UNIVERSIDADE FEDERAL DO PARANÁ

LUAN GABRIEL DA SILVA
RICARDO KROHN DE JESUS

PLATAFORMA PARA CONTROLE DE ESTACIONAMENTO MUNICIPAL

CURITIBA
2016

LUAN GABRIEL DA SILVA
RICARDO KROHN DE JESUS

PLATAFORMA PARA CONTROLE DE ESTACIONAMENTO MUNICIPAL

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná, para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Alessandro Brawerman.

CURITIBA
2016

TERMO DE APROVAÇÃO

LUAN GABRIEL DA SILVA
RICARDO KROHN

PLATAFORMA PARA CONTROLE DE ESTACIONAMENTO MUNICIPAL

Trabalho de Conclusão do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, pela banca examinadora:

_____

Orientador: Prof. Dr. Alessandro Brawerman
Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná, UFPR.

_____

Examinador: Prof. Dr. Mario de Paula Soares Filho
Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná, UFPR.

_____

Examinador: Prof. Dr. Dieval Guizelini
Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná, UFPR.

Curitiba, 08 de dezembro de 2016

AGRADECIMENTOS

Agradecemos aos familiares e amigos por todo o apoio, paciência, incentivo e motivação para que este trabalho tivesse sucesso. Agradecemos a eles também, por colaborar durante todo o processo de formação profissional, que nos fez chegar até aqui.

Agradecemos também ao Bob pelo apoio incondicional e pela confiança.

RESUMO

O presente trabalho refere-se à proposta de desenvolvimento de uma plataforma para controle de estacionamento municipal. A plataforma é formada por dois aplicativos móveis, um para o condutor, tornando mais fácil a compra de crédito para estacionar, alertas configuráveis para lembrar seu tempo de permanência restante e prorrogar o período de estacionamento com um clique. O segundo consiste em uma aplicação voltada para os agentes de trânsito, os quais podem realizar uma rápida consulta e obter a situação dos veículos estacionados em suas proximidades. Este aplicativo apresenta ao agente informações básicas do veículo, do proprietário, tempo de permanência, local estacionado e registro de advertências. Desta forma, espera-se tornar mais fácil e ágil a maneiras como cidadãos e agentes de trânsito lidam com o gerenciamento de estacionamento municipal.

Palavras-chave: Plataforma, aplicativos móveis, condutor, fiscal, estacionar.

ABSTRACT

The presente work conerns about the propose and development of a platform for municipal parking control. The platform is formed by two mobile apps, one for the driver, making it easier to purchase credit to park, configurable alerts to remind their remaining parking time, extending the parking time with one click. The second consists of one application focused for traffic agents, which may do a fast search and get the status of vehicles parked nearby. This application presents to the agent information basic vehicle information, owner information, dwell time, parked location and warning log. Therefore, it is expected to become easier and more responsive to ways citizens and traffic agents deal with municipal parking management.

Palavras-chave: Platform, mobile apps, driver, trafficagents, parking.

LISTA DE FIGURAS

LISTA DE SIGLAS

API	-	Interface de Programação de Aplicativos
ASP	-	Active Server Pages
FCM	-	FirebaseCloudMessaging
FTP	-	File TransferProtocol
GCM	-	Google CloudMessaging
HTTP	-	Hypertext TransferProtocol
JSON	-	JavaScriptObjectNotation
JUDE	-	Java and UML DevelopersEnvironment
OOA	-	ObjectOrientedAnalysis
OOD	-	ObjectOriented Design
OOP	-	ObjectOrientedPrograming
PHP	-	PHP: Hypertext Preprocessor
REST	-	RepresentationalStateTransfer
SQL	-	Structured Query Language
WEB	-	World Wide Web
XML	-	eXtensibleMarkupLanguage

SUMÁRIO

1. INTRODUÇÃO

A criação de novas tecnologias proporcionou um papel importante na evolução da sociedade, permitindo maior praticidade, rapidez e eficiência. Os benefícios alcançam diversos setores, entre eles: medicina, transportes, meios de comunicação, indústria, comércio, financeiro e entretenimento. Com aplicativos para celulares é possível ter rápido acesso à informação, troca de mensagens e mídias, realizar transações bancárias, traçar rotas, acompanhar horário dos ônibus, efetuar compras e pagar contas. O smartphone tornou-se um recurso indispensável na vida contemporânea e, com base nisso, identificar oportunidades se faz necessário para que inovações sejam criadas, acompanhando o avanço tecnológico.

Nosso estudo abrange o estacionamento regulamentado do munícipio de Curitiba/PR. A regulamentação das áreas de estacionamento se deu com o objetivo de democratizar o espaço público da cidade, promovendo a rotatividade no uso das vagas e, ao mesmo tempo, auxiliando na fluidez do tráfego (Setran,2016). Esse serviço possui oportunidade de inovação e criação de tecnologias, pois o trabalho é feito por agentes de trânsito que fiscalizam os cartões em carros estacionados, cujo uso é obrigatório e são adquiridos exclusivamente em lojas lotéricas.

O presente projeto propõe uma plataforma que facilita a comercialização e fiscalização do estacionamento regulamentado através de aplicativos que permitem a compra online de créditos para estacionar, sem a necessidade de obter cartões físicos nas Lotéricas, além da opção aos usuários, de receber lembretes para que não ultrapasse seu tempo limite. Em paralelo, o fiscal poderá consultar a validade dos créditos dos veículos estacionados e enviar notificação aos condutores sobre a autuação aplicada. Desse modo, o serviço se tornaria mais rápido, prático e eficiente, tanto para o condutor, quanto para o fiscal, beneficiando a todos os envolvidos.

1.1. JUSTIFICATIVA

O interesse pelo tema surgiu devido à oportunidade de melhoria identificada no setor de estacionamento regulamentado em Curitiba/PR, uma vez que a fiscalização desse serviço é feita por agentes de trânsito, os quais monitoram veículo a veículo, consultando nos locais onde os carros foram estacionados. Com a proposta da plataforma , é possível consultar os dados dos veículos em tempo real, de qualquer lugar.

O condutor que deseja estacionar nas áreas regulamentadas no sistema atual, precisa comprar cartões físicos exclusivamente nas Lotéricas, o que exige deslocamento e disponibilidade de horário dos locais de venda para a aquisição do cartão. A plataforma digital permite que os créditos para estacionar sejam comprados online, em tempo real, dispensando a dependência das lotéricas e possível deslocamento do condutor.

1.2. OBJETIVOS

Nesta seção são apresentados os objetivos do projeto, divididos em um objetivo geral e diversos objetivos específicos.

1.2.1. Objetivo Geral

Desenvolvimento de uma plataforma digital capaz de inovar e automatizar o serviço de estacionamento regulamento em Curitiba/PR. A partir de uma aplicação de celular, permite que fiscais de trânsito consultem dados dos veículos estacionados e possibilita a compra de créditos para estacionar em locais públicos regulamentados seja realizada online e em tempo real.

1.2.2. Objetivos Específicos

Os objetivos específicos foram propostos para que fosse possível o cumprimento do objetivo geral e consequentemente a finalização do projeto. São eles:

– Proporcionar praticidade na rotina dos agentes de trânsito;

– Desenvolver dois aplicativos Android, um para o agente de trânsito, e um para o condutor de veículos;

– Desenvolver um sistema web central, que interligasse os dois aplicativos Android, armazenando dados de usuários e disponibilizando os mesmos aos agentes;

– Consultar online e em tempo real dados dos veículos estacionados;

– Compra rápida de créditos para o estacionamento regulamentado;

– Redução da emissão de cartões físicos;

– Validação e realização de testes.

2. FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica está centrada em apresentar aspectos teóricos e utilizados para o desenvolvimento do projeto, alem das ferramentas escolhidas.

2.1. ESTAR

O Estar (Estacionamento Regulamentado) foi criado em Curitiba, em 1º de julho de 1980, pela Lei N.º 3979/1971 e regulamentado pelos Decretos N.º 569/1980 e N.º 934/1997. (SETRAN, 2016).

Atualmente o Estar funciona com cartões de papel em blocos onde o motorista, tratado como cliente no trabalho a seguir, compra esses cartões e necessita utilizá-los da maneira correta para que os mesmos continuem validos. É comum vermos problemas com motoristas que raspam o cartão de maneira incorreta e, as vezes até por falta de informação, acabem sendo multados por isso ou acabem tendo que gastar com outro cartão. Cada cartão vale por 1h (uma hora), podendo ser utilizados até 3 (três) cartões seguidos, de forma a permanecer no local até 3 horas.

O fiscal, agente de trânsito, percorre as ruas em locais onde se necessita o Estar, e verifica um por um os cartões dos veículos, multando ou não os mesmos ou seja, não tem a informação prévia de qual veículo está com seu período vencido ou não. Tudo isso é feito manualmente pelos fiscais.

O uso do Estar é essencial em áreas centrais da capital, nas quais a disputa por vagas de estacionamento público é alta e a rotatividade se faz necessária. Pensando nisso, o protótipo vem em boa hora para a facilitação e automação do uso do Estar, tanto pelo motorista, que com este sistema pode de maneira fácil e rápida utilizar o Estar, e também para o fiscal que pode automatizar grande parte do seu serviço.

2.2. WEB SERVICE

Web Service é uma interface projetada para se comunicar via rede. Tipicamente, HTTP é o protocolo mais comumente usado para a comunicação. Web Services também usam SOAP, REST e XML-RPC como meio de comunicação. Ou seja, quando uma API precisa enviar dados através de rede, estamos falando de Web Services (COSTA,2015).

No trabalho em questão foi utilizado o protocolo REST pelo fato de ser compatível com o modelo JSON, o que facilita a troca de informações e é muito mais leve do que os demais. Os web services foram criados utilizando a linguagem de programação PHP.

2.3. SOAP

SOAP (Simple Object Access Protocol) é um protocolo de transferência de mensagens em formato XML para uso em ambientes distribuídos. O padrão SOAP funciona como um tipo de framework que permite a interoperabilidade entre diversas plataformas com mensagens personalizadas. Uma das grandes qualidades desse protocolo é sua independência de plataforma e linguagem além de ser simples e extensível por utilizar XML (ZARELLI, 2012).

2.4. REST

“REST, é uma abstração da arquitetura da web, um estilo arquitetural que consiste de um conjunto coordenado de restrições arquiteturais aplicadas a componentes. REST ignora os detalhes da implementação de componente e a sintaxe de protocolo com o objetivo de focar nos papéis dos componentes, nas restrições sobre sua interação com outros componentes e na sua interpretação de elementos de dados significantes.” (SAUDATE, 2014).

A Figura 1 mostra o processo genérico de requisição do aplicativo Cliente com o servidor, onde a aplicação faz uma requisição para a API (Web service), essa API realiza uma consulta SQL na base de dados do servidor e retorna o resultado em JSON para a aplicação.

Resumidamente, o protocolo REST permite que o desenvolvedor escolha qual o formato de mensagens pode utilizar, sendo JSON o mais comum entre eles. Isso faz com que ele seja mais rápido e mais leves, porém, como é aberto à vontade do desenvolvedor, é comum ocorrer problemas de interoperabilidade.

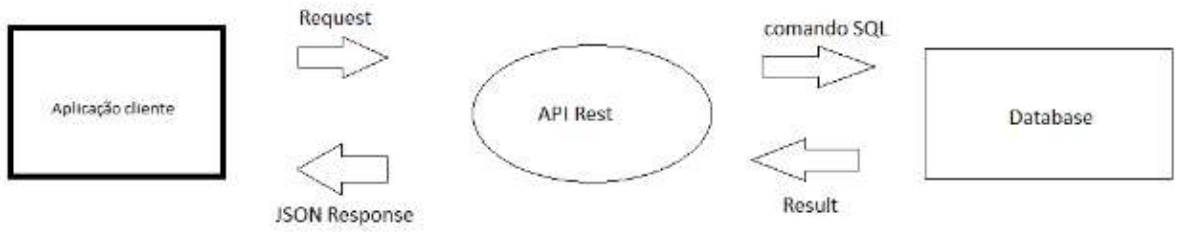


FIGURA 1- PROCESSO GENÉTICO DE REQUISIÇAO REST

2.5. JSON

JSON nada mais é do que um modelo de transmissão de dados que vem se tornando o mais popular para troca de informações entre aplicações. É bastante utilizado pois, ao contrário do XML, suas informações são melhor estruturadas e compactas tendo um melhor desempenho na hora de converter os dados (CORRÊA, 2016).

Na Figura 2 é possível ver um exemplo de código em JSON, usado na aplicação, desenvolvida neste projeto, para o Fiscal. Este array de objetos JSON retorna todos os períodos ativos em um raio de distância pré-determinado. É possível notar como é distribuído sua estrutura de atributos, valores e objetos.

```
{
"content": [{
        "idEstar": "120",
        "placa": "ABC1234",
        "inicio": "03/10/2016 16:30:54",
        "horas": "2",
        "alerta": null,
        "Usuario_id": "18",
        "valor": null,
        "latitude": "-25.432854",
        "longitude": "-49.2742094",
        "Endereco_idEndereco": "0",
        "numero": null,
        "address": "Rua Voluntários da Pátria, 250 - Centro, Curitiba - PR",
        "situacao": "0",
        "diff": "103514",
        "vencido": "1",
        "distancia": "0.24124530587047954"
}]
}
```


FIGURA 2- EXEMPLO DE CÓDIGO EM JSON.

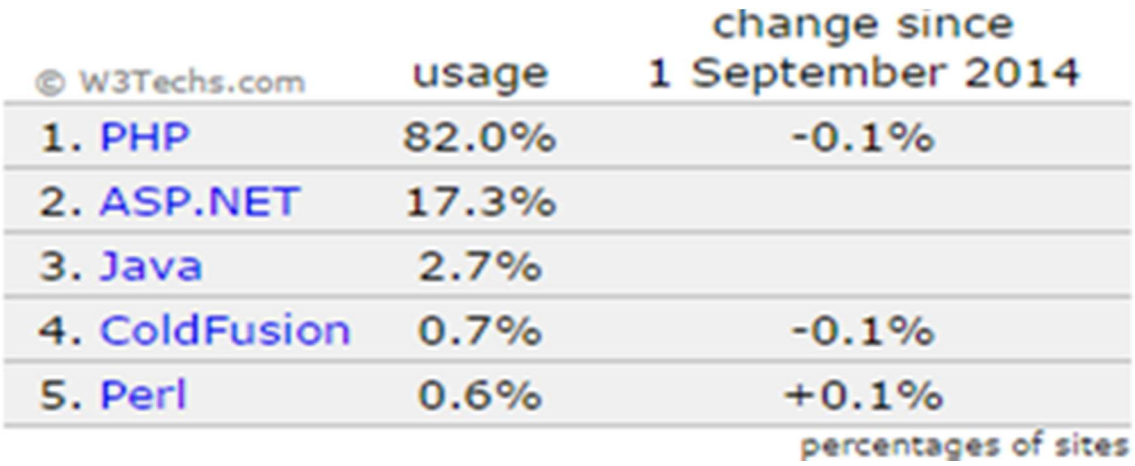
2.6. XML

XML é uma recomendação para gerar linguagens de marcação para necessidades especiais. XML é capaz de descrever diversos tipos de dados organizados hierarquicamente, e seu objetivo principal é a facilidade de compartilhamento de informações através da Internet. A linguagem XML é classificada como extensível porque permite definir os elementos de marcação (FERREIRA, 2009).

2.7. PHP

A linguagem de programação PHP é uma linguagem usada originalmente para aplicações atuando no lado do servidor. Por isso, seu código é interpretado ao lado do servidor pelo módulo PHP, diferente de linguagens como JavaScript, por exemplo. Tem suporte a protocolos como HTTP, SOAP, entre outros.

Atualmente, a ferramenta é uma das lideres de mercado, pelo fato de ser de fácil entendimento e de desenvolvimento ágil. Esses fatores fizeram as empresas utilizarem do PHP em seus servidores, fato que apresenta a Figura 3 mostrando a grande dominância da linguagem no mercado. Alguns grandes sites adotaram a ferramenta, como Facebook e Yahoo!



© W3Techs.com	usage	change since 1 September 2014
1. PHP	82.0%	-0.1%
2. ASP.NET	17.3%	
3. Java	2.7%	
4. ColdFusion	0.7%	-0.1%
5. Perl	0.6%	+0.1%

percentages of sites

FIGURA 3 - USO DE LINGUAGENS DE SERVIDOR NA WEB

FONTE: W3TECHS.COM

O PHP foi escolhida para a criação das API's por ser de fácil entendimento, rápido desenvolvimento e ser suportada pelo servidor usado para armazenar os dados das aplicações. Para o desenvolvimento, foi utilizado a IDE PHPStorm, da IntelliJ.

2.8. ANDROID

Android é um sistema operacional para dispositivos móveis baseado em plataforma de código aberto, Linux, e é permitido que os fabricantes possam modificá-lo e adicionar novos recursos, fazendo o software evoluir constantemente devido a isso.

Internamente suporta as seguintes funcionalidades:

- Armazenamento – usa o SQlite uma base de dados relacional leve, para armazenamento de dados;
- Conectividade – suporta WIFI, Bluetooth, GSMe outras;
- Troca de mensagens – suporta tanto SMS quanto MMS;
- Navegador Web – baseado no WebKit de código aberto, com JavaScript V8 do chrome;
- Suporte a meios – inclui MP3, MP4, MIDI, JPEG, ETC;
- Suporte a hardware – sensor de acelerômetro, câmara bússola digital, sensor de proximidade;
- Multitoques – suporte a telas multitoques;
- Multitarefa – suporte aplicativo multitarefa;
- Tethering – suporte ao compartimento de conexões de Internet como hotspot com e sem fios.

(LEE,2011).

2.8.1. Arquitetura

A arquitetura do Android é dividida em camadas, onde:

- Camada zero: Fica o Kernel, o cérebro do sistema operacional. É ele que faz a ligação entre o software e o hardware.
- Camada um: Nessa camada se encontram as bibliotecas e o *RunTime*. É nessa camada em que o desenvolvedor vai aplicar seu código.
- Camada dois: É a camada de framework de aplicação, onde ficam os programas que gerenciam as aplicações básicas como gerenciamento de telefone e de notificações.
- Camada três: É onde ficam as aplicações e onde ocorre a interação com o usuário do sistema.

(LECHETA,2010).

A Figura 4 mostra tudo isso em uma imagem, ilustrando todas as camadas e seus programas pertencentes a ela.

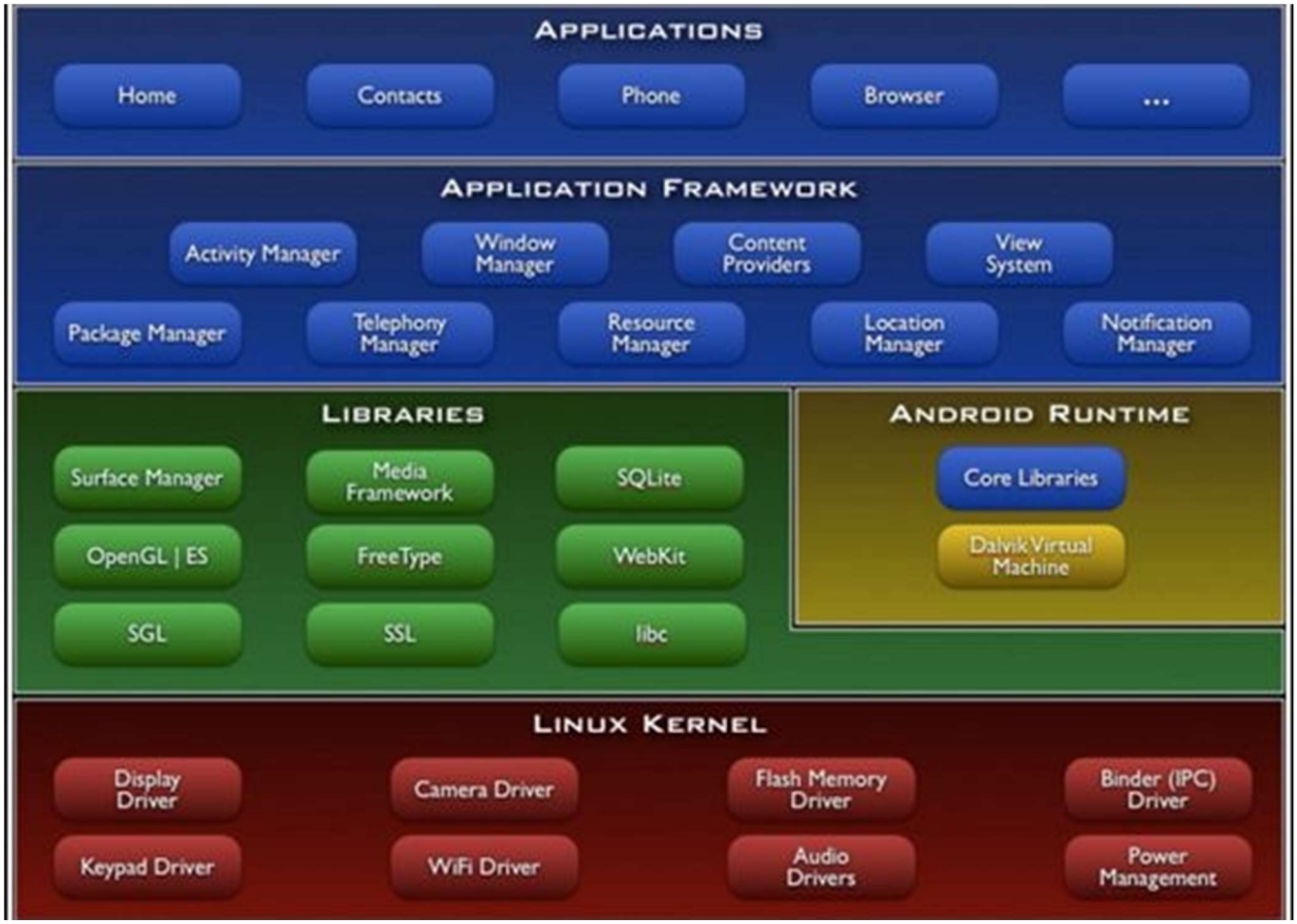


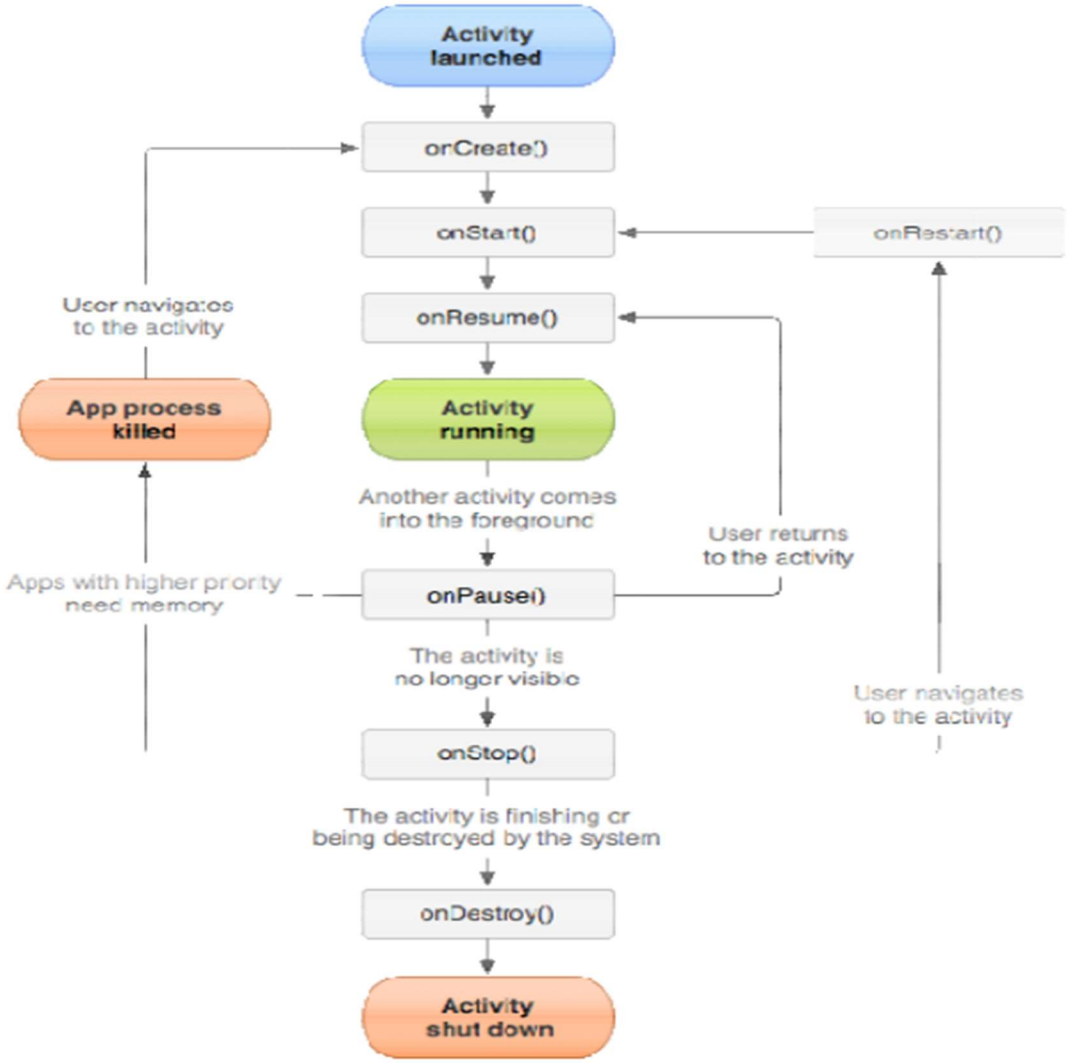
FIGURA 4- ARQUITETURA DO ANDROID

FONTE: LECHETA, 2010

2.8.2. Activity

A activity é uma janela que contem a interface do seu aplicativo. Todo aplicativo começa por uma activity, e é nela que o usuário interage com o seu sistema (LECHETA, 2010).

