. Disponível em <https://reactnavigation.org/docs/getting-started/> Acesso em 12 de mar. 2020.

REDUX. Disponível em: <https://redux.js.org/> Acesso em: 12 de mar. 2020.

SCRUM. Disponível em <https://www.desenvolvimentoagil.com.br/scrum/>. Acesso em 12 de mar. 2020.

SCRUM: FRAMEWORK ÁGIL PARA PROJETOS COMPLEXOS. Disponível em: <http://www.metodoagil.com/scrum/>. Acesso em: 13 de mar. 2020.

STYLED-COMPONENTS. Disponível em: <https://styled-components.com/docs/> Acesso em 12 de mar. 2020.

TRELLO. Disponível em: <https://help.trello.com/article/708-what-is-trello/>. Acesso em: 27 de fev. 2020.

VISUAL STUDIO CODE. Disponível em: <https://code.visualstudio.com/> Acesso em 12 de mar. 2020.

YARN. Disponível em: <https://yarnpkg.com/> Acesso em 12 de mar. 2020.

APÊNDICE A - QUESTIONÁRIO DE AVALIAÇÃO HEURÍSTICA

Avalie o aplicativo Next Msger, de acordo com as 10 Heurísticas de Nielsen descritas abaixo:

H1 - Visibilidade do Estado do Sistema

- O sistema possui *feedback* rápido indicando o que você está fazendo na interface no momento?
    - () Sim
    - () Não
- O sistema possui *feedback* rápido indicando em qual interface você está acessando no momento?
    - () Sim
    - () Não
- O sistema possui *feedback* rápido indicando como você pode prosseguir na navegação do sistema?
    - () Sim
    - () Não

H2 - Correspondência entre o Sistema e o Mundo Real

- O sistema utiliza palavras, termos, expressões e conceitos familiares ao usuário?
    - () Sim
    - () Não
- As informações aparecem em uma ordem lógica e natural como se fossem representações do mundo real?
    - () Sim
    - () Não

H3 - Controle e Liberdade do Usuário

- O sistema possui alguma saída de emergência?
    - () Sim
    - () Não

- O sistema possui funções “Desfazer” e “Refazer” facilmente disponíveis?
    - () Sim
    - () Não

H4 - Consistência e Padrões

- O sistema usa apenas palavras e expressões de fácil entendimento e interpretação?
    - () Sim
    - () Não
- O sistema possui padrões e estilos consistentes?
    - () Sim
    - () Não

H5 - Prevenção de Erros

- O sistema possui ícones que ajudam a impedir a ocorrência de erros?
    - () Sim
    - () Não

H6 - Reconhecimento no Lugar de Memorização

- O sistema possui instruções, ações e opções visíveis ou facilmente recuperáveis sempre que apropriado para o uso?
    - () Sim
    - () Não

H7 - Flexibilidade e Eficiência de Uso

- O sistema possui características de personalização de ações que podem ser feitos pelo próprio usuário?
    - () Sim
    - () Não
- O sistema possui atalhos para aumentar a eficiência de usuários novatos ou experientes?
    - () Sim
    - () Não

H8 - Desenho Estético e Minimalista

- O sistema possui diálogo apenas com informações relevantes e necessárias?
    - () Sim
    - () Não
- Os links disponibilizados pela aplicação sempre trazem informações extras que são necessárias?
    - () Sim
    - () Não

H9 - Auxílio no Reconhecimento, Diagnóstico e Recuperação de Erros

- O sistema possui mensagens de erros com linguagem simples?
    - () Sim
    - () Não
- O sistema possui mensagens de erros que indicam precisamente o problema?
    - () Sim
    - () Não
- O sistema possui mensagens de erros com sugestão de soluções construtivas?
    - () Sim
    - () Não

H10 - Ajuda e Documentação

- O sistema possui opção de ajuda?
    - () Sim
    - () Não
- O sistema possui opção de ajuda de fácil acesso ou localização?
    - () Sim
    - () Não