A activity tem vários estágios durante sua execução, chamado de ciclo de vida de uma atividade, demonstrado na Figura 5.



Fonte: http://developer.android.com/about/versions/index.html

FIGURA 5- CICLO DE VIDA DE UMA ACTIVITY

3. METODOLOGIA

Neste capítulo é fornecida uma visão geral do projeto e a metodologia usada para o desenvolvimento do trabalho.

3.1. VISÃO GERAL

As primeiras etapas do projeto envolveram pesquisas sobre o funcionamento e regulamentação do EstaR de Curitiba/PR, levantamento de requisitos e funcionalidades do sistema.

Inicialmente, pesquisou-se sobre a legislação e funcionamento atual sistema de controle do EstaR, que já está há anos implantado na cidade e foi proposto adequar o sistema ao costume já aderido pelos usuários do Estar. De inicio, foi criado um *mapa mental* (APÊNDICE 1) para ter em mente o escopo inicial.

Com isso, tiveram-se materiais necessários para o levantamento de requisitos e um modelo com as principais funcionalidades do sistema. Ao imaginar um esqueleto para o trabalho, foi feito um protótipo de baixa fidelidade, ilustrando como seria o fluxo do aplicativo do cliente (APÊNDICE 2) e o fluxo também do aplicativo do fiscal (APÊNDICE 3).

No passo seguinte, foi criado um artefato *User Stories* para cada um dos aplicativos, onde foi detalhado cada funcionalidade necessária do sistema. Tendo em vista esse artefato, foi possível modelar o banco de dados e criar todos os diagramas UML. Diagrama de casos de uso também foi criado também foi criado baseado no artefato de *User Stories*.

Assim, tendo o artefato de *User Stories*, os diagramas UML, a base de dados criada, e todo o fluxo dos aplicativos desenhados em baixa fidelidade, foi dado inicio ao desenvolvimento do sistema. Iniciando-se pela parte de protótipo navegável.

A metodologia adotada no projeto, baseada em partes no modelo em cascata e em parte na metodologia ágil, estipulava que as etapas de desenvolvimento se dariam em entregas semanais, nas quais o professor orientador pediria um conjunto de funcionalidades e estipularia uma semana para a entrega. Na primeira semana, por exemplo, ficou estipulado usar a API de localização do Android e a API de mapas do Google Maps para buscar o endereço atual do usuário. Na semana 2, fazer um login com autenticação no servidor, e assim por diante. Lembrando que, o projeto se deu início com o aplicativo do cliente, para depois que finalizado, fosse então desenvolvido o aplicativo do fiscal.

Com base nas entregas semanais, foram criados também os *Web Services* do sistema, pois de acordo com a necessidade da entrega era essencial a comunicação com o servidor. Nessa altura do projeto, o objetivo já era finalizar um protótipo funcional.

O curso do projeto seguiu nesse padrão, o professor orientador traçava a meta da próxima entrega, que durava sempre 1 semana, e discutia com a equipe qual era a melhor forma de implementar tal função, caso que ocorreu no desenvolvimento de ambos os aplicativos.

3.2. MODELO DE PROCESSO UTILIZADO

Levando em consideração a metodologia utilizada pela equipe para desenvolvimento desse trabalho, o modelo de processo utilizado foi o Modelo Evolucionário de Prototipação, levando em conta que não se tinha certeza se as bibliotecas usadas iriam suprir nossa necessidade de negócio, e também se a ideia que se tinha de interação entre usuário e sistema era adequada.

A prototipação se dava basicamente com uma reunião semanal, na qual o professor orientador requisitava funcionalidades para a equipe desenvolver. Visto isso, a iteração era planejada rapidamente, que levava a um projeto rápido. Esse projeto rápido leva a construção do protótipo, que retornava ao professor orientador para que o mesmo desse seu feedback. Esse loop iria ocorrendo conforme o protótipo atendesse as necessidades do usuário.

A Figura 6 mostra as etapas do modelo utilizado. Nela, é ilustrado as iterações do modelo.

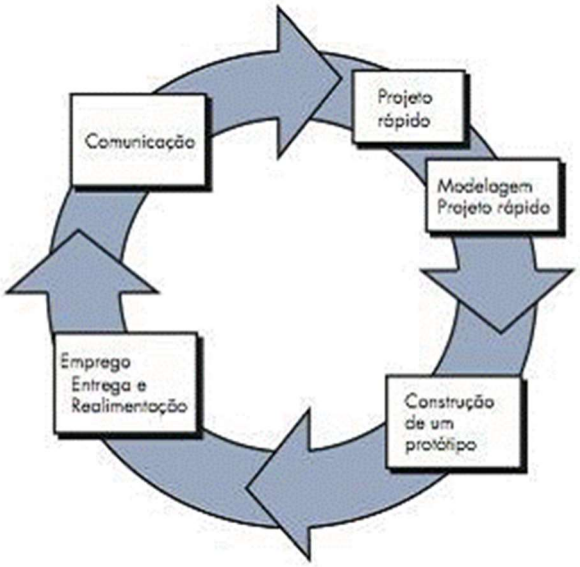


FIGURA 6- ETAPAS DA PROTOTIPAÇÃO.

3.3. GERENCIAMENTO DE RISCOS

O risco em um projeto de software é uma medida da probabilidade e da perdarelacionadas à ocorrência de um evento negativo que afete o próprio projeto, seuprocesso ou o seu produto. Em outras palavras, qualquer coisa que possa acontecer eameaçar o bom andamento do projeto é um risco (ISLIG-RIO,2016).

A (TABELA 1) indica os riscos identificados no projeto e a analise qualitativa dos mesmos.

CONDIÇÃO	CONSEQUENCIA	AÇÃO	PROBABILIDADE	IMPACTO	CLASSIFICAÇÃO
Dificuldade com hardware que seja compatível com o Android Studio	Não conseguir tempo suficiente para desenvolvimento.	Verificar a possibilidade de usar outra IDE.	Moderado	Alto	8
Não cumprimento de prazos.	Atraso em entregas.	Focar nas partes mais complicadas.	Moderado	Alto	7
Problemas com tecnologias do Android	Não conseguir reproduzir a regra de negócio do aplicativo.	Procurar soluções equivalentes, ou parecidas.	Alta	Alto	8
Problemas com versões do Android	Tecnologias não serem compatíveis com todas as versões do Android.	Procurar atender as versões mais recentes.	Alta	Médio	6
Problemas nas especificações	Retrabalho.	Revisar sempre a especificação.	Moderado	Médio	6

Tabela 1 – Plano de riscos.

3.4. RESPONSABILIDADES

A divisão de responsabilidades, mostrada na tabela abaixo (TABELA 2) foi realizada de acordo com os conhecimentos de cada integrante da equipe, focando em um desenvolvimento de um produto de qualidade e em menor tempo, respeitando o cronograma inicial.

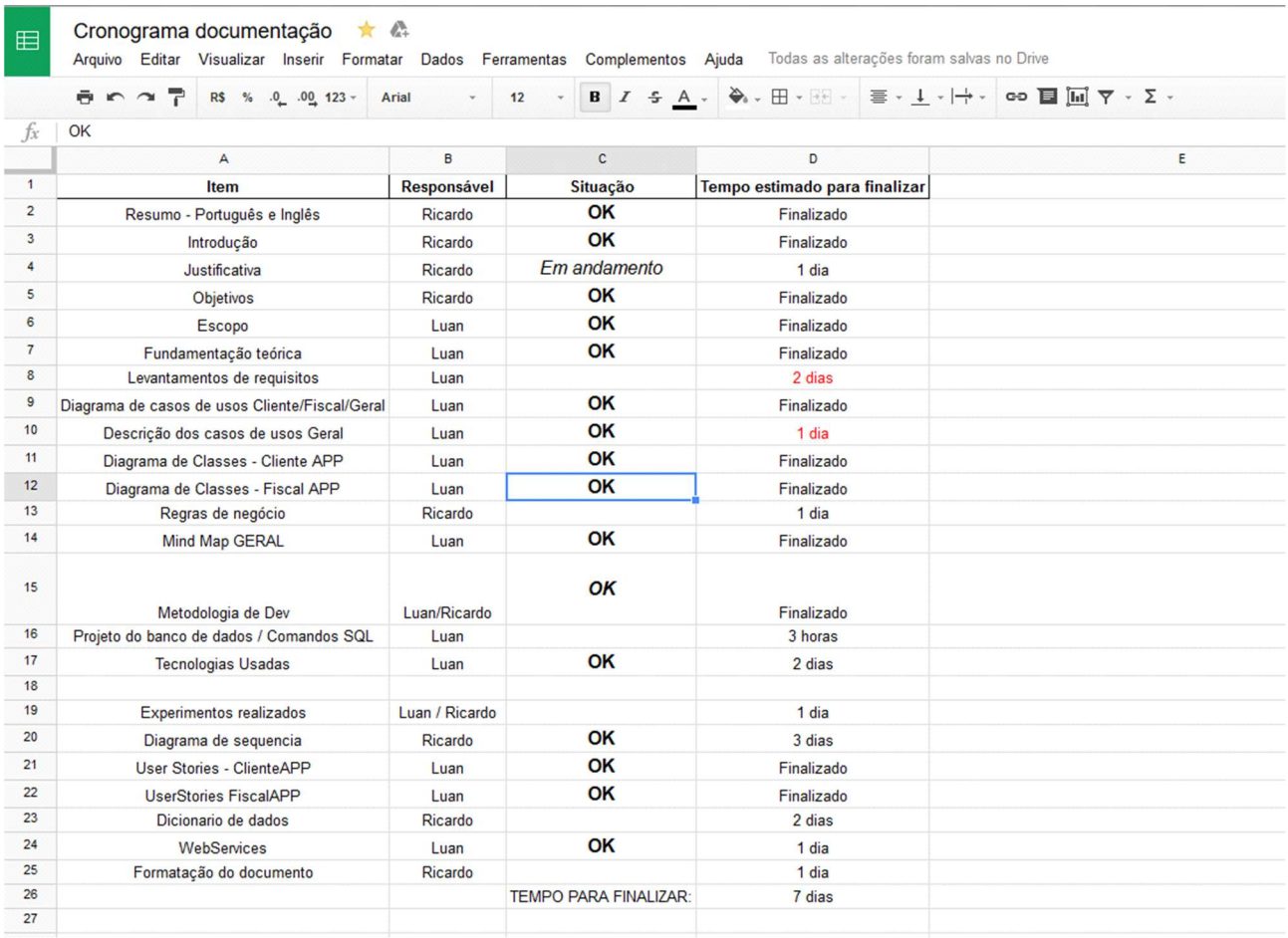
ATIVIDADE	RESPONSÁVEL
Reunião inicial	Luan, Ricardo
Definição de tema	Luan, Ricardo
Definição de requisitos	Luan, Ricardo
Mapa mental	Luan
Divisão de responsabilidades	Luan
Desenvolvimento do modelo de banco de dados	Ricardo

Diagrama de classes	Ricardo
Desenvolvimento das aplicações	Luan
Testes	Ricardo
Documentação	Ricardo
Bug Fixes	Luan
Teste de correções	Ricardo
Revisão de documentação	Luan, Ricardo

3.5. CRONOGRAMA

O cronograma foi criado em uma planilha no Google Drive compartilhado com a equipe. Nele, é detalhado a tarefa de cada integrante, o responsável pela tarefa, a situação que se encontra (Se finalizada ou em andamento) e o tempo estimado para finalizar a mesma, levando em conta as habilidades do integrante e a complexidade da tarefa a ser realizada.

A Figura 7 apresenta esse cronograma e todos os detalhes acima citados.



	A	B	C	D	E
1	**Item**	**Responsável**	**Situação**	**Tempo estimado para finalizar**	
2	Resumo - Português e Inglês	Ricardo	**OK**	Finalizado	
3	Introdução	Ricardo	**OK**	Finalizado	
4	Justificativa	Ricardo	*Em andamento*	1 dia	
5	Objetivos	Ricardo	**OK**	Finalizado	
6	Escopo	Luan	**OK**	Finalizado	
7	Fundamentação teórica	Luan	**OK**	Finalizado	
8	Levantamentos de requisitos	Luan		2 dias	
9	Diagrama de casos de usos Cliente/Fiscal/Geral	Luan	**OK**	Finalizado	
10	Descrição dos casos de usos Geral	Luan	**OK**	1 dia	
11	Diagrama de Classes - Cliente APP	Luan	**OK**	Finalizado	
12	Diagrama de Classes - Fiscal APP	Luan	**OK**	Finalizado	
13	Regras de negócio	Ricardo		1 dia	
14	Mind Map GERAL	Luan	**OK**	Finalizado	
15	Metodologia de Dev	Luan/Ricardo	***OK***	Finalizado	
16	Projeto do banco de dados / Comandos SQL	Luan		3 horas	
17	Tecnologias Usadas	Luan	**OK**	2 dias	
18					
19	Experimentos realizados	Luan / Ricardo		1 dia	
20	Diagrama de sequencia	Ricardo	**OK**	3 dias	
21	User Stories - ClienteAPP	Luan	**OK**	Finalizado	
22	UserStories FiscalAPP	Luan	**OK**	Finalizado	
23	Dicionario de dados	Ricardo		2 dias	
24	WebServices	Luan	**OK**	1 dia	
25	Formatação do documento	Ricardo		1 dia	
26			TEMPO PARA FINALIZAR:	7 dias	
27					

FIGURA 7- CRONOGRAMA DE DESENVOLVIMENTO

3.6. ANÁLISE DE REQUISITOS

A análise de requisitos é um processo que envolve o estudo das necessidades do usuário para se encontrar uma definição correta ou completa do sistema ou requisito de software (IEEE, 1990).

Através da analise de requisitos foram definidos os requisitos que são requisitos funcionais ou não funcionais. Sendo que os requisitos funcionais são aqueles que definem características e segurança e não funcionais aqueles que afetam questões como desempenho, interface, usabilidade.

3.6.1. REQUISITOS FUNCIONAIS DO CLIENTE

A seguir são listados os requisitos funcionais da aplicação do cliente:

- Para utilizar o sistema, o usuário deverá ter cadastro.
- Capturar sua localização mais precisa possível.
- Ter conta na Google, para realizar os pagamentos via Google Wallet.
- Autenticação de e-mail e senha.
- Sincronização de saldo do usuário.
- Contador com o tempo restante do período ativo.
- Notificação quando o período estiver vencendo e/ou ao vencer.
- Executar serviços em *background*.

3.6.2. REQUISITOS FUNCIONAIS DO FISCAL

A seguir estão os requisitos funcionais da aplicação do fiscal.

- Listar todos os períodos, veículos estacionados, em um raio de 1km.
- Mostrar o local em um mapa do período escolhido.
- Filtros de acordo com o desejo do usuário.
- Detalhar pela cor se um período está ativo ou não.
- Notificar um cliente por *push*.

3.6.3. REQUISITOS NÃO FUNCIONAIS DO CLIENTE

A seguir são listados os requisitos não funcionais da aplicação do cliente.

- Persistência de dados do usuário, como o manter logado sempre que desejar.
- Notificação com som de alerta.
- Interface intuitiva.
- Tela principal dividida em tabs.
- Contador e notificações sobre o tempo sendo locais, não precisando de acesso a internet.

3.6.4. REQUISITOS NÃO FUNCIONAIS DO FISCAL

A seguir são listados os requisitos não funcionais da aplicação do fiscal:

- Não ser necessária autenticação.
- OCR com *pattern* para placa de veículos para pesquisar uma placa de automóvel.
- Retirar um determinado período da lista principal.

4. DESENVOLVIMENTO DO PROJETO

A plataforma para controle de estacionamento municipal tem como objetivo automatizar o serviço de estacionamento regulamento em Curitiba/PR. A partir de uma aplicação de celular, permite que fiscais de trânsito consultem dados dos veículos estacionados e possibilita a compra de créditos para estacionar em locais públicos regulamentados seja realizada online e em tempo real.

São disponibilizados dois aplicativos para Android, um para cliente que deseja utilizar do estacionamento municipal, e um para o agente de trânsito que irá gerenciar os estacionamentos.

4.1. VISÃO GERAL

Para que o dois aplicativos compartilhem dados e possam “conversar” entre eles, é utilizado *web service.* Com isso, o aplicativo envia dados que são salvos no servidor, e recupera esses dados para exibi-los ao usuário. Essa estrutura é utilizada para que os dados sejam atualizados sempre em tempo real e para que não tenha divergência de informações.

A Figura 8 exibe a arquitetura utilizada pelo sistema, onde as requisições POST são os dados enviados do celular para o servidor, e as requisições GET são os dados vindos do servidor para o celular.

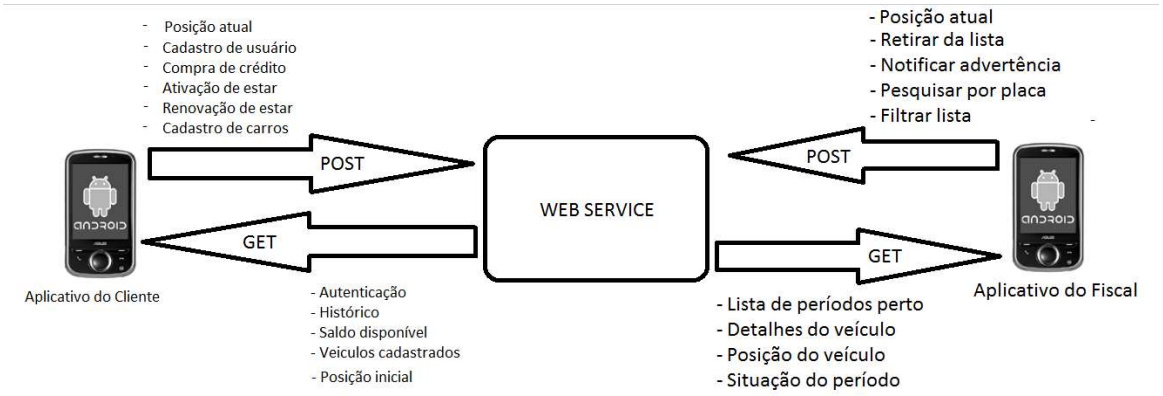


FIGURA 8- ARQUITETURA DO PROJETO.

4.2. TECNOLOGIA E APLICAÇÕES UTILIZADAS

As ferramentas e aplicações utilizadas para a realização deste trabalho foram ASTAH Community para a construção de diagramas, a linguagem JAVA para Android como linguagem de programação das aplicações, a IDE Android Studio para o desenvolvimento, o servidor HOSTINGER para hospedagem de arquivos, o

PHPMYADMIN como gerenciador de banco de dados, e a linguagem PHP como linguagem de desenvolvimento dos *web services*.

A escolha da aplicação ASTAH Community se deu ao fato de ser uma ferramenta de fácil uso. Outro fator importante foi pela gratuidade. Caso parecido com o servidor *HOSTINGER*, que foi escolhido por ser o melhor servidor gratuito disponível no mercado. Com ele, foi possível hospedar os *web services* em PHP e montar o banco de dados. Foi avaliado também a quantidade de requisições gratuitas eram possíveis fazer, tendo assim disponibilidade para ser aplicado no trabalho.

4.3. ASTAH COMMUNITY

A ferramenta *ASTAH* usada no projeto para criar diagramas para a melhor visualização do sistema como um todo. Astah, anteriormente chamado *JUDE*, é uma IDE em java para modelagem de dados. É de fácil uso e ainda possibilita o *Java ReverseCode*, ou seja, a importação de um código java para criar um modelo. Também possui o *Java Forward* que é a criação do código java a partir do modelo.

Na Figura 9 é apresentada a tela do software Astah community, onde é possível visualizar seu organizador de projeto, sua área de visão de propriedades e sua área de edição de diagramas.

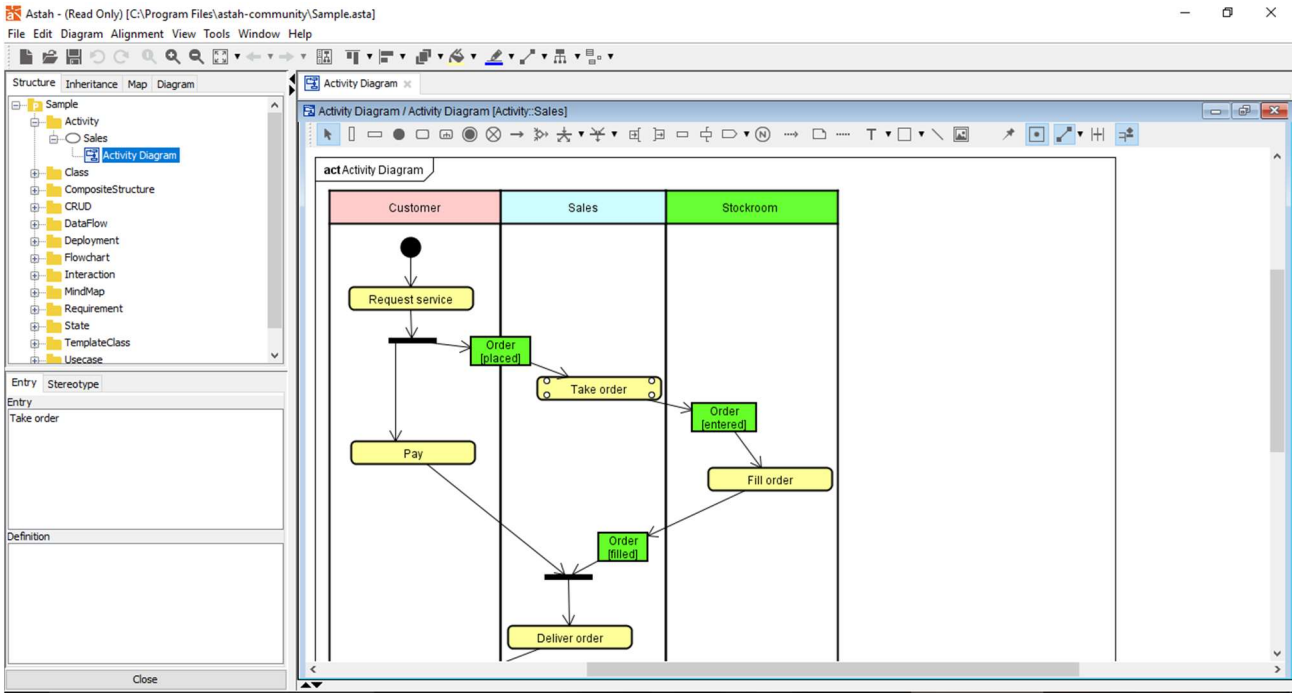


FIGURA 9- TELA DO SOFTWARE ASTAH COMMUNITY

4.4. JAVA

Java é uma linguagem de programação orientada a objetos, compilada para um bytecode e executada em uma máquina virtual, o que a difere das demais linguagens e o que a torna mais pesada também (CAELUM, 2014). Foi a linguagem escolhida, pois é a linguagem nativa para desenvolvimento Android.

Para desenvolver foi utilizada a IDE Android Studio, software baseado na *IntelliJ IDE* da *JetBrains.* Atualmente, se encontra na versão 2 e é a IDE primária do Google para desenvolvimento Android. É um software extremamente pesado, e seus requerimentos são bastante altos. Na Figura 10 é possível ver a tela da IDE, mostrando seu organizador de projetos, e sua área de edição de texto.

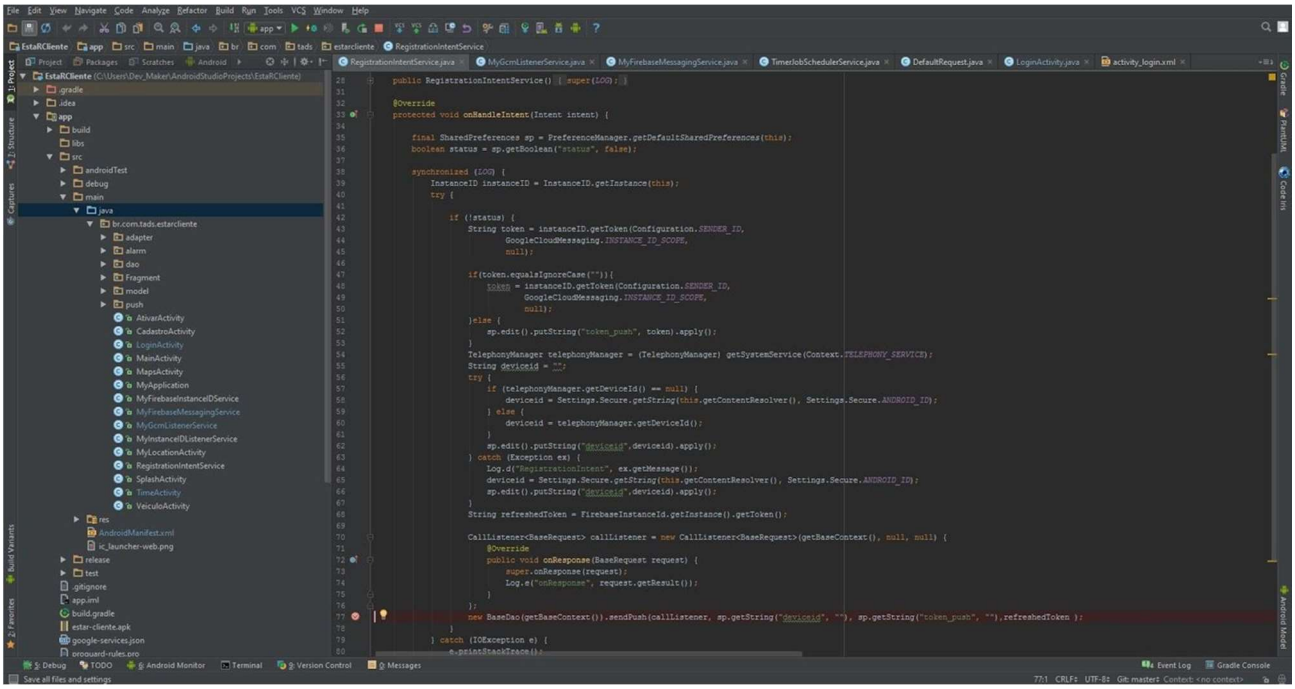


FIGURA 10 - TELA PRINCIPAL DA IDE ANDROID STUDIO.

4.5. HOSTINGER

Hostinger é um sistema de hospedagem, onde é possível registrar um subdomínio gratuitamente. Possui servidor HTTP Apache banco de dados MYSQL e suporta linguagens de programação como PHP, ASP, entre outros. Nele foi possível armazenar os arquivos de *web service* das aplicações deste trabalho, via *FTP*. Na sua versão gratuita, é possível ter acesso a todas as opções essenciais, mas com uma limitação de requisições.

Na Figura 11 é possível visualizar uma parte da tela de dashboard do site, onde se pode gerenciar banco de dados, transferência de arquivos, número de acessos, IP e entre outros.

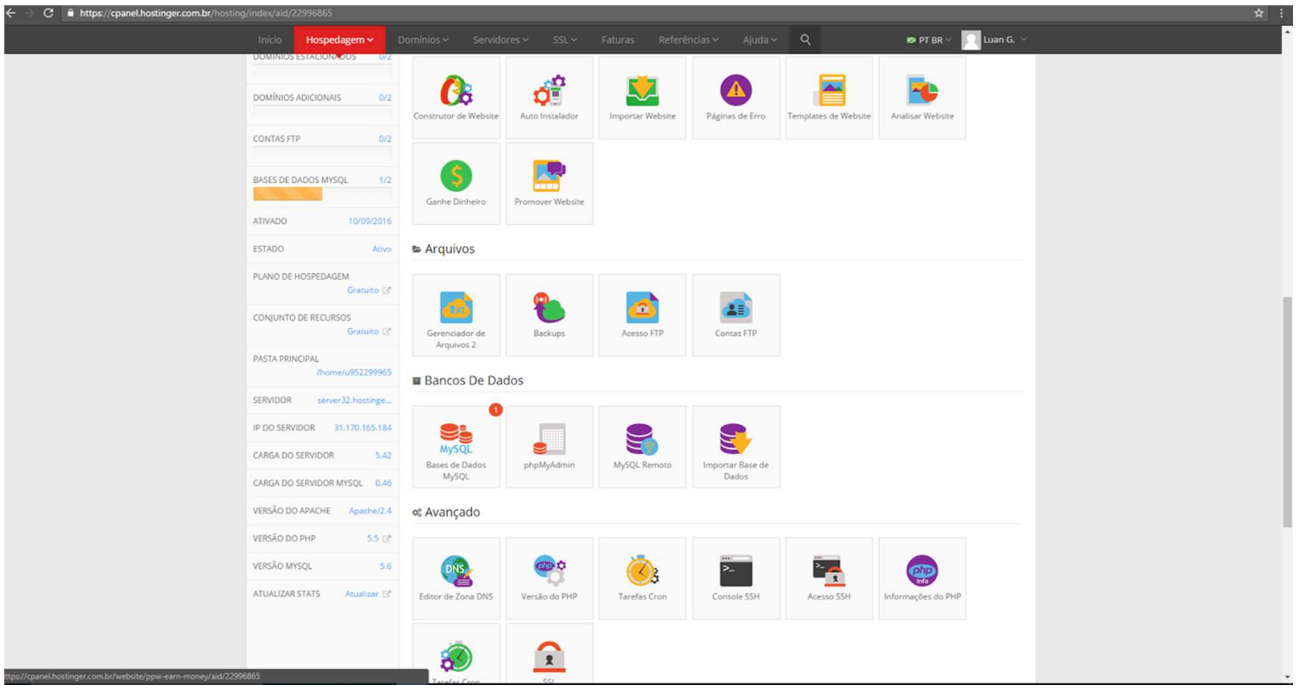


FIGURA 11- DASHBOARD DO SITE HOSTINGER

4.6. PHPMYADMIN

O phpMyAdmin é um aplicativo web desenvolvido na linguagem PHP com o objetivo de administração do MySQL pela internet. Nele é possível gerenciar e manter tabelas, executar comandos *SQL*, entre outros. Na Figura 12 é apresentado à tela inicial do sistema web.

Nele foi construído o modelo físico do banco de dados do sistema, onde serão armazenados os dados relativos aos aplicativos.

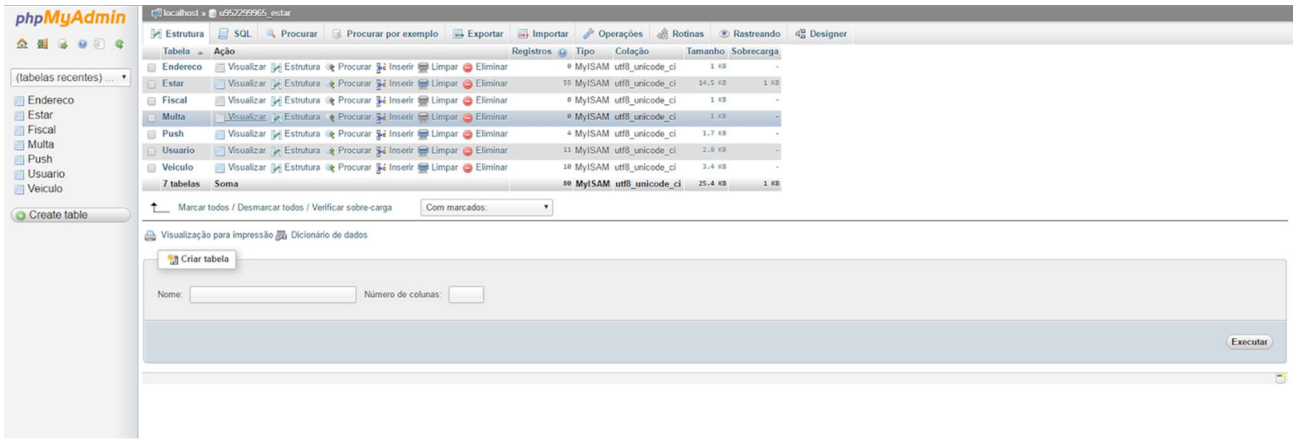


FIGURA 12- TELA INICIAL PHPMYADMIN.

4.7. GOOGLE MAPS

Google Maps é um serviço do Google para visualização e pesquisa de mapas gratuitos. Nele é possível visualizar mapas de cidades, rotas, imagens de satélite e etc. É mantido pela Google e atualizado.

O sistema operacional Android possui integração com a API do Google Maps nativa. Apesar de não poder alterar sua interface ou sua camada gráfica de mapas, o sistema operacional oferece opções interessantes para manejar recursos para o usuário. Outra vantagem do Android é que é extremamente fácil utilizar dos serviços do Google Maps, sendo gerado até automaticamente se utilizado o Android Studio como IDE.

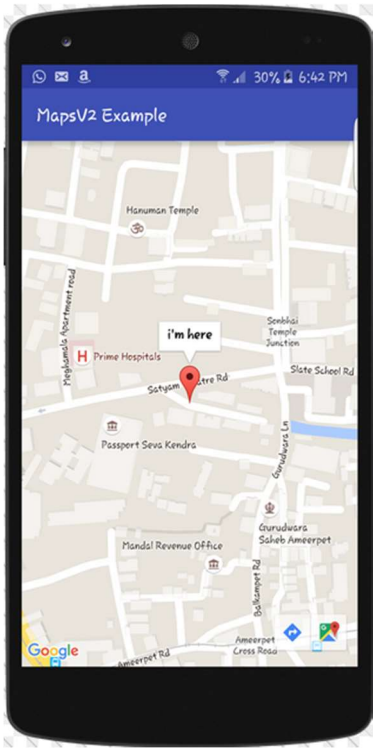


FIGURA 13 -EXEMPLO DE APLICAÇÃO COM GOOGLE MAPS NO ANDROID.

4.8. OBTENÇÃO DE *API KEY* PARA GOOGLE MAPS

Para desenvolvedores conseguirem trabalhar com o Google Maps em sua aplicação, é necessário primeiro criar uma *API KEY* para utilizar esses e alguns outros serviços do Google.

A documentação referente à obtenção da chave para utilização do Google Maps API, está disponível em Google Developers.

A chave pode ser assinada com o *fingerprint,* certificado digital,onde apenas a sua aplicação pode utilizar do serviço, ou pode ser pública, como no case desse trabalho, onde várias aplicações que tiverem a *apikey* podem utilizar o serviço da Google.

4.9. FIREBASE CLOUD MESSAGING

O FCM é um serviço gratuito de mensagens *push* entre plataformas. É a versão mais nova que tomou lugar do antigo GCM e também é mantido pela Google. Diferente do GCM, o FCM é um serviço *cross-plataforma,* ou seja, funciona com apps Android, iOs, Web e Chrome.

O grande diferencial do FCM é a versatilidade de envio de mensagens, onde você tem até três maneiras de envio, o tamanho de dados que você pode enviar que conseguem chegar até a 4KB e o envio de mensagens do cliente para o servidor, indispensável para plataformas de chat, por exemplo.

Para utilizar do serviço, basta criar um projeto no sistema do Firebase e seguir os passos descritos na documentação, que pode ser encontrada aqui:

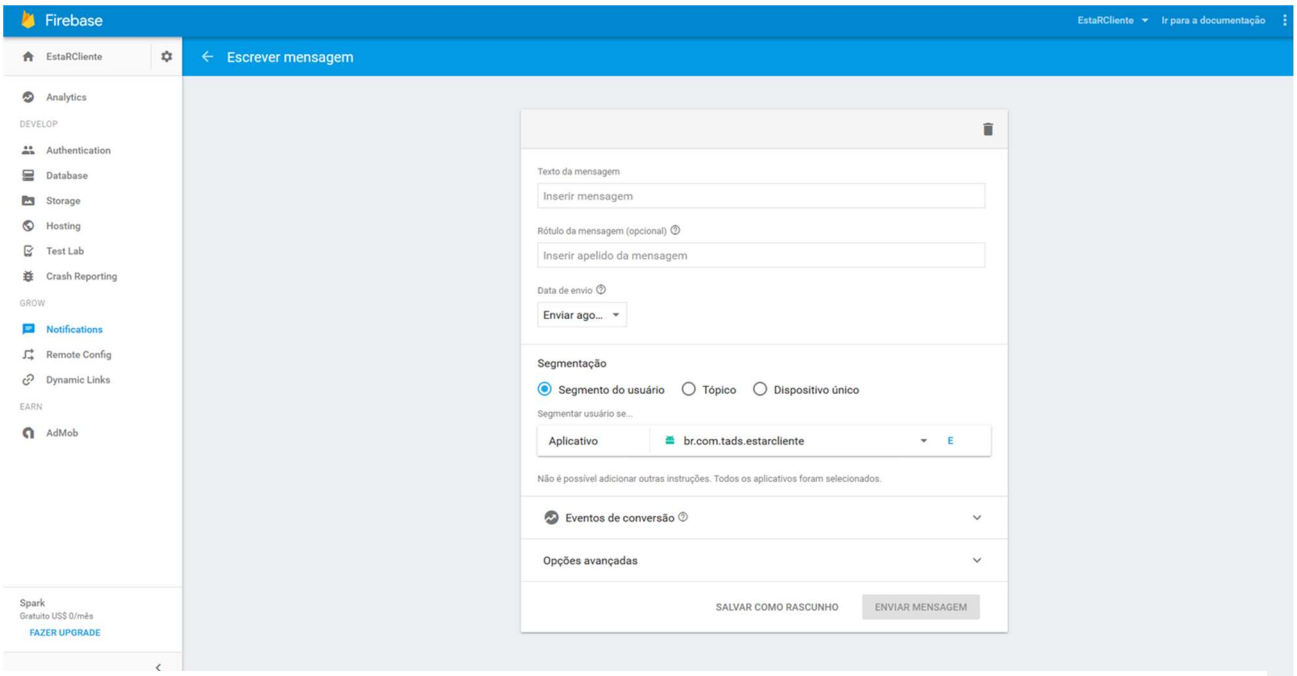


FIGURA 14 - CONSOLE DE ENVIO DE NOTIFICAÇÕES DO FIREBASE.

4.10. USER STORIES

User stories são artefatos de desenvolvimento, bastante similares com casos de usos, que foca no objetivo do usuário e como o sistema realiza para alcançar esses tais objetivos. É escrito segundo a visão do usuário para com o sistema, sendo sempre curtos e claros. É importante especificar o ator, a funcionalidade e a ação requerida (PRIMO, 2011).

As user stories referentes a este projeto encontram-se no (APÊNDICE 4) e (APÊNDICE 5) , como exemplo é apresentada a user storie de LOGIN do condutor e a user storie INICIAL do fiscal.

4.11. USER STORIES CLIENTE

LOGIN

Como cliente, quero acessar o aplicativo por meio de uma tela de login simplificada, em que eu possa digitar os meus dados de acesso, para que eu não tenha que criar múltiplos usuários de senhas para acompanhamento do meu Estar. Gostaria também de não precisar digitar meus dados toda vez que for acessar o aplicativo.

1 A estrutura do login é composta por:
   - 1.1 E-mail.
   - 1.2 Senha.
   - 1.3 Login com o Google.
2 O e-mail de acesso deve ser o e-mail cadastrado pelo usuário.
3 O aplicativo deve guardar os dados do login para permitir a entrada sem a necessidade de autenticação, caso seja requerido pelo usuário no item *manter logado*.
4 O login com Google é feito automaticamente, escolhendo a conta desejada. Caso não faça o *logout*, sempre manterá logado.

4.12. USER STORIES FISCAL

INICIAL

Como fiscal, quero ter acesso a uma lista de estar's ativos e vencidos perto de mim. Também quero que seja possível utilizar de filtros na lista, para que eu possa visualizar períodos de acordo com meu desejo.

Quero também que a lista seja atualizada automaticamente a cada 30 segundos, sem a necessidade de ação do usuário. Deve também haver a opção de atualização com *swipe* da lista.

1. A estrutura do card de Estar é composta por:
   1.1. Endereço.
   1.2. Placa.
   1.3. Tempo restante, ou vencido.
2. O período que estiver vencido, deve ter seu *background* da cor vermelha.
   Enquanto o ativo deve conter seu *background* da cor verde.
3. Os filtros devem ser de:
   3.1. Ativos.
   3.2. Vencidos.
   3.3. Mais perto.
   3.4. Horário de ativação.

4.13. DIAGRAMA DE SEQUÊNCIA

Um diagrama de seqüência representa a seqüência de processos, ou seja, mensagens passadas entre objetos. Basicamente, esse diagrama mostra a comunicação entre objetos de um cenário, realizados por *métodos* ou funções onde é dada ênfase à ordenação temporal entre a troca de mensagens entre um objeto e outro (Martinez, 2015).

O Diagrama de Seqüência (APÊNDICE 8) representa o fluxo de dados de um evento no sistema, iniciando do ator, representando o usuário, até a inserção ou busca de alguma informação presente no servidor. Foi criado um diagrama de seqüência para cada funcionalidade importante no sistema.

4.13.1. Diagramas de Seqüência do Aplicativo do Cliente

A descrição do diagrama de sequência de Login do condutor é mostrado como exemplo, todos as outros descrições de diagramas encontram-se no (APÊNDICE 9).

Login (APÊNDICE 8.1): Esse diagrama define o funcionamento da autenticação do sistema no aplicativo do cliente. (1) O usuário abre o aplicativo. (2) O sistema apresenta a tela de login. (3) O usuário preenche seus dados cadastrados. (4) O cliente pressiona o botão de logar. (5) O sistema valida o login no servidor. (6) O sistema redireciona o cliente para a tela principal. (6) O usuário pode fazer o cadastro, caso não tenha ainda. (7) O usuário pressiona o botão “Cadastro”. (8) O sistema apresenta a tela de cadastro. (9) O usuário pode redefinir sua senha. (10) O usuário pode se autenticar usando sua conta Google.

4.13.2. Diagramas de Seqüência do Aplicativo do Fiscal

A descrição do diagrama de sequência de listar estar do agente de trânsito é mostrado como exemplo, todos as outros descrições de diagramas encontram-se no (APÊNDICE 10).

Listar estar (APÊNDICE 8.8): Esse diagrama defina o funcionamento da listagem principal do aplicativo do fiscal. (1) O sistema exibe a tela principal com os períodos abertos em uma distancia de 1 km (um quilometro) da posição atual do usuário. (2) O usuário define um filtro para apenas vencidos aparecerem. (3) O servidor envia a lista filtrada e o sistema exibe para o usuário. (4) O usuário clica em um item da lista e o sistema redireciona para a tela de visualização.

4.14. DIAGRAMA DE CASOS DE USO

O diagrama de casos de uso mostra tudo que o sistema faz de acordo com a visão do usuário, ou seja, as funcionalidades do sistema e as interações do usuário com essas funcionalidades. No diagrama não são mostrados aprofundamentos com as funcionalidades, algo presente apenas nas descrições dos casos de uso, onde são detalhados os caminhos do funcionamento de cada caso de uso presente no diagrama (Sampaio, 2007).

Na Figura 15 é possível ver o Diagrama de casos de uso do aplicativo do Cliente e todas suas funcionalidades. Suas especificações mais detalhadas podem ser vistas no (APÊNDICE 6).

Na Figura 16 mostra o mesmo diagrama, mas agora para o aplicativo do Fiscal. Já suas especificações estão presentes no (APÊNDICE 7).

A Figura 17 nos mostra o sistema geral, unindo ambos os aplicativos e mostrando basicamente as interações do sistema completo.

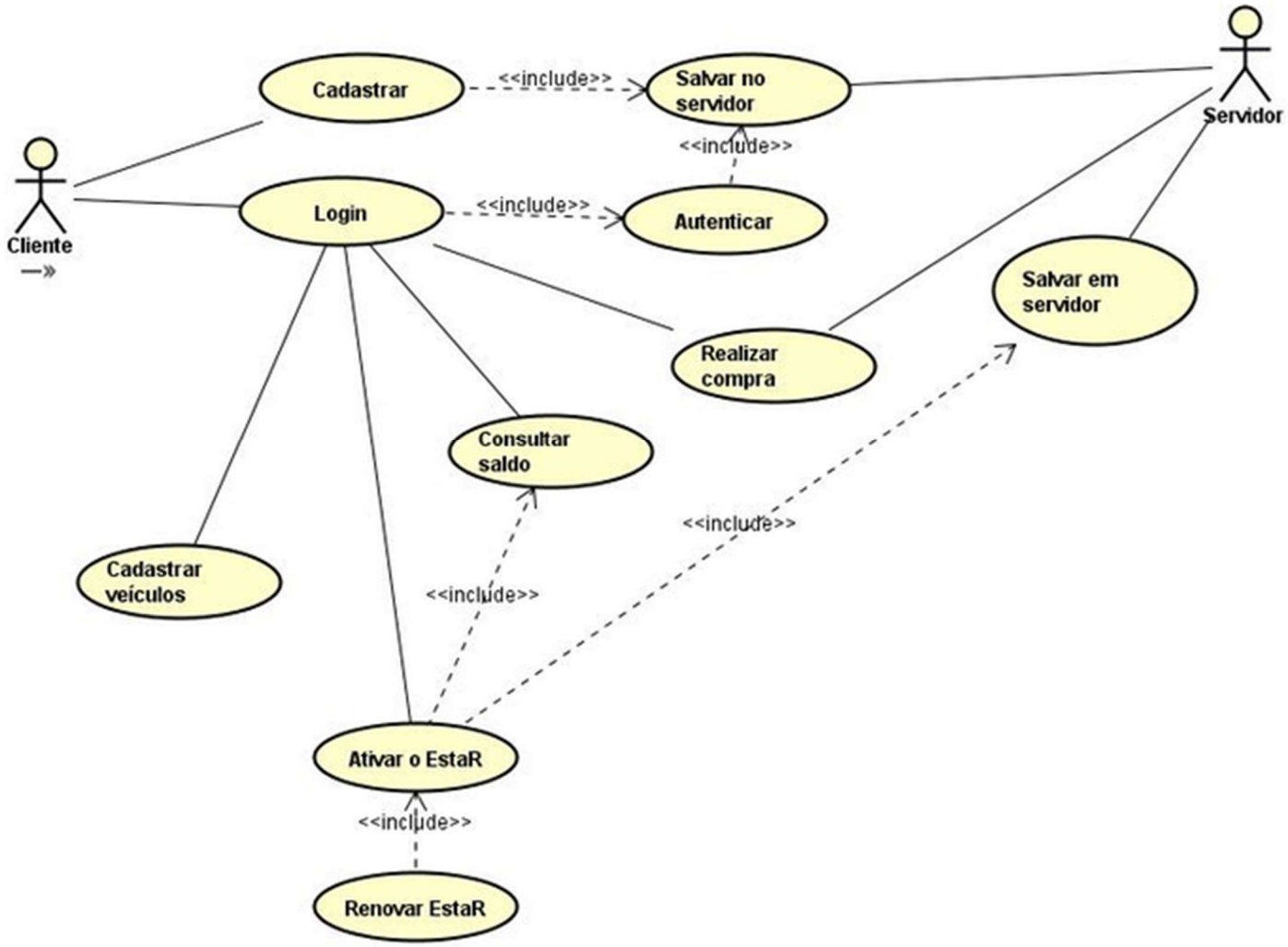


FIGURA 15- DIAGRAMA DE CASOS DE USO DO APP CLIENTE.

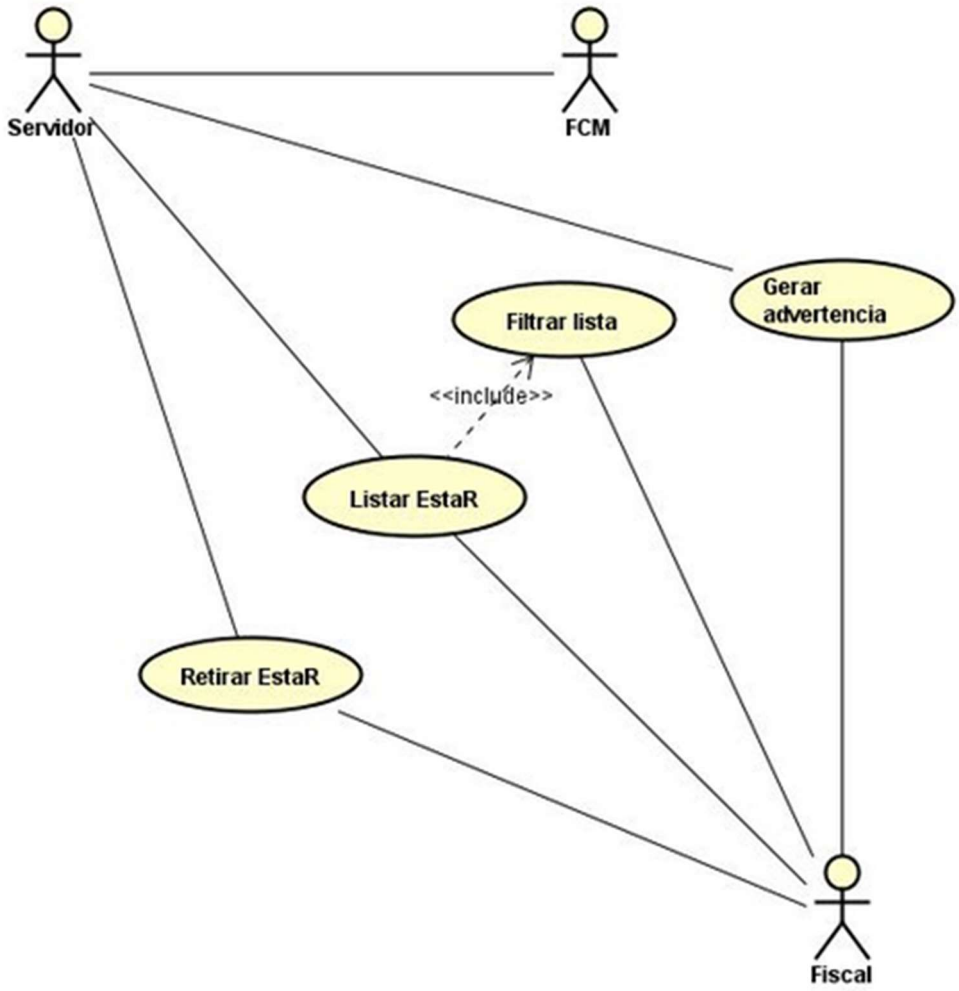


FIGURA 16 - DIAGRAMA DE CASOS DE USO DO APPFISCAL.

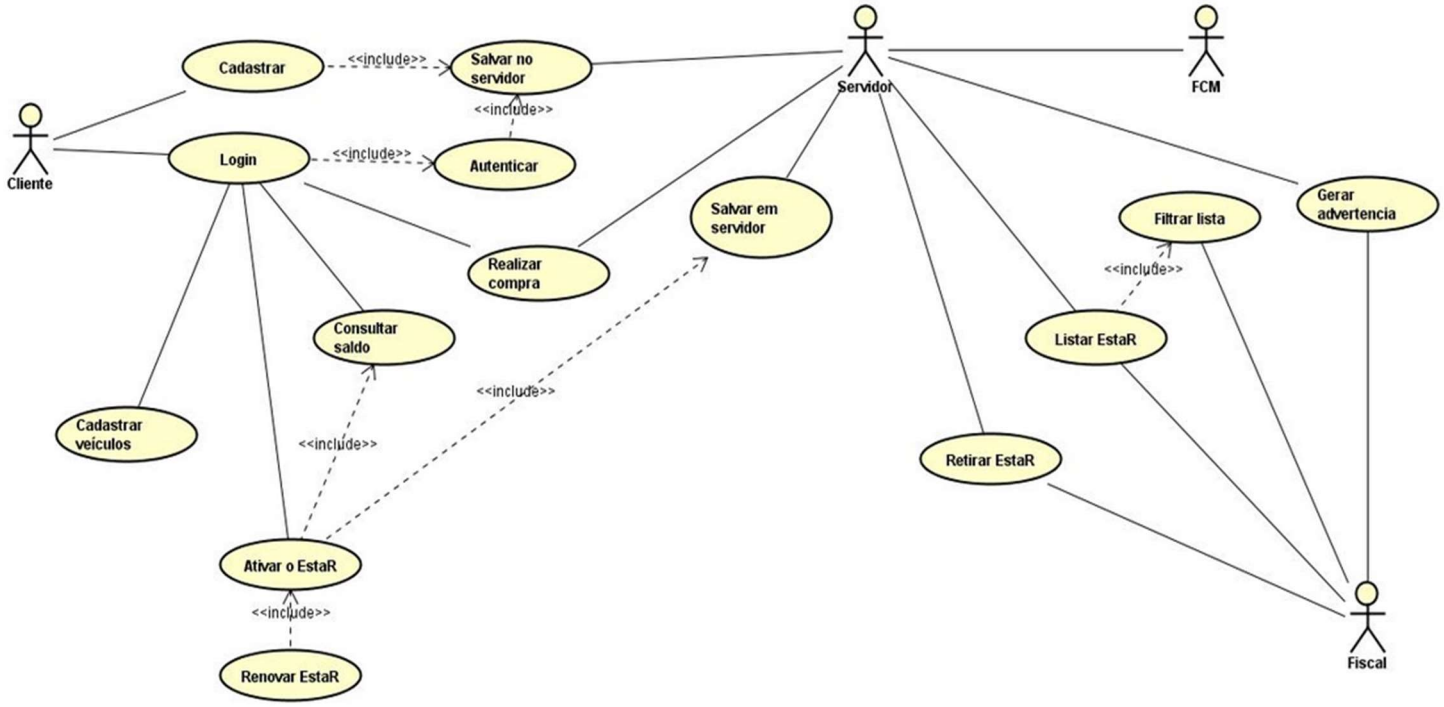


FIGURA 17- DIAGRAMA DE CASOS DE USO GERAL DOS APLICATIVOS

4.15. DIAGRAMA DE CLASSE

Um diagrama de classes representa a estrutura de relações das classes que servem de modelo para os objetos do sistema. Nele é importante conter, além das classes, seus métodos mais importantes, moldando assim uma base importante para a construção do sistema (Martinez, 2016).

Foi modelado antes e depois do desenvolvimento dos aplicativos, sendo no começo uma modelagem para ilustrar apenas as entidades modelo do sistema, e no final utilizando também as entidades de consumação de *web services*.

É possível ver o diagrama de classe do aplicativo do cliente (APÊNDICE 11) que mostra o pacote *request,* que é onde estão as classes que consomem os *web services.* Essas classes acionam as classes *model,* que são as responsáveis pela criação de objetos. Há também a classes de *countdown* responsáveis pelo contador do sistema.

A classe *countdown* permite criar um contador com um número determinado de segundos que, quando iniciado pelo método `start()`, fará uma contagem decrescente, até zero, notificando um *listener*, associado pelo método `setCountDownListener()`, quando o valor do tempo restante for decrementado ou incrementado através do método `increaseBy()` e quando o contador chegar ao fim.

É possível parar a contagem com o método `stop()`e retomá-la com o método `resume()`. Para controlar o comportamento desse contador, foi criado uma outra classe que encapsule o comportamento necessário para a regra de negócio do aplicativo durante a contagem, que é a *CountDownBehavior* que quando associado

ao contador, através do método *setCountDownListener*(), o contador chamará os métodos *onChange()*e *onEnd()*, toda a vez que a contagem é decrementada e quando chegar ao fim. O método *onAlarm( )* foi utilizado para atender a regra de negocio de alertar o usuário o termino de seu periodo, já o método displayTimeLeft() é usado para mostrar em tela o tempo restante.

Já no diagrama de classe do aplicativo do fiscal (APÊNDICE 12) é possível notar que só há 1 (uma) classe *model,* que é a classe Estar usada para preencher a lista principal do aplicativo. Como foi utilizado nesse caso a *lib*Retrofit para as requisição HTTP, é possível ver a estrutura da utilizada pela api, onde é criada uma classe para os métodos de requisições do aplicativo. Essas classes também acionam a classe *model*, responsável por criar o objeto. É destacado também a classe *adapter*, utilizada em conjunto com a classe *model* para a popularização da listview. Nela é possível trabalhar com a customização da lista, como mudar as cores e trabalhar com os objetos como itens separados.

4.16. BANCO DE DADOS

Banco de dados é o local onde toda a informação processada pelos aplicativos fica salva. No caso do trabalho em questão, o banco de dados usado é alocado com o servidor Hostinger, e toda requisição feita são através dos *web services.*

O banco foi modelado logo após a análise de requisitos, onde foi focado atender não somente aos dois protótipos, como preparar o banco já para uma versão posterior, mais robusta.

A Figura 18 apresenta o modelo do banco de dados que hoje o sistema possui.

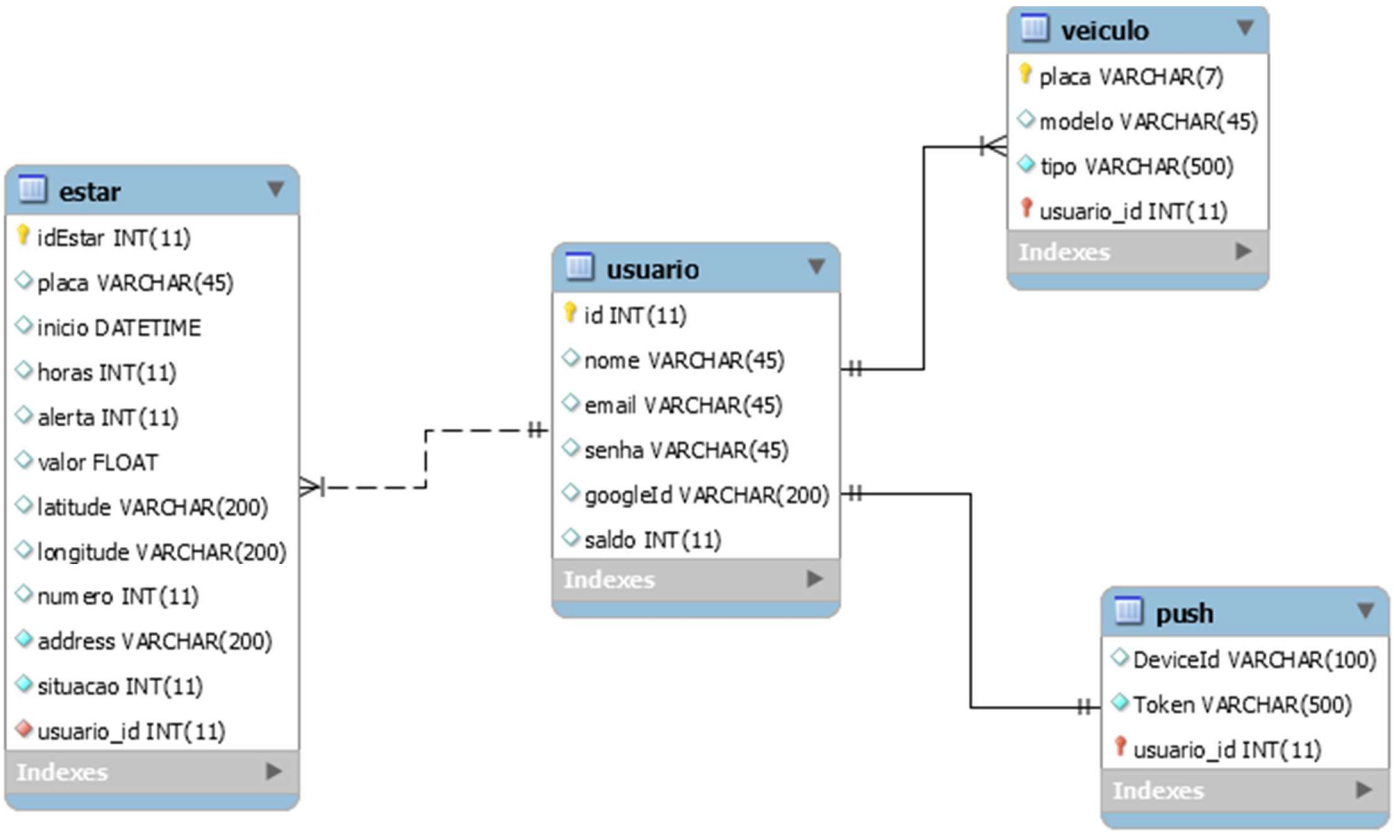


FIGURA 18- MODELO LÓGICO DO BANCO DE DADOS

Como dito, o sistema foi pensado para atender não só ambos os protótipos mas também a versão definitiva, tendo assim tabelas que só viram a ser usadas posteriormente, caso das tabelas “Fiscal”, onde no protótipo não há autenticação para o fiscal, da tabela “Endereço” que seria o caso de, quando já em produção, serem cadastrado os endereços que possuem Estar para que o sistema possa validar se o endereço está correto ou não, e da tabela “Multa” onde o fiscal poderia aplicar uma multa direto da aplicação.

Já as outras tabelas atendem as necessidades dos protótipos, onde o Usuário pode ter um ou mais veículos, também pode ter um ou mais períodos de estar. A tabela push é única e exclusivamente usada para armazenar o *token* e o *deviceid* do usuário para enviar push para seu dispositivo.

O modelo físico das bases utilizadas no protótipo (APÊNDICE 13) foi construído usando o SGBD MySQL, devido a ser o único suportado pelo servidor escolhido. A construção foi feita a partir do modelo lógico, nem sempre respeitando o relacionamento.

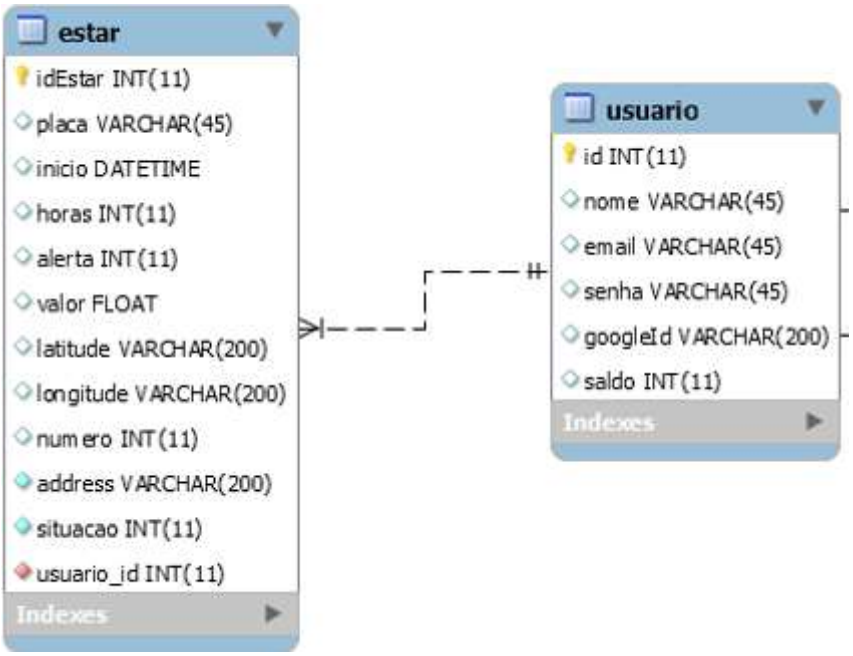


FIGURA 19- EXEMPLO DE CARDINALIDADE ENTRE TABELAS

4.17. API

API é como uma interface entre dois programas diferentes de modo que eles possam se comunicar um com o outro. Ou seja, uma API é a forma que terceiros disponibilizam uma interface de modo que possa ser consumido um determinado serviço deles sem nos preocuparmos com a implementação do mesmo. As API podem usar qualquer meio de comunicação para iniciar a interação entre as aplicações. Por exemplo, as chamadas de sistema (System Calls) são invocados usando interrupções da API do kernel Linux (COSTA,2015).

4.17.1. Api Estar Cliente

A descrição da API de Login do condutor é mostrado como exemplo, todos as outros descrições de API encontram-se no (APÊNDICE 14).

LOGIN

Nome	Login mobile	URL	http://wsestarapp.esy.es/safe/loginmobile.php
Método	POST	POST	
Input	e-mail	pass	
Output	result	content	exception
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	retorna um array de dados do usuário que acabou de se logar.	em caso de exceção, retorna o motivo da falha

4.17.2. ApiEstar Fiscal.

A descrição da API de Lista principal do agente de trânsito é mostrado como exemplo, todos as outros descrições de API encontram-se no (APÊNDICE 15).

4.18. REGRA DE NEGÓCIO

Regra de negócio se diz respeito as alterações que foram utilizadas apenas para fins de apresentação e demonstração dos protótipos, tendo seu código sido preparado para alterações futuras.

São elas:

- É possível renovar o período apenas 1 (uma) vez.
- Os preços nem sempre condizem que o preço real do Estar.

4.19. GLOSSÁRIO DE DADOS

Um dicionário de dados contém as características lógicas dos dados que serão utilizados no sistema que foi desenvolvido. Estes dicionários se desenvolvem durante a análise de fluxo de dados e ajuda aos analistas que participam na determinação dos requerimentos do sistema, seu conteúdo também se emprega durante o desenho do projeto (ALVAREZ, 2014).

O glossário de dados baseado no diagrama de classes (APÊNDICE 16) mostra a descrição de atributos presente no diagrama em questão. Já o glossário de dados baseado no banco de dados (APÊNDICE 17) oferece suporte para a descrição dos atributos presente no modelo lógico.

5. APRESENTAÇÃO DO SOFTWARE DO CLIENTE

Esse capítulo demonstra a apresentação das logos e das interfaces do aplicativo do cliente, citando suas funcionalidades, detalhes de navegação e regras de negócio.

5.1. LOGO

FIGURA 20- LOGO DO APLICATIVO ESTAR CLIENTE

5.2. TELA DE LOGIN

A tela de login () é visualizado assim que o aplicativo é iniciado e o usuário não tenha selecionado para manter-se logado, e contem as seguintes componentes:

Login: formulário composto pelos campos email e senha, onde o usuário já cadastrado faz o login no sistema. Um campo de checkbox define se o usuário manterá ou não seus dados de login salvos na próxima entrada.

Link: quando acessado redireciona o usuário para a tela de recuperar senha.

Botões “cadastro” para redirecionar o usuário para a tela de cadastro, “Login com o Google” para utilizar a conta do Google para se autenticar no sistema, e “Login” onde o usuário previamente cadastrado faz o login

5.3. TELA DE CADASTRO

A tela de cadastro (APÊNDICE 6: A.2 UC002 – Cadastrar) é onde ocorre o cadastro de usuários no sistema. É acessível após o usuário pressionar o botão “Cadastro” na tela de login. A tela contem os seguintes componentes:

Formulário: com campos de nome, email e senha onde o usuário preenche com os dados que deseja usar na hora de logar.

Botões “cadastrar” que envia os dados ao servidor e “voltar” que leva o usuário para a tela de login novamente.

5.4. RECUPERAR SENHA

A tela de recuperação de senha (APÊNDICE 6: A.3 UC003 – Recuperar senha) é onde o usuário pode recuperar sua senha com cadastro previamente feito no sistema. É acessível quando acionada a opção “Esqueci minha senha” na tela de login, e possui os seguintes componentes:

Formulário: com o campo email, onde o usuário preenche com seu email cadastrado para que seja enviado sua nova senha.

Botão de salvar, para enviar ao servidor o email cadastrado e gerar uma nova senha para o usuário.

5.5. FLUXO PRINCIPAL

A tela principal (APÊNDICE 6: A.4 UC004 – Tela principal) é a interface principal do aplicativo cliente. É acionada após o usuário se autenticar no sistema previamente. A tela principal é dividida em *tabs,* onde cada tab redireciona o usuário para uma tela diferente.

As tabs são compostas por:

- Principal: a tela de inicial do aplicativo, onde é possível:
- Visualizar sua posição atual em um *mapview.*
- Acionar o botão “ativar” que redireciona o usuário para a tela de ativação.
- Créditos (APÊNDICE 6: A.7 UC007 – Realizar compra): A tela para comprar é acionada após o usuário selecionar a *tab* “créditos” na tela principal do aplicativo e é composta por:
- Itens disponíveis para a compra, com a quantidade de créditos e o preço de cada um deles. Ao selecionar, é exibido um *alert* de confirmação para o usuário.
- Texto com o saldo atual do usuário.

- Veículos: (APÊNDICE 6: A.8 UC008 – Realizar compra): A tela para cadastro de veículos e é acionada após o usuário selecionar a *tab* “veículos” na tela principal do aplicativo. É composta pelos componentes:
- Formulário: com os campos modelo e placa do veiculo desejado para cadastro.
- Lista: com os veículos já cadastrados pelo usuário.
- Botão “adicionar” para enviar ao servidor os dados inseridos.
- Histórico: (APÊNDICE 6: A.9 UC009 – Histórico): A tela de histórico é acionada após o usuário selecionar a *tab* “histórico” na tela principal do aplicativo e é composto pelos componentes:
- Lista: com todos os períodos já abertos pelo usuário.

5.6. TELA DE ATIVAÇÃO

A tela de ativação de ativação (APÊNDICE 6: A.5 UC005 – Ativar) é onde o cliente pode ativer o seu EstaR. É acionado pelo botão “Ativar” presente na tela principal. É composta pelos componentes:

- Formulário: com um *Spinner* com todas as placas de veículos já cadastrados pelo usuário. Um *Spinner* com o tempo desejado. E um dialog com o tempo que o usuário quer que toque um alerta de tempo se esgotando.
- Botão “ativar” para enviar os dados ao servidor.

5.7. TELA DE GERENCIAMENTO

A tela de gerenciamento do estar (APÊNDICE 6: A.6 UC006 – gerenciar estar) é a tela onde o usuário gerencia seu período ativo. É acionado após o servidor aprovar a ativação do período. É importante que, quando o usuário tiver um período ativo, ele fique preso na tela de gerenciamento até que o período seja finalizado, mesmo se fechar o aplicativo. É composta pelos componentes:

- Contador com o tempo restante.
- Botões “+1” para renovar o período por +1 minuto no protótipo e “Finalizar” para finalizar o período ativo.
- Mapa com a posição onde foi estacionado e a posição atual do usuário.

6. APRESENTAÇÃO DO SOFTWARE DO FISCAL

Esse capitulo abrange a apresentação das logos e das interfaces do aplicativo do fiscal, citando suas funcionalidades, detalhes de navegação e regras de negócio.

6.1. LOGO

FIGURA 21 - LOGO APLICATIVO ESTAR FISCAL

6.2. TELA PRINCIPAL

A tela de gerenciamento do estar (APÊNDICE 7: A.1 UC001 – gerenciar estar) é a tela onde o usuário gerencia os períodos próximos a ele. É composta pelo componente:

Lista: com os períodos ativos em verde, e os vencidos em vermelho.

6.3. TELA DE VISUALIZAÇÃO

A tela de gerenciamento do estar (APÊNDICE 7: A.2 UC002 – visualizar estar) é a tela onde o usuário visualiza o período selecionado. É acionada a partir da tela principal, ao clicar em algum item da lista. É composto por:

Textos com data e hora de ativação, placa do veículo, endereço, quantidade de horas compradas, modelo do veículo.

Mapa com a posição do veículo e a posição atual do usuário.

6.4. TELA DE SCANNER

A tela de scanner (APÊNDICE 7: A.2 UC003 - scannear) é a tela onde o usuário possui um *OCR* (OpticalCharacterRecognition) que possui junto a ele um *pattern* para reconhecer placas de veículos nacionais. Ao reconhecer a placa, o sistema exibe um retângulo com o texto e nele o usuário clica é levado a um *dialog*

onde confirma se a leitura foi correta. Se sim, pressiona o botão de “Ok” e a lista é filtrada pela placa selecionada.

7. CONSIDERAÇÕES FINAIS

O objetivo do projeto Plataforma para controle de estacionamento municipal, ou simplesmente EstaR, é um sistema de gerenciamento de estacionamento publico, voltado a compra e utilização por parte do cliente. É voltado também para o fiscal, que possui seu aplicativo próprio para gerenciamento do estacionamento publico. Além disso, o sistema é disponibilizado apenas para uma plataforma mobile, sendo ela a mais utilizada e de mais fácil acesso no país.

É importante salientar que o aplicativo vem com o objetivo de automatizar o processo que hoje é feito através de papel, e que gera muita desconforto de boa parte de seus usuário. Utilizando o aplicativo do EstaR, é possível dizer que esse objetivo foi alcançado, tendo assim uma facilidade e uma acomodação ainda maior.

Durante o desenvolvimento do projeto, algumas dificuldades foram encontradas como: dificuldade em conciliar a agenda da equipe, já que ambos possuem atividades no mercado de trabalho; não possuir o hardware necessário para o desenvolvimento de aplicativos mobile; falta de conhecimento de parte da equipe sobre a plataforma escolhida; tecnologias usadas que tem o funcionamento dependente do sistema operacional, o que faz com que seja ele que controle seu funcionamento correto.

Dessa forma, novos conhecimentos foram adquiridos ao longo do projeto, fora os aprendidos durante o curso de Tecnologia em Analise e Desenvolvimento de Sistemas da Universidade Federal do Paraná. Levando em conta que o curso em questão não tem uma carga horária muito pesada para desenvolvimento mobile, grande parte das tecnologias utilizadas foram aprendidas no decorrer do projeto, tais como: Services, JobScheduler, Push.

É importante salientar que ambos os aplicativos tratam-se de protótipos, onde foi escolhido não utilizar-se de autenticação para aplicação do fiscal, nem de pagamentos reais na aplicação do cliente. Algumas regras de negócio foram alteradas para atender ao tempo de apresentação do software.

Analisando o sistema, é possível visualizar algumas vantagens de seu uso para o cliente, tais como: facilidade de acesso, comodidade, porcentagem pequena de erro, fácil gerenciamento. Há vantagens também para o agente de trânsito, onde pode gerenciar facilmente todos os períodos de estar ativos perto dele, pode notificar facilmente um motorista, e também visualizar a rota do local de estacionamento pela sua posição atual.

Há também as desvantagens, tais como: ter internet sempre disponível; necessidade do dispositivo possuir carga durante o período necessário; ter um dispositivo com GPS e estar em um local onde o sinal de GPS é encontrado.

Por fim, o EstaR possui algumas limitações como gerenciamento de alertas, onde o sistema operacional é responsável por exibi-los. Sendo assim, para futuras implementações e atualizações do sistema, recomenda-se que haja o desenvolvimento de interfaces para outros sistemas móveis, como iOS e Windows Phone, aumentando a integração com as mais diversas plataformas móveis disponíveis no mercado; O desenvolvimento de uma rotina, onde através dela será

possível gerar os alertas no servidor, e não local como é feito no protótipo; A criação de um painel administrativo para CRUD de fiscais; A criação de um painel administrativo para gerencias as advertências e multas; A geração de multas através do aplicativo do Fiscal e gerenciamento dessas multas no aplicativo do Cliente.

REFERÊNCIAS

FELIPE, João. **Orientação a Objetos: Uma visão inicial.** DevMedia, 2016. Disponível em <http://www.devmedia.com.br/orientacao-a-objetos-uma-visao-inicial/34308> Acessado em 10/11/2016.

SETRAN. **About.** Disponível em <http://www.setran.curitiba.pr.gov.br/estar> Acessado em 10/11/2016.

FREITAS, Marcio**. UML Fundamentos.** DevMedia, 2016. Disponível em <http://www.devmedia.com.br/uml-fundamentos/8640> Acessado em 11/11/2016.

ORACLE.**The importance of Using the Unified Modeling Language (UML)**. Disponível em <https://blogs.oracle.com/JavaFundamentals/entry/the_importance_of_using_unified > Acessado em 25/10/2016.

CAELUM. **O que é Java.** Disponível em <https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java/> Acessado em 13/11/2016.

SOA. **Como funcionam os Web Services.** Disponível em <http://www.soawebservices.com.br/como-funciona.aspx> Acessado em 09/11/2016.

SPRING**. Understanding REST.** Disponível em <https://spring.io/understanding/REST> Acessado em 10/11/2016.

CORRÊA, Eduardo.**Introdução ao formato JSON.** DevMedia, 2016. Disponível em <http://www.devmedia.com.br/introducao-ao-formato-json/25275> Acessado em 15/11/2016.

PHP. **Uso da linguagem PHP em sistemas server-side**. Disponível em<http://w3techs.com/technologies/overview/programming_language/all> Acessado em 12/11/2016.

MYSQL. **About**. Disponível em <http://www.mysql.com/> Acessado em 12/11/2016.

Lee,Wei-Meng. **Introdução Ao Desenvolvimento de Aplicativos Para o Android**. Editora Ciência Moderna. Rio de Janeiro – RJ,2011.

GOOGLE MAPS. **About**. Disponível em <https://developers.google.com/maps/documentation/android-api/?hl=pt-br> Acessado em 15/11/2016.

FIREBASE. **About**. Disponível em <https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html> Acessado em 15/11/2016.

VOLLEY. **About**. Disponível em <https://developer.android.com/training/volley/index.html> Acessado em 16/11/2016.

ANDROID. **About**. Disponível em <https://developer.android.com/develop/index.html> Acessado em 16/11/2016.

RETROFIT. **About**. Disponível em <https://square.github.io/retrofit/> Acessado em 16/11/2016.

MEDEIROS, Higor. **Introdução aos Processos de Software e o Modelo Incremental e Evolucionário.** DevMedia, 2016. Disponível em

<http://www.devmedia.com.br/introducao-aos-processos-de-software-e-o-modelo-incremental-e-evolucionario/29839> Acessado em 15/11/2016.

MICROSOFT. Identificar e se planejar para riscos. Disponível em <http://office.microsoft.com/pt-br/project-help/objetivo-identificar-e-se-planejar-para-riscos-HA010225438.aspx> Acessado em 15/11/2016.

GUSMÃO, Cristine. Visão da gerência de riscos na engenharia de software. Disponível em <http://www.devmedia.com.br/visao-da-gerencia-de-riscos-na-engenharia-de-software/9144> Acessado em 18/11/2016.

SOBRINHO, Cristiana. Apresentação do Android (Sistema Operativo). Disponível em <http://knoow.net/ciencinformtelec/informatica/android-sistema-operativo/> Acessado em 18/11/2016.

MALIMI, Ngeleki. **The Beginner's Guide to Android: Android Architecture**. Disponível em <http://ngeleki.blogspot.com.br/2014_03_26_archive.html> Acessado em 18/11/2016.

PALPITEDIGITAL. **O que é Kernel?** Disponível em <https://www.palpitedigital.com/o-que-e-kernel> Acessado em 18/11/2016.

SAMPAIO. **Casos de Uso.** Disponível em <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm> Acessado em 28/11/2016.

PRETTO, Samuel. **COLETA DE DADOS COM DISPOSITIVOS MÓVEIS: UM ESTUDO DE CASO APLICADO À PRODUÇÃO AVÍCOLA.** Disponível em <https://www.univates.br/bdu/bitstream/10737/384/1/Samuel%20Pretto.pdf> Acessado em 28/11/2016.

QUITÉRIO, Ana Paula. **Análise de Requisitos.** Disponível em <http://www.infoescola.com/engenharia-de-software/analise-de-requisitos/> Acessado em 28/11/2016.

ISLIG-RIO. **Gerência de Riscos.** Disponível em <http://www.bfpug.com.br/islig-rio/Downloads/Gerencia_de_Riscos.pdf> Acessado em 28/11/2016.

PRIMO, Glauco. **UserStories – O que são? Como Usar?** Disponível em < http://blog.myscrumhalf.com/2011/10/user-stories-o-que-sao-como-usar> Acessado em 28/11/2016.

COSTA, Félix. **Diferença entre API e Web Service de maneira simples.** Disponível em https://fxcosta.wordpress.com/2015/05/31/diferenca-entre-api-e-web-service-de-maneira-simples> Acessado em 28/11/2016.

ZARELLI. **Como funciona o SOAP.** Disponível em <https://zarelli.wordpress.com/2012/03/22/como-funciona-o-soap-protocolo-simples-de-acesso-a-objetos/> Acessado em 05/12/2016.

LEE, Wei-Meng. **Introdução ao Desenvolvimento de Aplicativos para Android**. 1. ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2011

SAUDATE, Alexandre. **Construa Api S Inteligentes de Maneira Simples.** 2013 , Casa do Codigo.

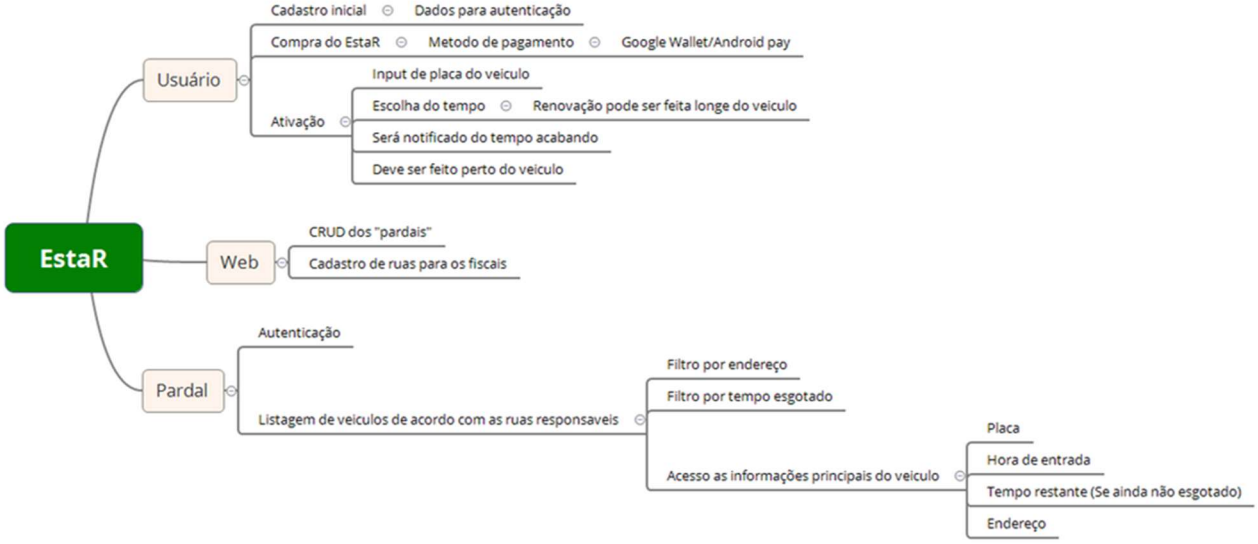
LECHETA, Ricardo R. **Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK**. Ed. Novatec. 2ª ed. São Paulo, 2010.

DEVELOPERS, Google. Disponível em < https://developers.google.com/maps/documentation/android/mapkey.> Acessado em 05/12/2016.

APÊNDICES

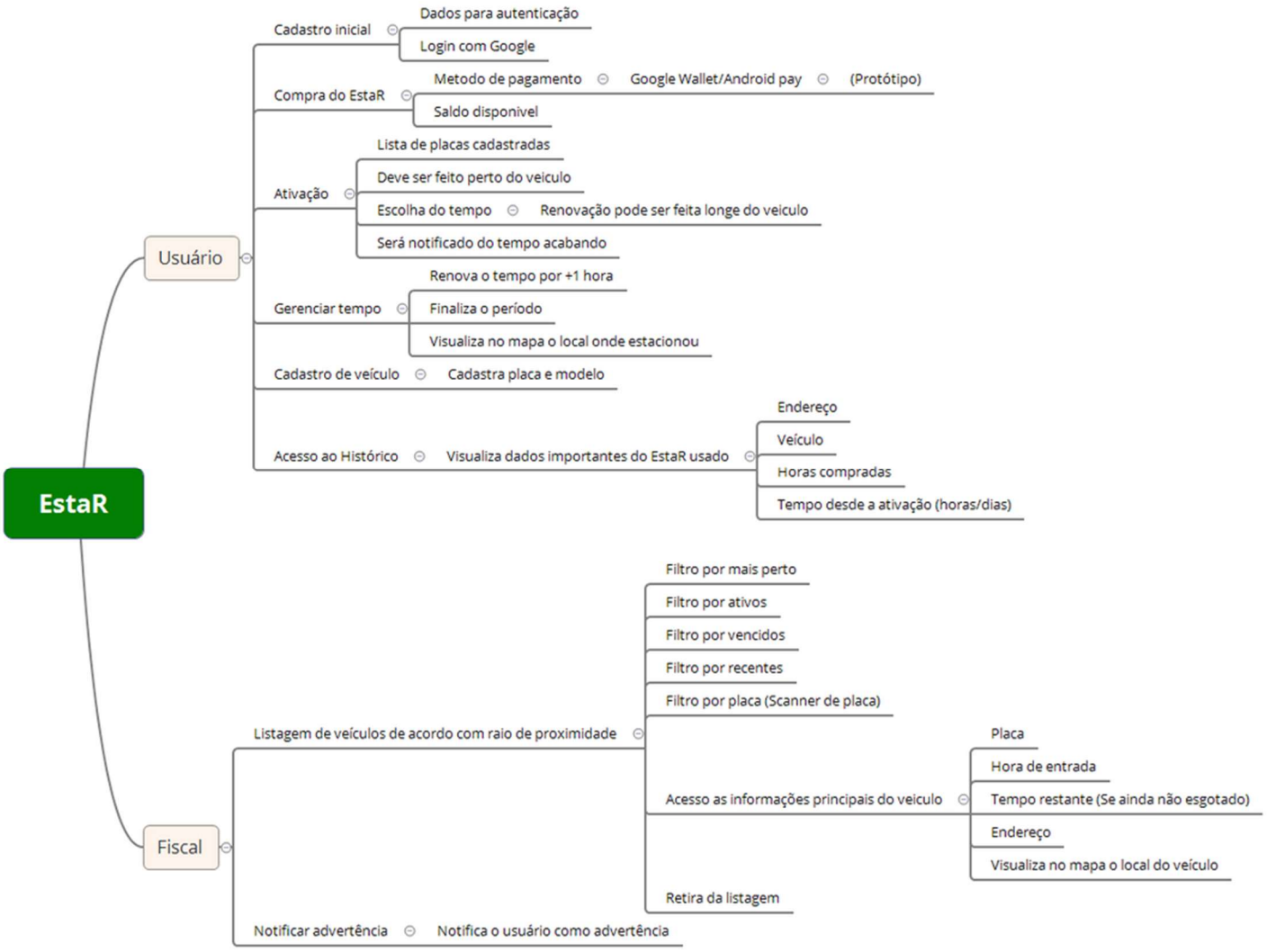
APÊNDICE 1 – MAPA MENTAL

PRIMEIRA VERSÃO

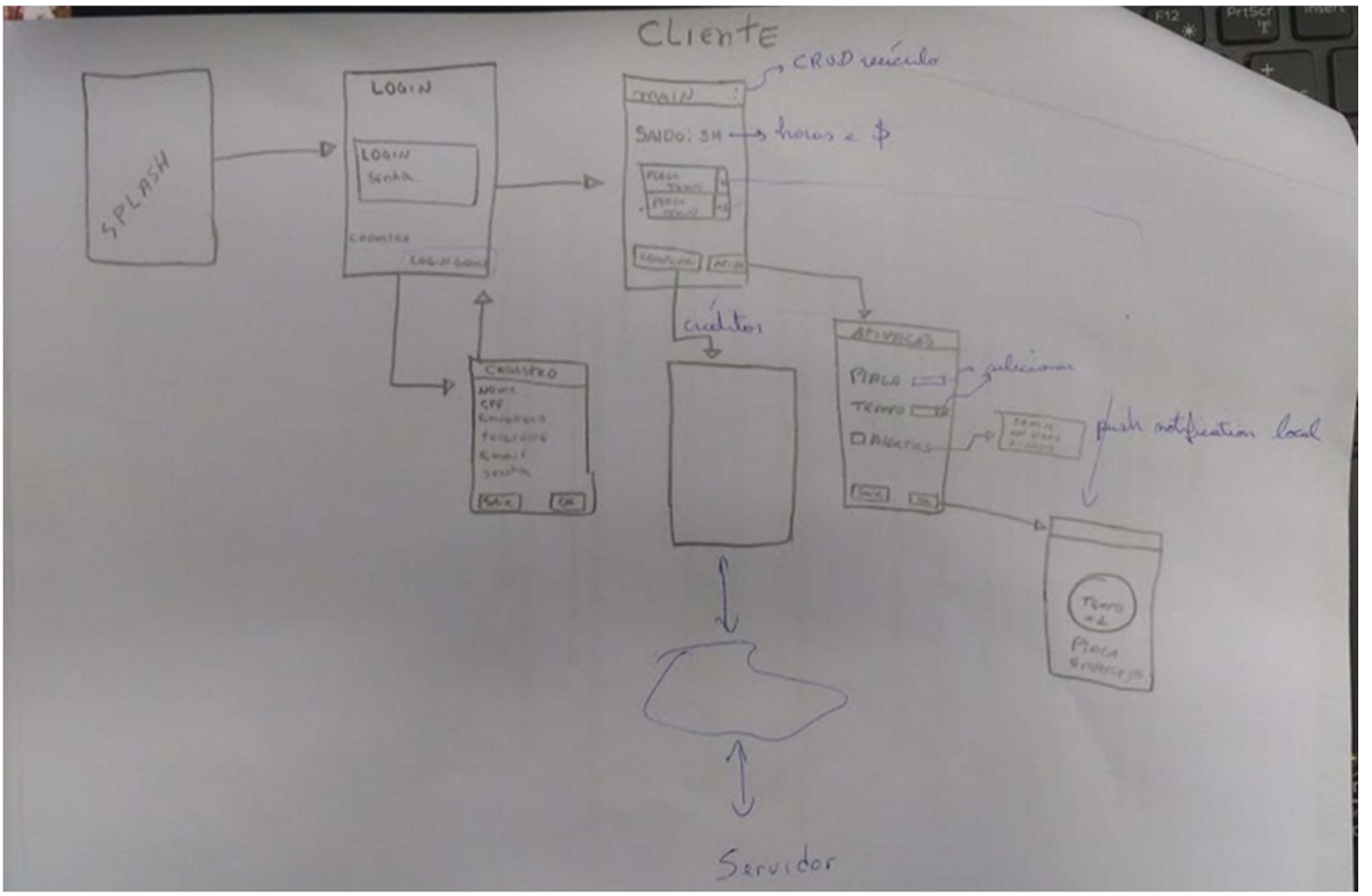


É possível notar que na primeira versão havia a idéia de criar um painel administrativo para cadastro de fiscais. Algo que na versão final foi retirado do escopo. É possível notar uma mudança nas funcionalidades também, de acordo com a versão final.

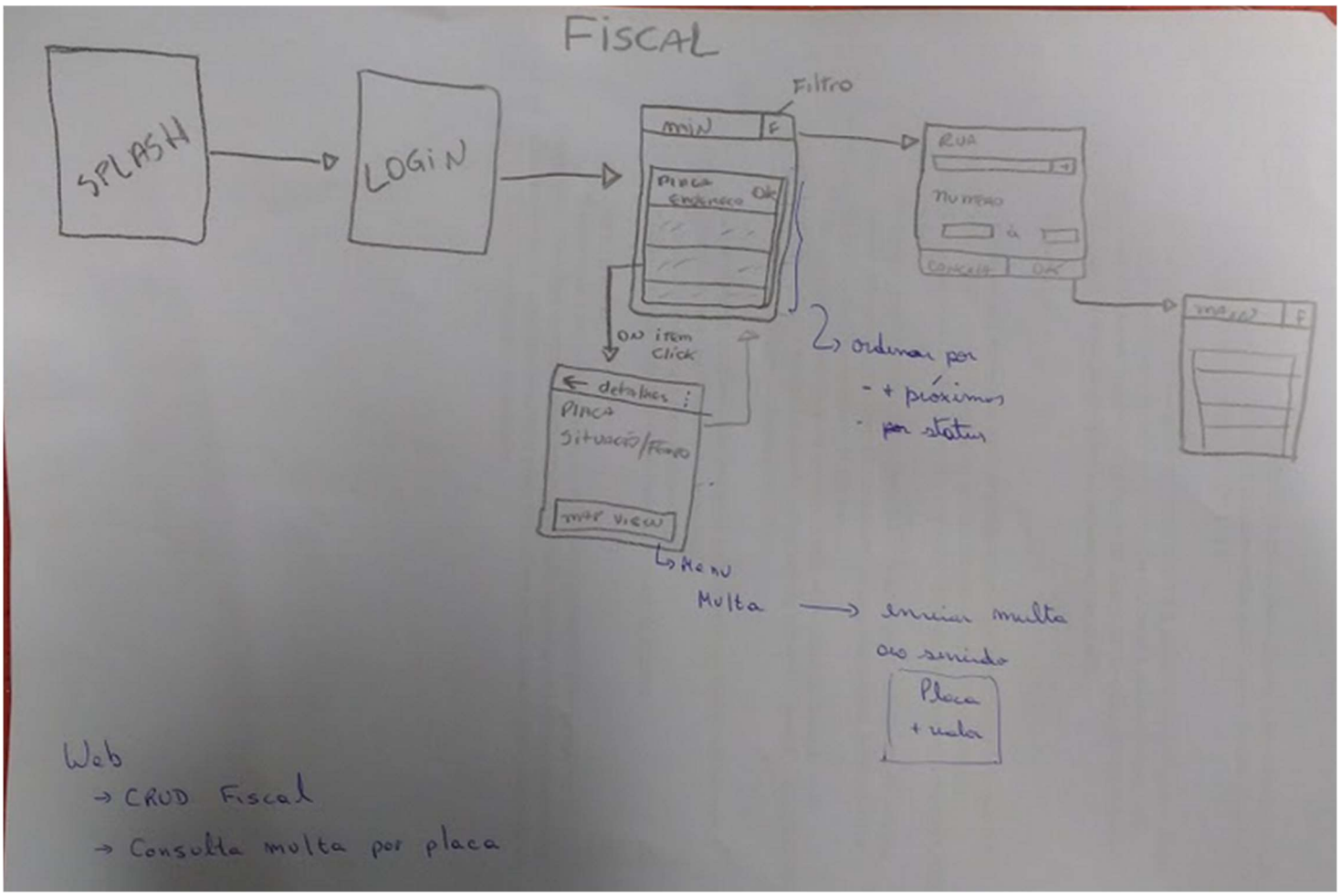
VERSÃO FINAL



APÊNCICE 2 – FLUXO APP CLIENTE BAIXA FIDELIDADE



APÊNDICE 3 – FLUXO APP FISCAL BAIXA FIDELIDADE



APÊNDICE 4 – USER STORIES APP CLIENTE

LOGIN

Como cliente, quero acessar o aplicativo por meio de uma tela de login simplificada, em que eu possa digitar os meus dados de acesso, para que eu não tenha que criar múltiplos usuários de senhas para acompanhamento do meu Estar.

5 A estrutura do login é composta por:
  - 5.1 E-mail.
  - 5.2 Senha.
  - 5.3 Login com o Google.

6 O e-mail de acesso deve ser o e-mail cadastrado pelo usuário.

7 O aplicativo deve guardar os dados do login para permitir a entrada sem a necessidade de autenticação, caso seja requerido pelo usuário no item *manter logado*.

8 O login com Google é feito automaticamente, escolhendo a conta desejada. Caso não faça o *logout*, sempre manterá logado.

CADASTRO

Como cliente, quero cadastrar meus dados para poder utilizar os recursos presentes no aplicativo em questão.

1. A estrutura do cadastro é composta por:
   - 1.1. Nome.
   - 1.2. E-mail.
   - 1.3. Senha.
2. O *nome* cadastrado será mostrado para o usuário na tela de boas vindas.
3. O *e-mail* e a *senha* serão os dados utilizados para o login do usuário, em caso de sucesso na requisição.

INICIAL

Como cliente, quero uma tela principal com várias abas, onde consiga navegar para realizar todas as funcionalidades presentes no aplicativo.

1. A estrutura das abas é composta por:
   1.1. Principal.
   1.2. Comprar.
   1.3. Veículos.
   1.4. Histórico.
2. Cada aba é levada para uma tela diferente, sendo a do *item A* tela inicial.

PRINCIPAL

Comocliente, quero que seja mostrada a minha posição atual em um mapa, para que eu saiba se o aplicativo está funcionando corretamente. Também quero que seja possível ativar um período de Estar.

1. Deve ser mostrado um *texto* de boas vindas para o usuário logado.
2. Logo abaixo das boas vindas, deverá ser mostrado um *mapa* com a posição atual do usuário, para que tenha certeza da funcionalidade do aplicativo.
3. No *bottom*, deve haver um botão para que o usuário seja encaminhado para a *tela de ativação do* Estar.

COMPRAR

Comocliente, quero comprar créditos para que possa utilizar o Estar em meu aplicativo.

1. Devem ser mostradas as opções de preços/horas disponíveis para compra.
2. Deve haver um dialog de validação para confirmar a compra.
3. Ao confirmar a compra, os créditos devem ser atualizados automaticamente.

VEÍCULOS

Comocliente, quero cadastrar meus veículos, para que seja mais fácil gerenciá-los mais tarde.

1. A estrutura é composta por:
   1.1. Placa.
   1.2. Modelo.
2. Deve haver também uma lista com os veículos já cadastrados pelo usuário.

3. Ao cadastrar, a lista deve ser atualizada automaticamente com o último veículo inserido.

HISTÓRICO

Como cliente, quero ver o histórico de ativações que meu perfil possui até então e todos os dados necessários.

1. O histórico deve conter:
   1.1. Endereço, em destaque.
   1.2. Placa do veículo.
   1.3. Horas usadas.
   1.4. Há quanto tempo foi ativado, contando do horário atual.

ATIVAÇÃO

Como cliente, quero ativar meu período de Estar assim que cheguei ao endereço desejado. Se eu não tiver saldo suficiente, eu devo ser avisado. Ao ativar, quero poder gerenciar o meu tempo e estar ciente de quanto tempo ainda falta para finalizar.

1. A estrutura de dados será:
   1.1. *Spinner* contendo os veículos cadastrados pelo usuário.
   1.2. *Spinner* com a quantidade de horas desejada.
   1.3. Campo de alertas, que abre um *dialog* com as opções de alertas: 15 minutos antes; 10 minutos antes; Não receber alertas.
   1.4. *Button* para confirmar e ativar o Estar.
2. Deve haver uma validação no *servidor* para verificar se o saldo do cliente é equivalente as horas desejadas.
3. O endereço deve ser capturado em *background*, sem a possibilidade de *input* do usuário.
4. O cliente deve ser enviado para a tela de *timer*, assim que a ativação for concluída com sucesso.
5. O *alerta* deve ser tocado no tempo em que o usuário escolheu. Ao clicar, será levado de novo para a tela de *timer*, para que possa renovar seu período (se ainda não tiver feito).

TIMER

Como cliente, quero gerenciar meu tempo e visualizar o local onde estacionei. Também quero ter a opção de finalizar meu período, para que não ocorram multas incorretas.

Quero também uma notificação com o tempo correndo, para que eu possa ter melhor acesso à informação.

Se houver um período ativo, o usuário deve ficar “preso” na tela de Timer, até finalizar o seu período. Para isso, são feitas validações no login, e no método de back.

1. A tela de tempo terá um cronômetro rodando, e também gerará uma *notification*para o usuário com o tempo mesmo cronômetro.
2. Será apresentado um *MapView*com a posição atual e a posição onde foi estacionado o veículo.
3. A tela é composta por dois botões, que possuem ações importantes:
4. Finalizar: Onde o usuário finaliza o período ativo.
5. Renovar: Onde o usuário tem a opção de renovar o seu tempo.

APÊNDICE 5 – USER STORIES APP FISCAL

INICIAL

Como fiscal, quero ter acesso a uma lista de estar's ativos e vencidos perto de mim. Também quero que seja possível utilizar de filtros na lista, para que eu possa visualizar períodos de acordo com meu desejo.

Quero também que a lista seja atualizada automaticamente a cada 30 segundos, sem a necessidade de ação do usuário. Deve também haver a opção de atualização com *swipe*da lista.

4. A estrutura do card de Estar é composta por:
   4.1. Endereço.
   4.2. Placa.
   4.3. Tempo restante, ou vencido.
5. O período que estiver vencido, deve ter seu *background* da cor vermelha.
   Enquanto o ativo deve conter seu *background* da cor verde.
6. Os filtros devem ser de:
   6.1. Ativos.
   6.2. Vencidos.
   6.3. Mais perto.
   6.4. Horário de ativação.

DETALHES

Como fiscal, quero poder visualizar detalhes de cada veículo presente na minha lista inicial. Também deve ser mostrado um mapa indicando o local onde o veículo está estacionado, para que possa ser traçado uma rota entre o local atual e o local de estacionamento.

Também é necessário que possa enviar um *push*de alerta para o usuário, quando há a necessidade de multa. Deverá também possuir a opção de retirar o veículo da lista inicial.

1. A tela deve conter os seguintes dados:
   1.1. Placa.
   1.2. Horas compradas.
   1.3. Hora de entrada, no formato *dd/mm/aaaahh:mm:ss.*
   1.4. Endereço do carro.

2. Deverá haver um *MapView* na parte inferior da tela, onde será mostrada a

posição atual do *device* e a posição do veículo em questão.

3. Deverá conter, logo acima do *MapView*, dois botões. O primeiro, de *label* “Retirar” deve realizar a ação de retirar o veículo da lista inicial. Já o segundo, com *label*“ Notificar” deve enviar um *push* para o usuário do veículo em questão, notificando sobre a advertência realizada pelo fiscal.

SCANNER

Como fiscal, quero poder realizar um scan de uma placa com minha câmera e pesquisar todos os períodos dela.

Deverá haver um *OCR* para o scan da placa do veículo desejado. Após efetuar o scan, deve aparecer um *dialog* com a informação da placa para confirmar se o texto está correto. Se correto, deverá filtrar da lista pela placa encontrada.

APÊNDICE 6 – DESCRIÇÃO DE CASOS DE USO APP CLIENTE

UC001 – Logar

Descrição
Este caso de uso serve para o usuário logar no sistema Estar via aplicativo Android.

Data View

DV1 – Tela de login

Pré-condições
Este caso de uso pode iniciar somente se:
1. O usuário deve ter instalado o aplicativo Android.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. Iniciar o Caso de Uso **UC004 – Página Principal Android.**

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela **(DV1)**;

- O usuário preenche os campos “Email” e “Senha”; **(A2)**
- O usuário pressiona o botão “Logar”;**(A1)(A3)**
- Os dados são verificados pelo sistema. **(E1)(E2)(E3)(R4)**;
- O sistema apresenta o **UC004 – Página Principal Android**

Fluxos Alternativos

A1. Se usuário clicar em “Login com o Google”:
- É apresentado uma tela com as contas Google presentes no telefone.
- O usuário seleciona a sua conta e os dados são enviados para o sistema.

A2. Se usuário clicar em “Cadastrar”:
- O sistema apresenta o **UC002 – Tela de cadastro do cliente.**

A3. Se usuário clicar em “Esqueci minha senha”:
- O sistema apresenta o **UC003 – Tela de recuperação de senha.**

Fluxos de Exceção

E1. Campos obrigatórios não preenchidos:
- O sistema consiste os campos **(R1)**
- O sistema retorna a mensagem “Todos os campos são obrigatórios!”.
- O Caso de Uso é reiniciado.

E2. Usuário não tem cadastro:
- O sistema consiste o campo **(R2)**
- O sistema retorna a mensagem “Email invalido”.
- O Caso de Uso é reiniciado.

E3. Senha incorreta:
- O sistema consiste os campos **(R3)**
- O sistema retorna a mensagem “Senha Incorreta”.
- O Caso de Uso é reiniciado.

Regras de Negócio

R1. Os campos da tela de login**(DV1)** são de preenchimento obrigatório.
R2. O usuário precisa estar cadastrado para poder usufruir das funções do sistema.
R3. Os campos precisam estar de acordo com o cadastro do usuário.
R4. O checkbox de “manter logado” diz se na próxima vez em que o usuário abrir o aplicativo, ele será encaminhado para a tela de login**(DV1)** ou para a tela principal **(DV4)**.

UC002 – Cadastrar

Descrição

Este caso de uso serve para o cliente se cadastrar no sistema.

Data View

DV2 – Tela de cadastro do cliente.

Pré-condições
Este caso de uso pode iniciar somente se:
1. O usuário deve ter instalado o aplicativo Android.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. Iniciar o Caso de Uso **UC001 – Login.**

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela **(DV2)**;
- O usuário preenche os campos “Nome”, “Email” e “Senha”;
- O usuário pressiona o botão “Cadastrar”; **(A1)**
- Os dados são verificados pelo sistema. **(E1)(E2)(E3)**;
- O sistema apresenta o **UC001 – Login.**

Fluxos Alternativos

A1. Se usuário clicar em “Back”:
- O sistema apresenta o **UC001 – Login.**

Fluxos de Exceção

E1. Campos obrigatórios não preenchidos:
- O sistema consiste os campos **(R1)**
- O sistema retorna a mensagem “Todos os campos são obrigatórios!”.
- O Caso de Uso é reiniciado.

E2. Usuário já tem cadastro:
- O sistema consiste o campo
- O sistema retorna a mensagem “Email já cadastrado”.
- O Caso de Uso é reiniciado.

Regras de Negócio

R1. Os campos da tela de cadastro **(DV2)** são de preenchimento obrigatório.

UC003 – Recuperar senha

Descrição
Este caso de uso serve para o cliente recuperar sua senha no sistema.

Data View

DV3 – Tela de recuperação.

Pré-condições
Este caso de uso pode iniciar somente se:
1. O usuário deve ter acesso ao sistema.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:
1. Iniciar o Caso de Uso **UC001 – Login.**

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela **(DV3)**;
- O usuário preenche o campo “Email”.
- O usuário pressiona o botão “Salvar”; (A1)
- Os dados são verificados pelo sistema. (E1)(E2)(E3)(R1);
- O sistema apresenta o UC001 – Login.

Fluxos Alternativos

A1. Se usuário clicar em “Back”:
- O sistema apresenta o **UC001 – Login.**

Fluxos de Exceção

E1. Campos obrigatórios não preenchidos:
- O sistema consiste os campos (R1)
- O sistema retorna a mensagem “Todos os campos são obrigatórios!”.
- O Caso de Uso é reiniciado.

E2. Usuário não tem cadastro:
- O sistema retorna a mensagem “Email não cadastrado”.
- O Caso de Uso é reiniciado.

Regras de Negócio

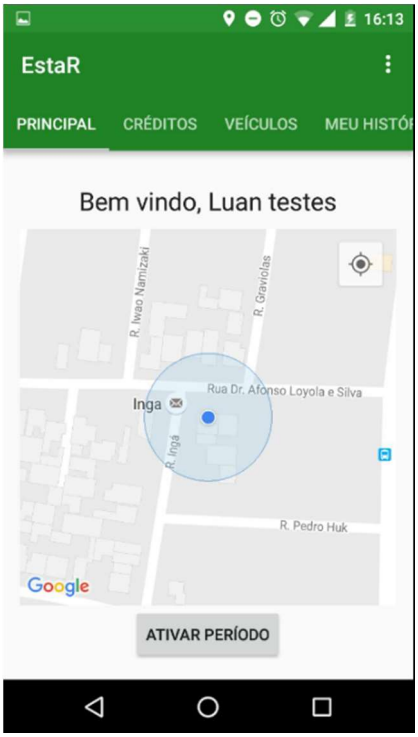
R1. Os campos da tela de cadastro **(DV3)** são de preenchimento obrigatório.

UC004 – Tela principal

Descrição
Este caso de uso serve para apresentar a tela principal do sistema.

Data View

DV4 – Tela principal.



Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter se autenticado ao sistema.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:

1. Iniciar o Caso de Uso **UC005 – Ativar estar.**

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela **(DV4)**;
- O usuário visualiza sua posição no mapa. (R1)
- O usuário pressiona o botão “Ativar período”; (A1)
- O sistema apresenta o UC005 – Ativar estar.

Fluxos Alternativos

A1. Se usuário clicar em outra aba.

- O sistema apresenta a tela da aba selecionada**.**

Regras de Negócio

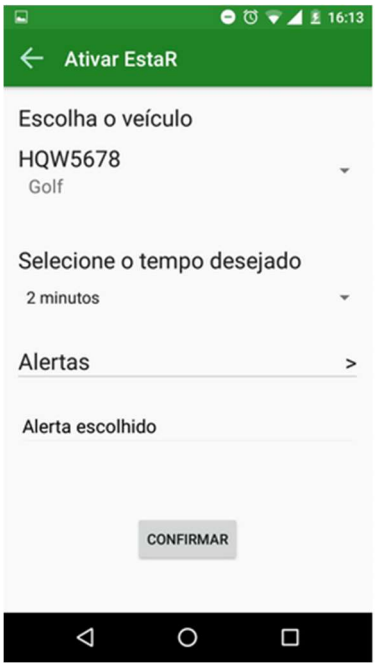
R1. A localização do usuário deve estar habilitada e permitida pelo usuário nas versões mais recentes do Android.

UC005 – Ativar

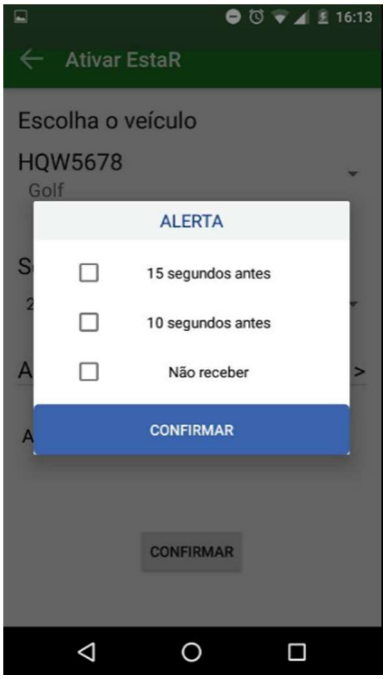
Descrição
Este caso de uso serve para o cliente ativar o seu período de estacionamento.

Data View

DV5 – Tela de ativação.



DV6 – Dialog de alertas



Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter autenticado no sistema.
- Deve ter sido apresentado a tela DV4 – Tela Inicial.

Pós-condições
Após o fim normal deste caso de uso o sistema deve:

1. Iniciar o Caso de Uso **UC006 – Gerenciar estar.**

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV5);
- O usuário seleciona a placa do veículo desejado.
- O usuário seleciona o tempo desejado para manter estacionado.
- O usuário seleciona a placa do veículo desejado.
- O usuário seleciona se deseja receber notificação do sistema.(DV6)
- Os dados são verificados pelo sistema. (E1)(E2)(R1)(R2);
- O sistema apresenta o UC006 – Gerenciar estar.

Fluxos Alternativos

A1. Se usuário clicar em “Back”:
- O sistema apresenta o UC004 – Tela principal.

Fluxos de Exceção

E1. Campos obrigatórios não preenchidos:
- O sistema consiste os campos **(R1)**
- O sistema retorna a mensagem “Todos os campos são obrigatórios!”.
- O Caso de Uso é reiniciado.

E2. Usuário não tem saldo:
- O sistema retorna a mensagem “Saldo insuficiente”.
- O Caso de Uso é reiniciado.

Regras de Negócio

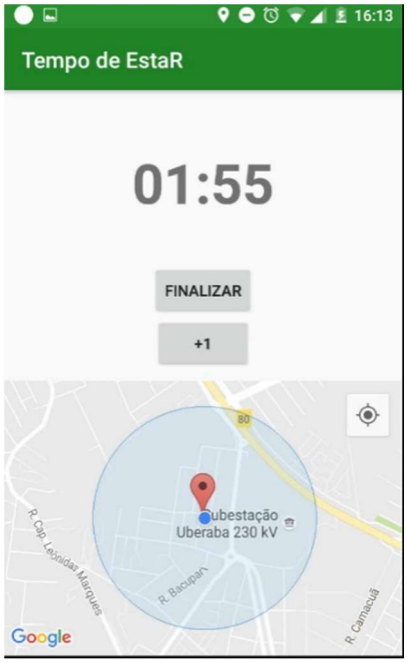
R1. Os campos da tela de cadastro **(DV5)** são de escolha obrigatória.
R2. A localização do usuário deve estar habilitada e permitida pelo usuário nas versões mais recentes do Android.

UC006 – Gerenciar estar

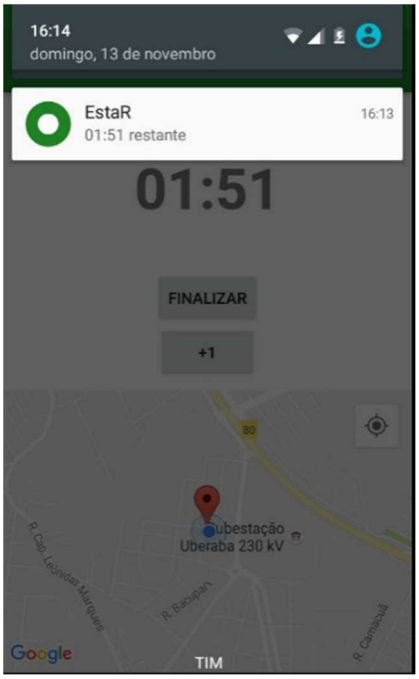
Descrição
Este caso de uso serve para o cliente visualizar o seu período ativo e gerenciar o mesmo.

Data View

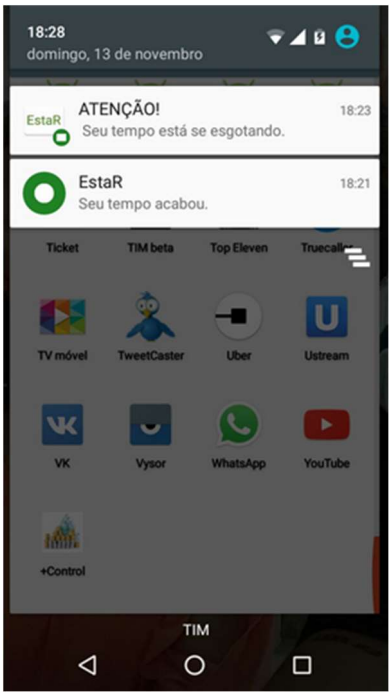
DV7 – Tela de gerenciamento.



DV8 – Notification timer.



DV9 – Notification ao finalizar.



Pré-condições

Este caso de uso pode iniciar somente se:

- O usuário deve ter autenticado no sistema.
- Deve ter iniciado um período no Caso de uso **UC005 – Ativar**.

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela **(DV7)**;
- O sistema cria uma notificação fixa **(DV8)**
- O usuário visualiza no mapa o local onde estacionou.
- O usuário recebe uma notificação quando seu tempo estiver se esgotando ou esgota **(DV9)**.
- O usuário pressiona o botão “Finalizar” e finaliza seu período. (A1)
- O sistema apresenta o UC004 – Tela principal.

Fluxos Alternativos

A1. Se usuário clicar em “+1”:
- O sistema envia ao servidor que o usuário renovou por mais uma hora o seu período.
- É exibida uma mensagem de sucesso “Seu período foi renovado”.

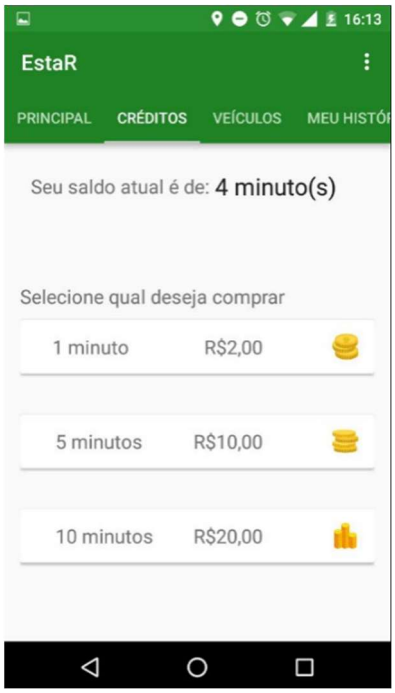
UC007 – Realizar compra

Descrição

Este caso de uso serve para o cliente realizar compra de créditos.

Data View

DV10 – Tela de recuperação.



Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter autenticado no sistema.
- Deve ter sido apresentado a tela DV4 – Tela Inicial.

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV10)(R1);
- O usuário seleciona quais opções deseja comprar.
- O sistema apresenta um alerta de confirmação.
- O usuário clica em “Sim”. (A1)
- Os dados são verificados pelo sistema. (E1).
- O sistema atualiza o saldo do cliente.

Fluxos Alternativos

A1. Se usuário clicar em “Não”:

- A compra é cancelada.
- O caso de uso é reiniciado.

Fluxos de Exceção

E1. Erro na compra:

- O sistema retorna a mensagem “Erro na compra!”.
- O Caso de Uso é reiniciado.

Regras de negócio

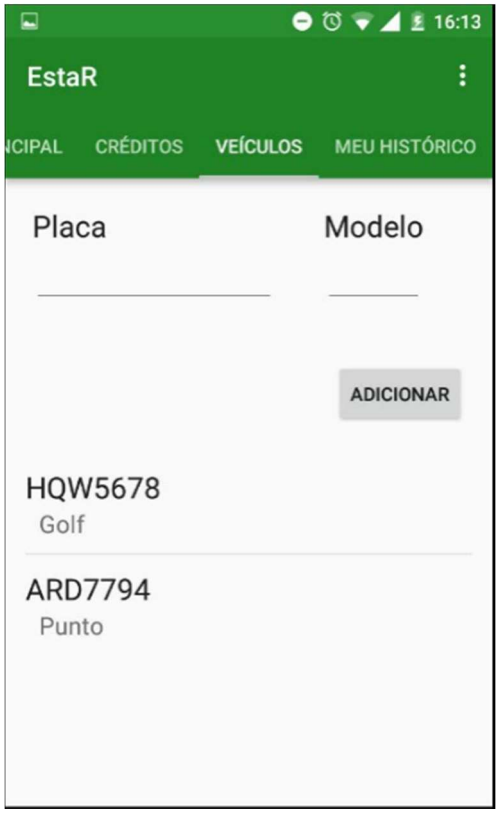
R1. O sistema deve mostrar o saldo atual atualizado do usuário na tela.

UC008 – Cadastrar veículo

Descrição
Este caso de uso serve para o cliente cadastrar veículos que vão ser usados para ativar períodos posteriormente.

Data View

DV11 – Tela de cadastro de veículo.



Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter autenticado no sistema.
- Deve exibir em uma lista o veículo cadastrado.

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV11)
- O usuário preenche os campos da tela (R1).
- O usuário pressiona o botão “Adicionar”.
- O sistema exibe uma mensagem de êxito. (E1)
- O sistema atualiza a lista de veículos do cliente.

Fluxos de Exceção

E1. Erro no cadastro:

- O sistema retorna a mensagem “Erro ao adicionar veículo!”.
- O Caso de Uso é reiniciado.

Regras de negócio

R1. Todos os campos são de preenchimento obrigatório pelo usuário.

UC009 – Visualizar histórico

Descrição
Este caso de uso serve para o cliente visualizar seu histórico de períodos.

Data View

DV12 – Tela de histórico.

Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter autenticado no sistema.

Ator Primário
Cliente

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV12)
- O sistema exibe a lista de histórico do cliente.

APÊNDICE 7 – DESCRIÇÃO DE CASOS DE USO APP FISCAL

UC001 – Listar estar

Descrição
Este caso de uso serve para o fiscal ter uma lista de todos os períodos em uma distância de até 1 km do seu local atual.

Data View

DV1 – Tela de listagem.

Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter o aplicativo instalado em seu dispositivo.

Ator Primário
Fiscal

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV1)
- O sistema lista todos os períodos perto (R1) (R2).
- O sistema atualiza a lista de períodos (R3).
- Ao clicar em um item, é apresentado ao caso de uso UC002 – **Visualizar estar.**
- O caso de uso é finalizado.

Regras de negócio

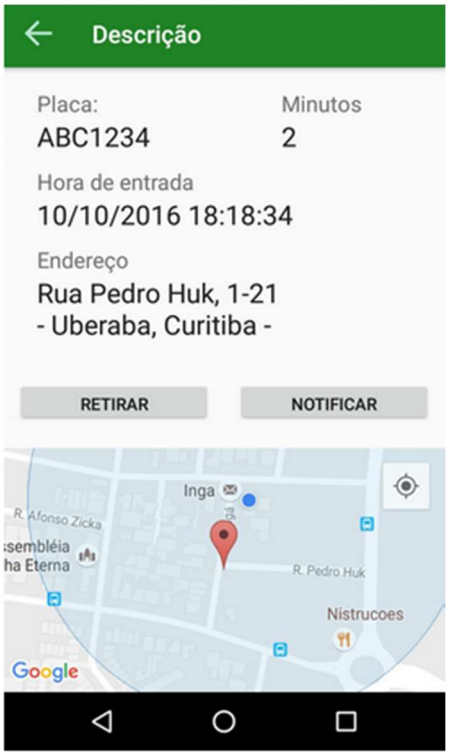
R1. Todos os campos ativos devem ficar com a cor VERDE, e todos os vencidos na cor VERMELHA.
R2. A distancia deve ser de no máximo 1km.
R3. A lista deve ser atualizada automaticamente a cada 30 segundos.

UC002 – Visualizar estar

Descrição
Este caso de uso serve para o fiscal ter todos os detalhes de um determinado período.

Data View

DV2 – Tela de detalhes.



Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter passado pelo **UC001 – Listar estar**.

Ator Primário
Fiscal

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV2)
- O sistema exibe todos os dados do período.(R1)(R2);
- O fiscal pressiona o botão “Notificar”, e um push é enviado para o cliente do veículo.(R3)

- O fiscal pressiona o botão “Retirar”, o período em questão é retirado da lista inicial.
- O caso de uso é finalizado.

Regras de negócio

R1. Todos os dados são obrigatórios.
R2. O mapa deve ter a posição atual e a posição do veículo em questão.
R3. O *push*deve ser enviado para notificar o cliente sobre uma advertência

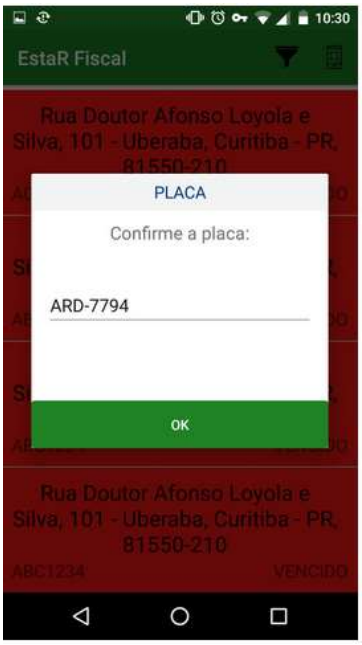
UC003 – Scannear

Descrição
Este caso de uso serve para o fiscal utilizar um OCR para visualização de período de determinado veículo.

Data View

DV3 – Tela de OCR.

DV4 – Tela de confirmação



Pré-condições
Este caso de uso pode iniciar somente se:

- O usuário deve ter passado pelo **UC001 – Listar estar**.
- O usuário deve ter pressionado o item do menu.

Ator Primário
Fiscal

Fluxo de Eventos Principal

- O sistema apresenta a tela (DV3);
- O sistema apresenta a tela de câmera;
- O fiscal posiciona a câmera em frente a placa;
- O fiscal pressiona o retângulo que indica o texto escaneado;
- O sistema exibe a tela (DV4);
- O fiscal pressiona o “OK” **(A1)**;
    - O caso de uso é finalizado;

Fluxos Alternativos

A1. Se o fiscal alterar o texto:

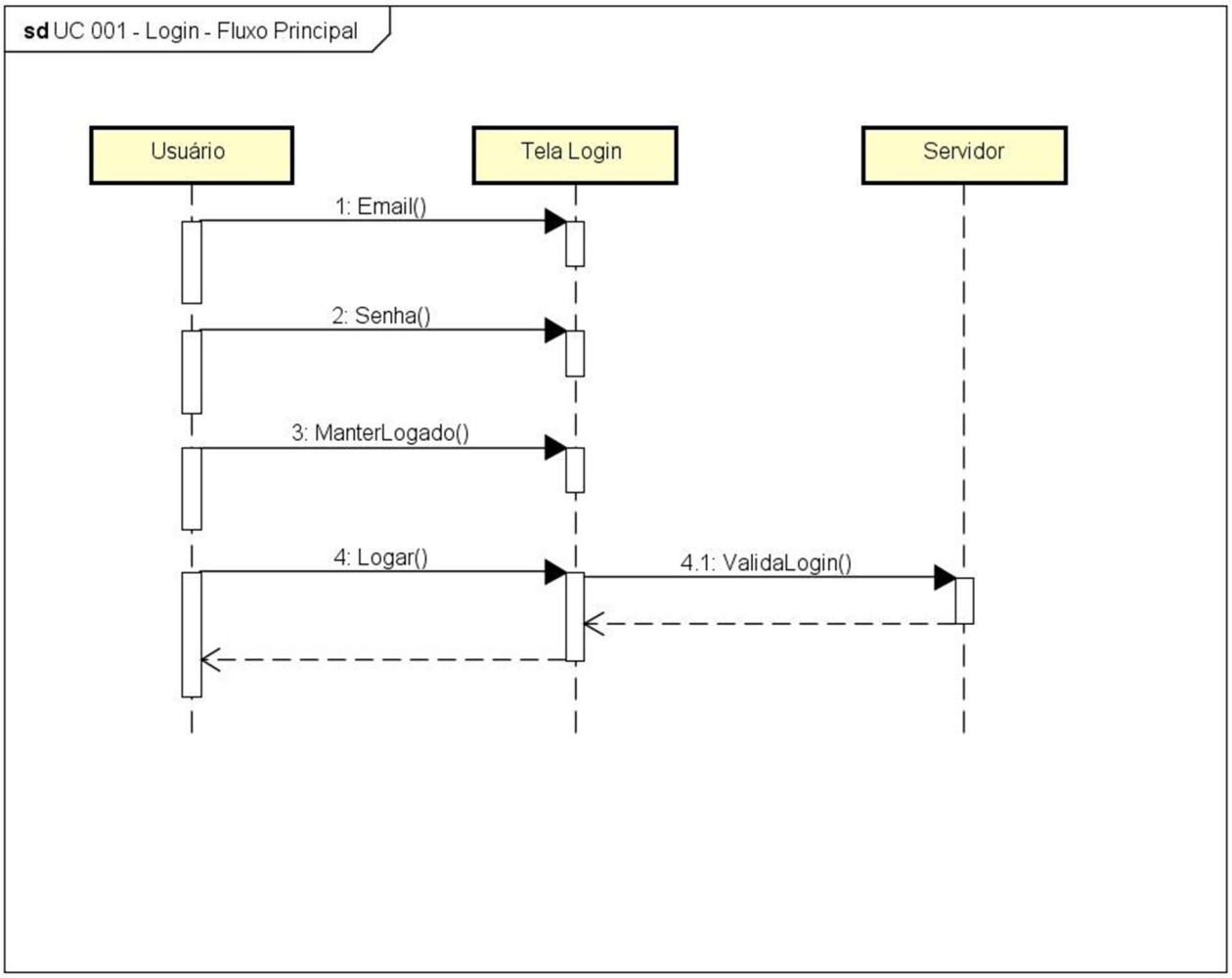
- O fiscal verifica que o texto da placa não esta correto.
- O fiscal altera o texto e pressiona o “OK”;
- O caso de uso é finalizado;

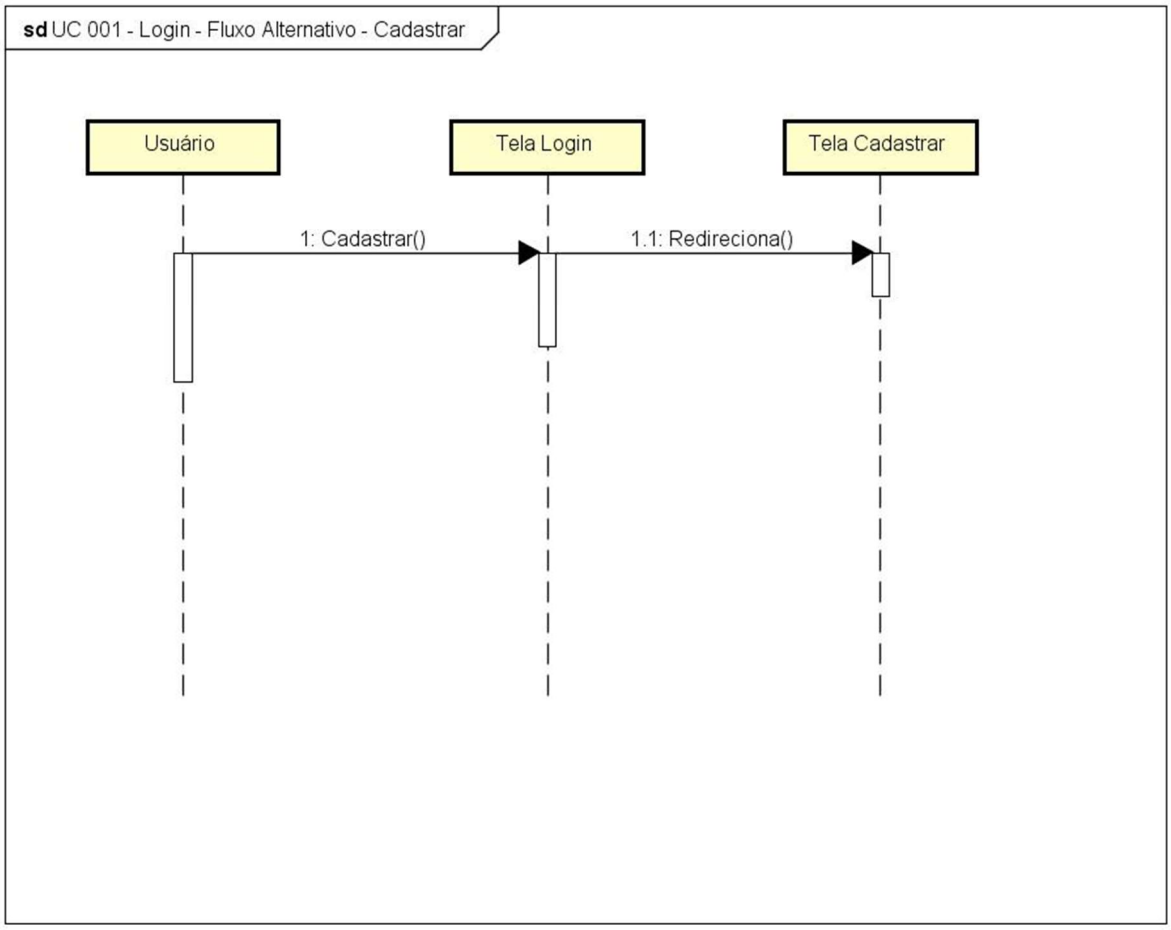
Regras de negócio

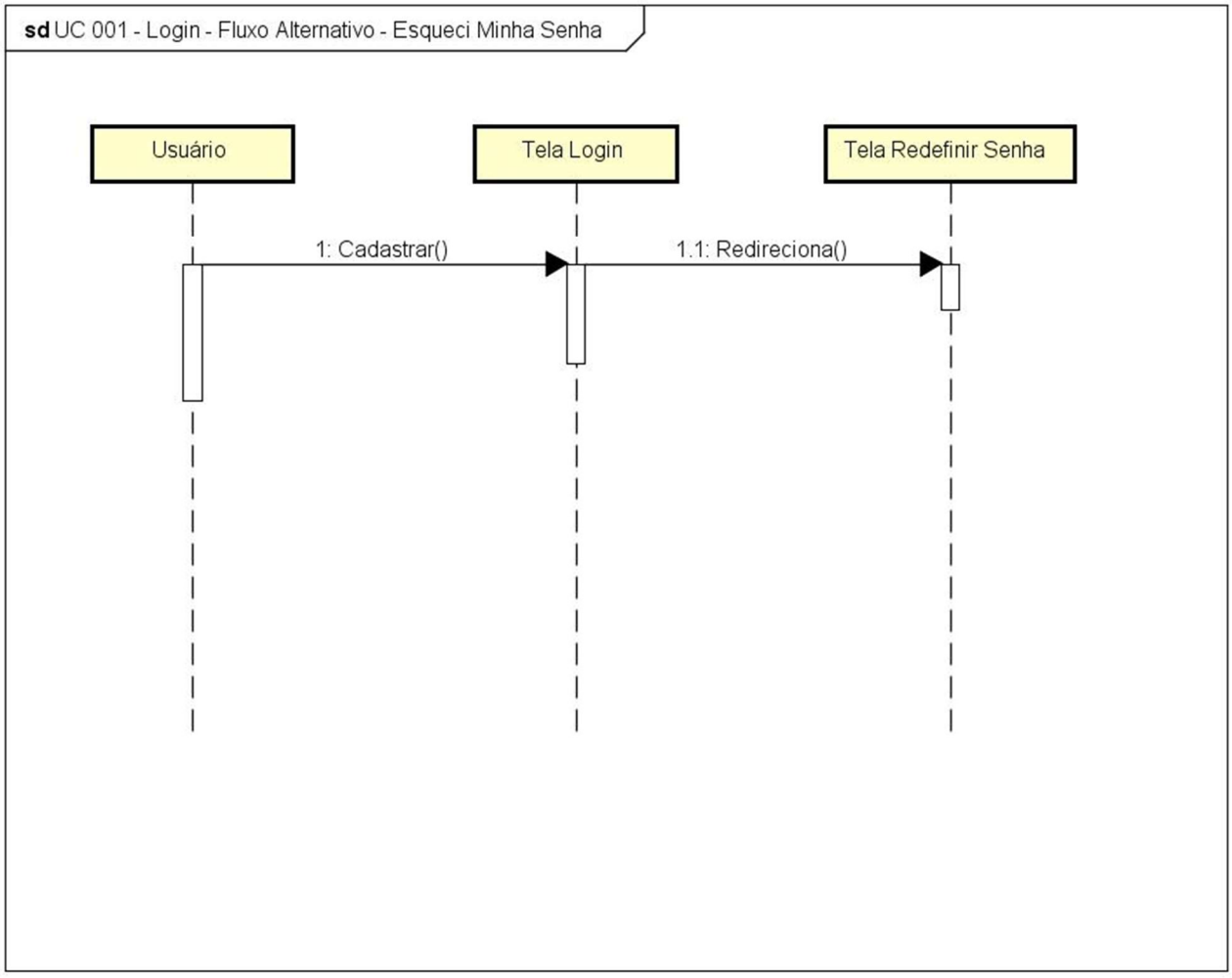
R1. Todos os dados são obrigatórios.
R2. O dispositivo deve ter câmera traseira superior a 8 MP.

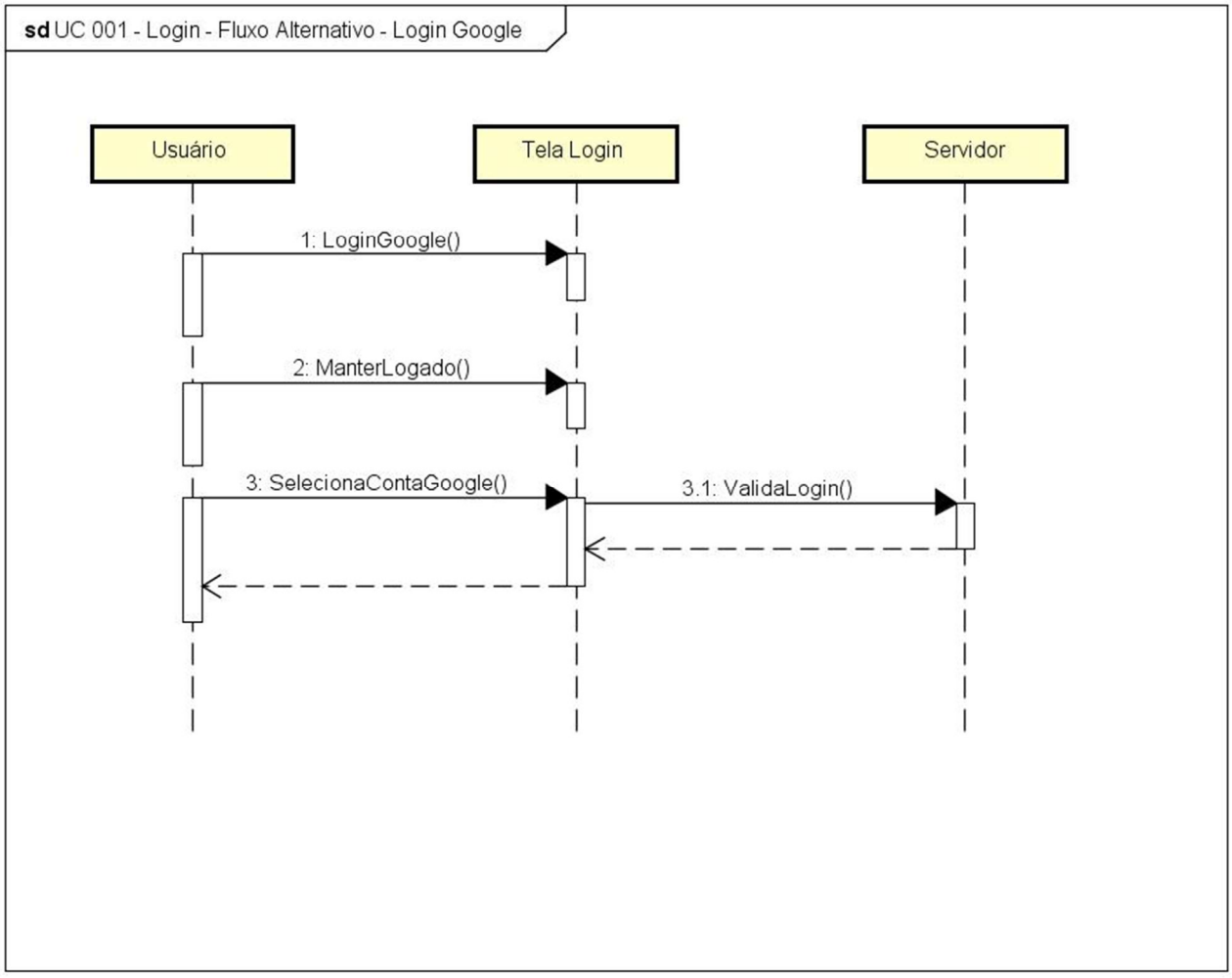
APÊNDICE 8 – DIAGRAMA DE SEQUÊNCIA

APÊNDICE 8.1 – LOGIN

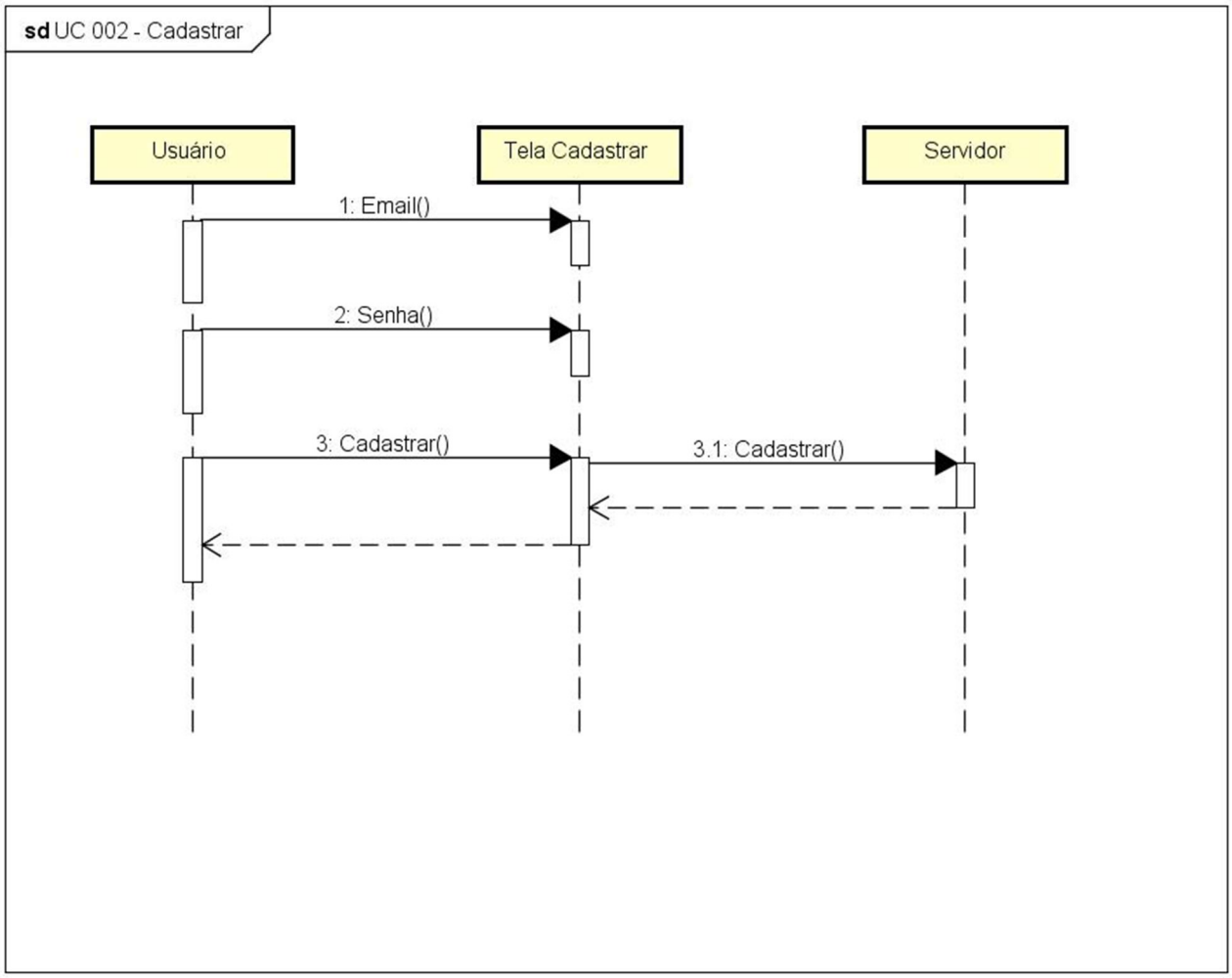



sd UC 001 - Login - Fluxo Alternativo - Cadastrar
Usuário
Tela Login
Tela Cadastrar
1: Cadastrar()
1.1: Redireciona()

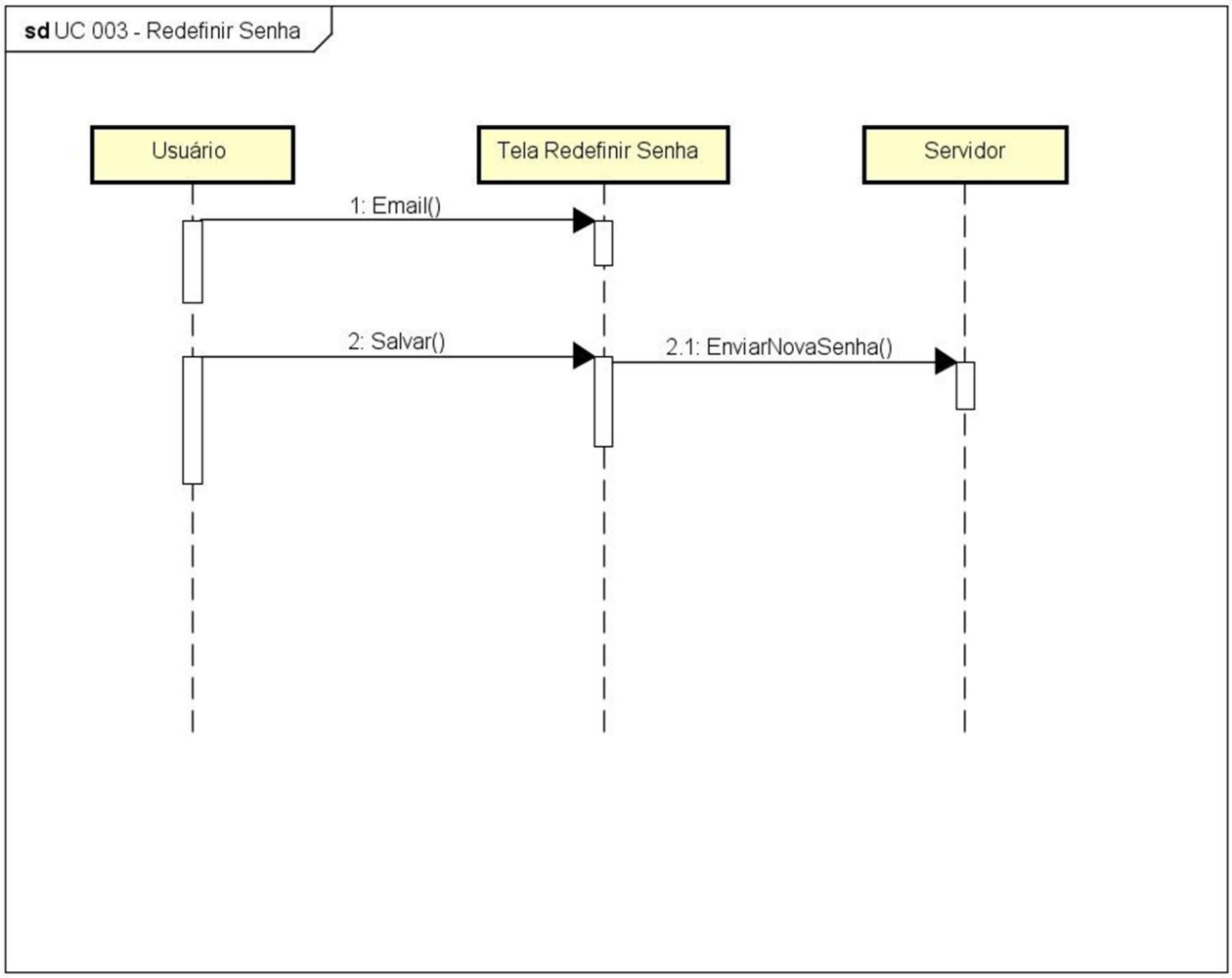

sd UC 001 - Login - Fluxo Alternativo - Esqueci Minha Senha
Usuário
Tela Login
Tela Redefinir Senha
1: Cadastrar()
1.1: Redireciona()


sd UC 001 - Login - Fluxo Alternativo - Login Google
Usuário
Tela Login
Servidor
1: LoginGoogle()
2: ManterLogado()
3: SelecionaContaGoogle()
3.1: ValidaLogin()

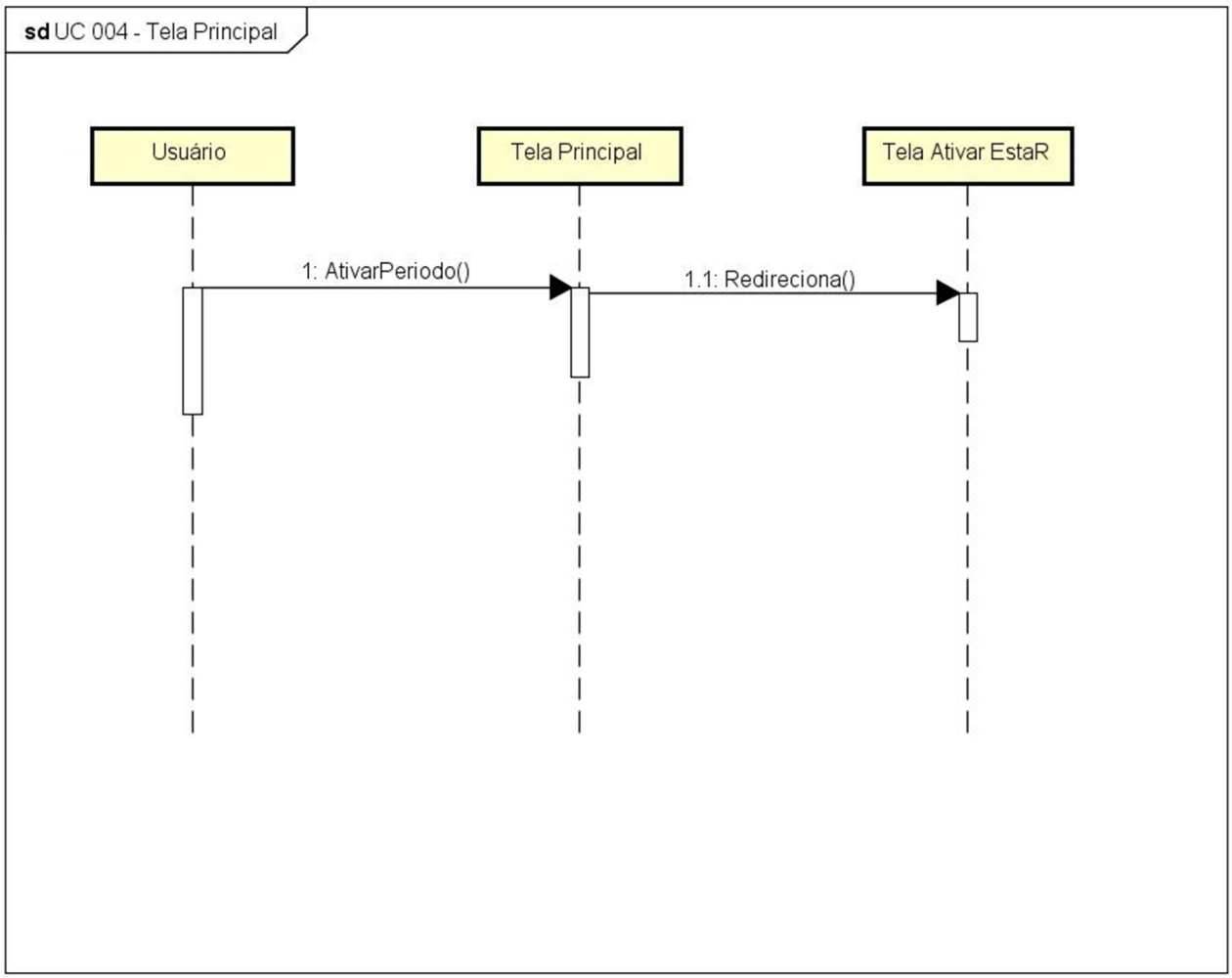
APÊNDICE 8.2 – CADASTRO



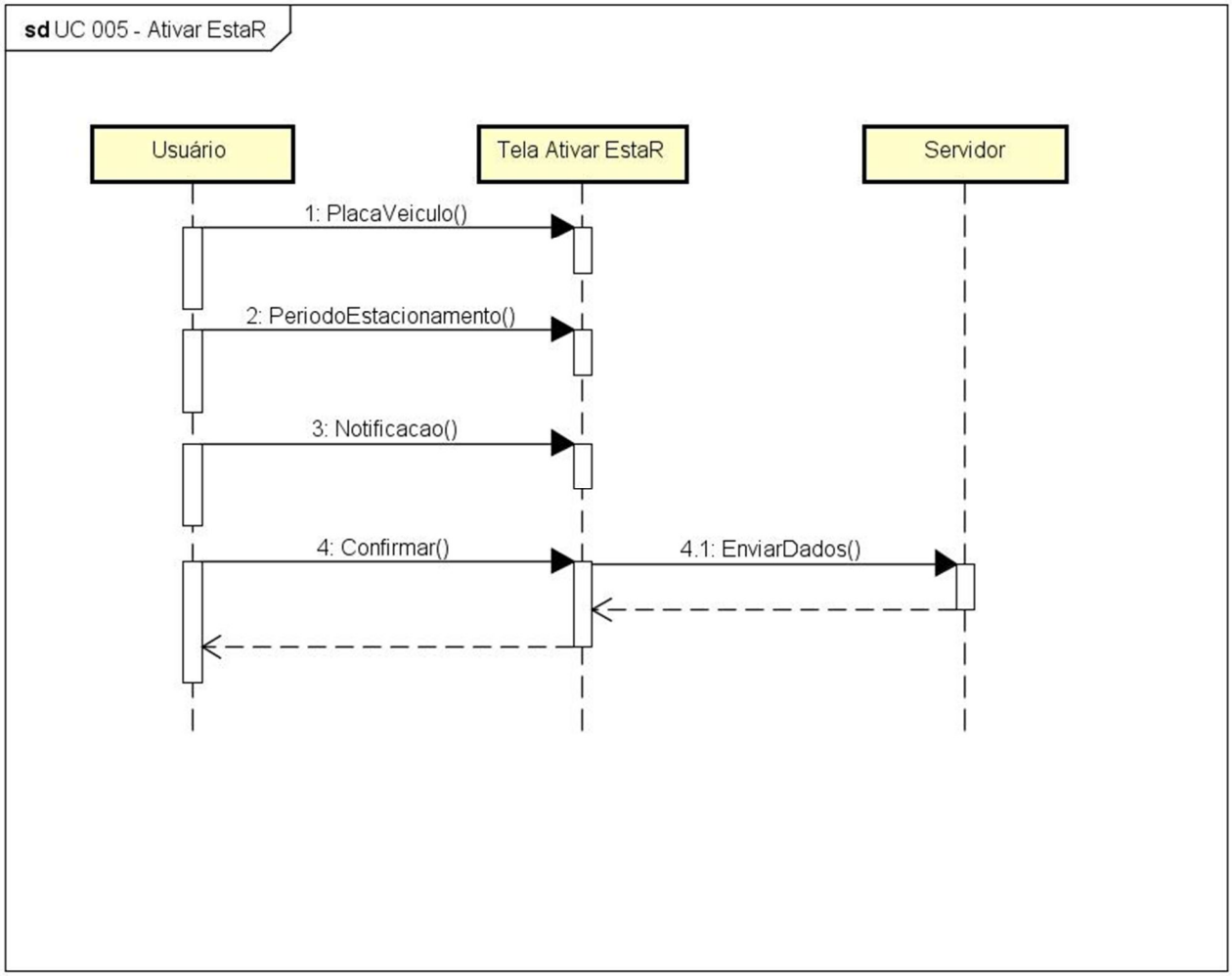
APÊNDICE 8.3 – REDEFINIR SENHA



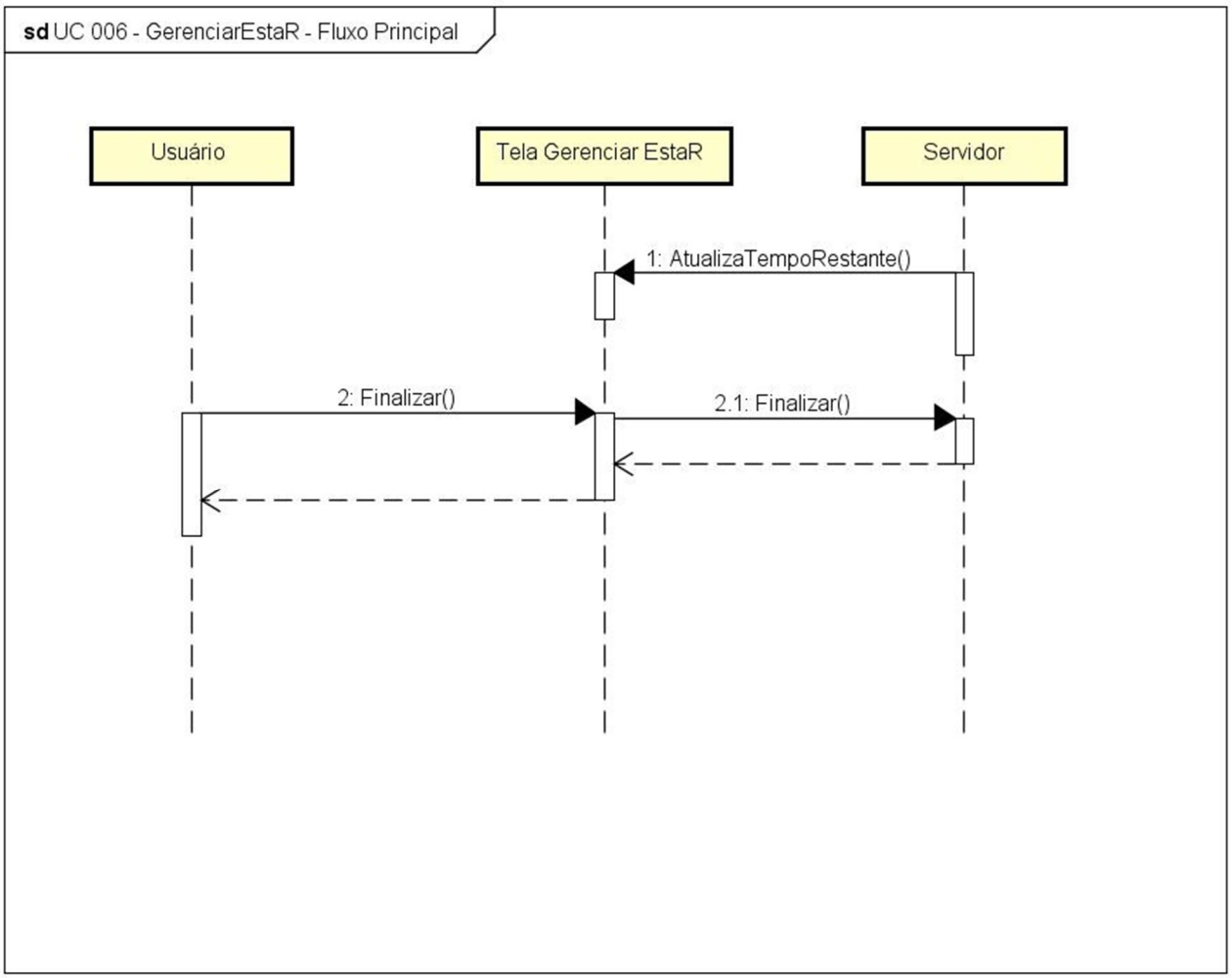
APÊNDICE 8.4 – TELA PRINCIPAL

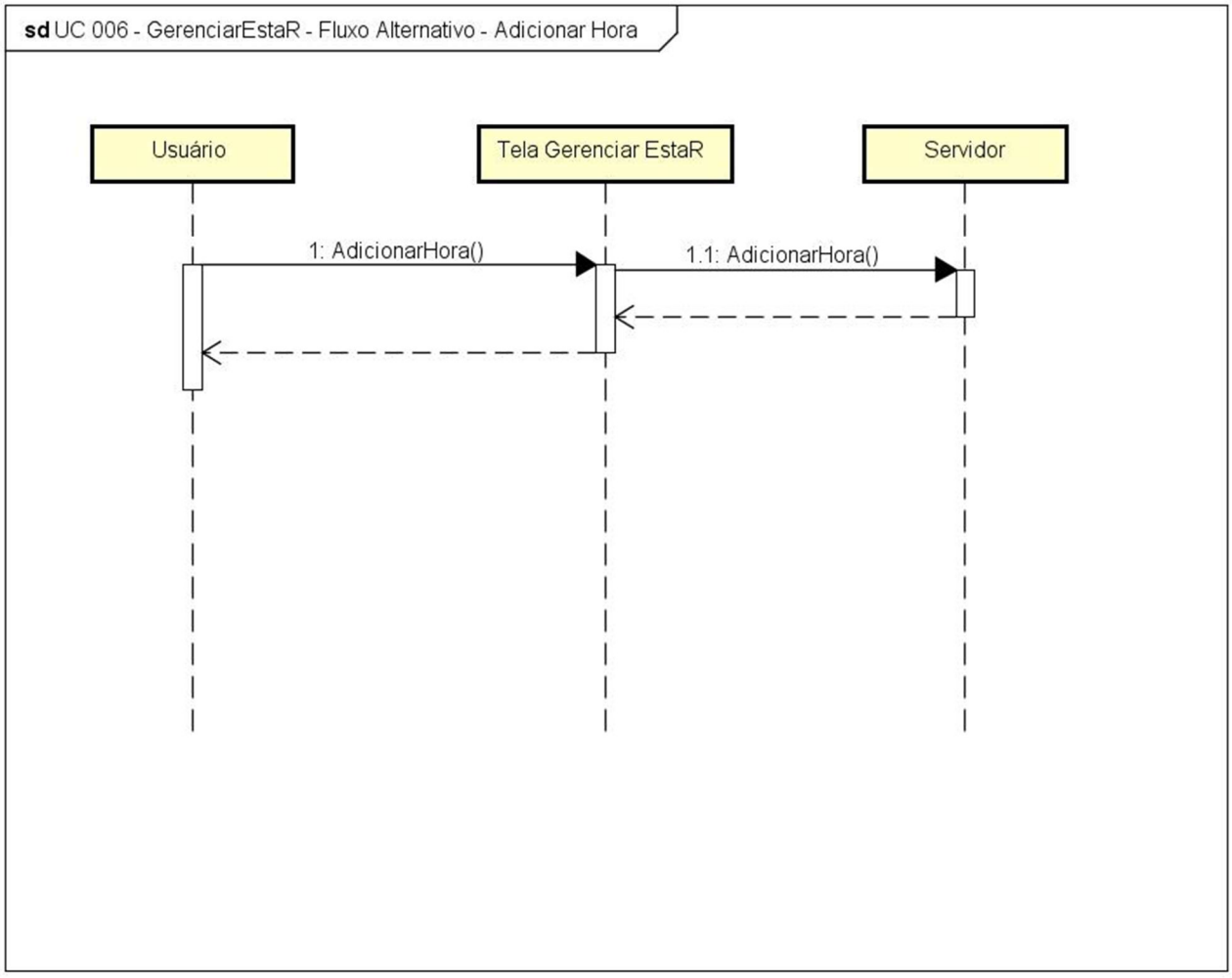


APÊNDICE 8.5 – ATIVAR ESTAR

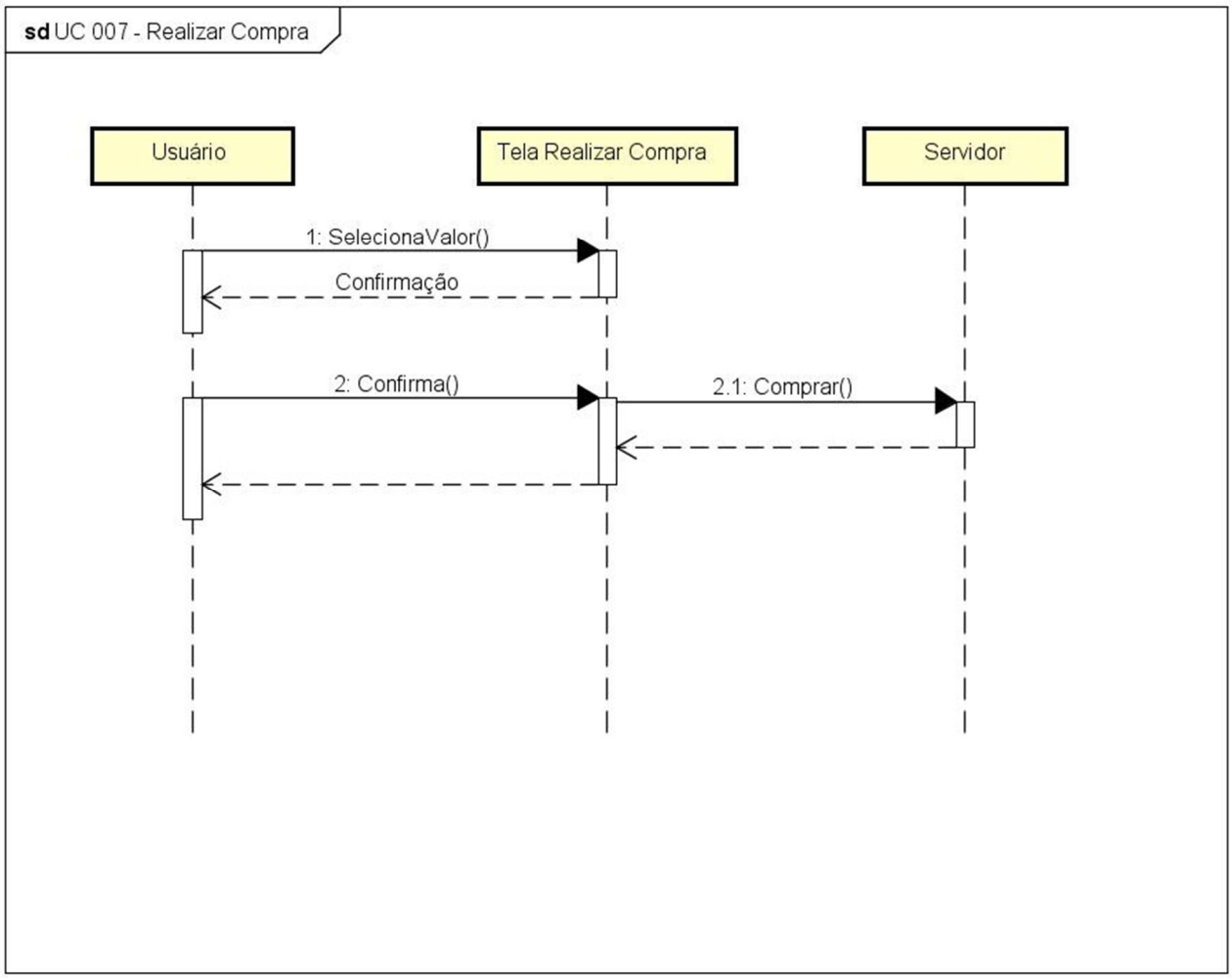


APÊNDICE 8.6 – GERENCIAR ESTAR

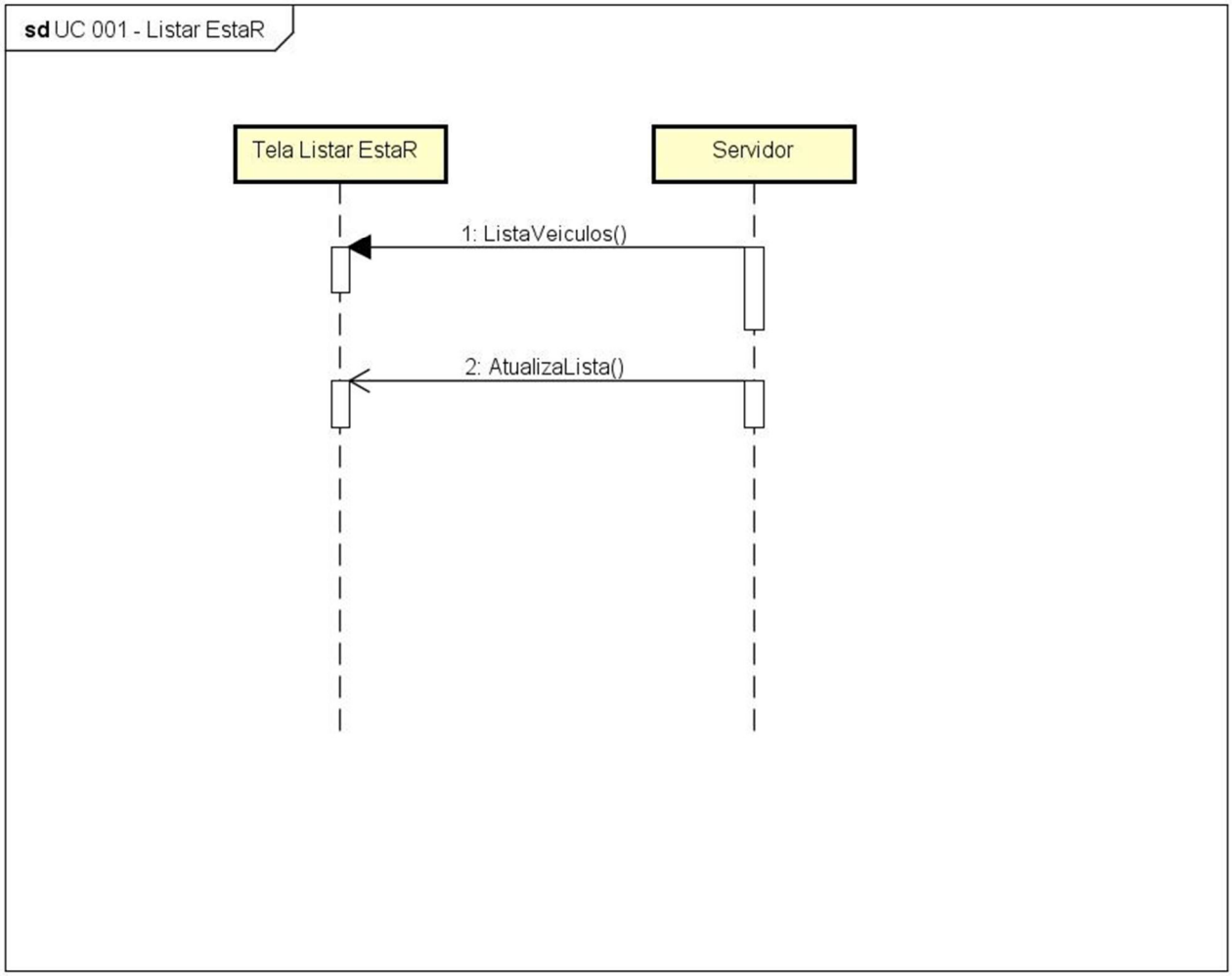



sd UC 006 - GerenciarEstaR - Fluxo Alternativo - Adicionar Hora
Usuário
Tela Gerenciar EstaR
Servidor
1: AdicionarHora()
1.1: AdicionarHora()

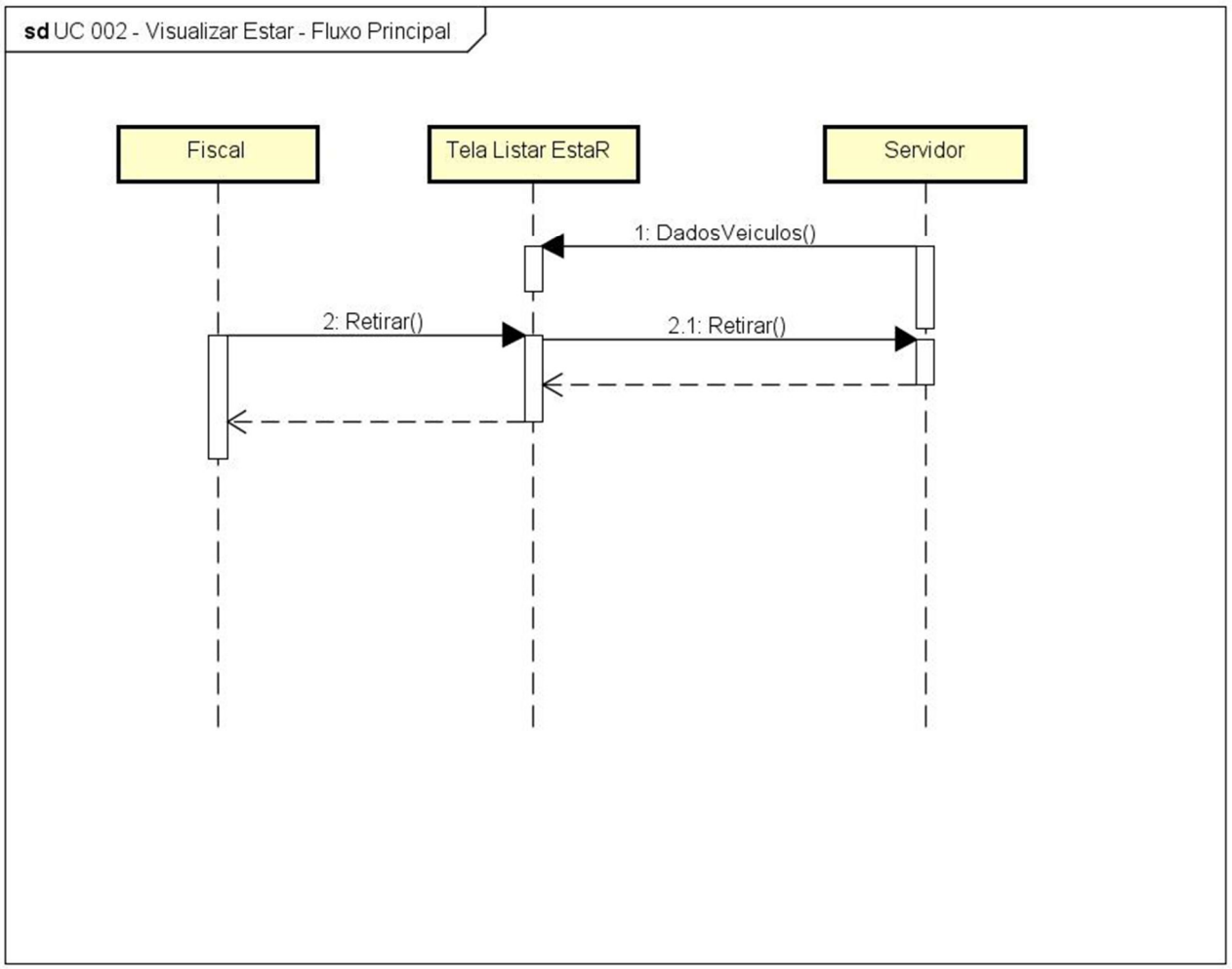
APÊNDICE 8.7 – REALIZAR COMPRA

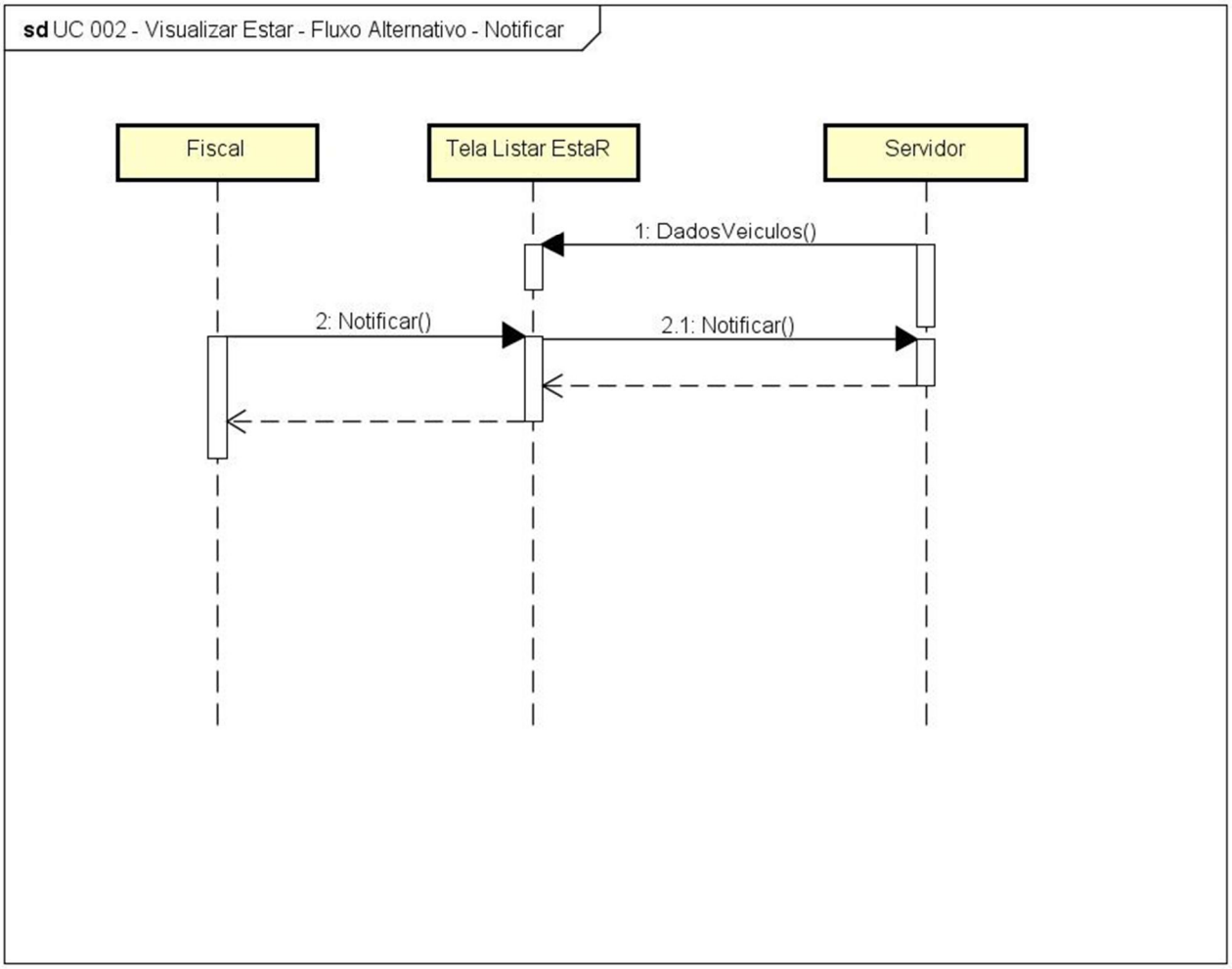


APÊNDICE 8.8 – LISTAR ESTAR



APÊNDICE 8.9 – VISUALIZAR ESTAR




sd UC 002 - Visualizar Estar - Fluxo Alternativo - Notificar
Fiscal
Tela Listar EstaR
Servidor
1: DadosVeiculos()
2: Notificar()
2.1: Notificar()

APÊNDICE 9 – DESCRIÇÃO DE DIAGRAMA DE SEQUÊNCIA APP CLIENTE

Login (APÊNDICE 8.1): Esse diagrama define o funcionamento da autenticação do sistema no aplicativo do cliente. (1) O usuário abre o aplicativo. (2) O sistema apresenta a tela de login. (3) O usuário preenche seus dados cadastrados. (4) O cliente pressiona o botão de logar. (5) O sistema valida o login no servidor. (6) O sistema redireciona o cliente para a tela principal. (6) O usuário pode fazer o cadastro, caso não tenha ainda. (7) O sistema apresenta a tela de cadastro. (8) O usuário pode redefinir sua senha. (9) O usuário pode se autenticar usando sua conta Google.

Cadastro (APÊNDICE 8.2): Esse diagrama define o funcionamento do cadastro do usuário no sistema. (1) O sistema exibe a tela de cadastro. (2) O usuário preenche todos os campos. (3) O usuário finaliza o cadastro. (4) O sistema valida os dados no servidor. (5) O sistema exibe o retorno para o usuário.

Recuperar senha (APÊNDICE 8.3): Esse diagrama define a interação de recuperação de senha. (1) O sistema exibe a tela de recuperação. (2) O usuário preenche o campo com seu email cadastrado. (3) O usuário faz a requisição para o servidor. (4) O servidor verifica os dados. (5) Nova senha é enviada para o email cadastrado.

Navegação da tela principal (APÊNDICE 8.4): Esse diagrama define a seqüência da interação com a tela principal. (1) O sistema apresenta a tela principal. (2) O sistema exibe em um mapa a posição do usuário. (3) O usuário pressiona o botão para ativação de Estar.(4) O sistema redireciona o usuário para a tela de ativação.

Ativar estar (APÊNDICE 8.5): Esse diagrama define o funcionamento da ativação do estar. (1) O sistema apresenta a tela de ativação. (2) O usuário preenche todos os campos. (3) O sistema valida se o usuário possui créditos suficientes. (4) O sistema autoriza a ativação e exibe um contador. (5) O sistema redireciona o usuário para a tela de gerenciamento do estar.

Gerenciar estar (APÊNDICE 8.6): Esse diagrama define o funcionamento do gerenciamento do estar. (1) O sistema exibe a tela com um contador, uma notification também com um contador, e um mapa mostrando a posição atual e a posição onde foi ativado o estar. (2) O usuário pressiona o botão de renovação. (3) O sistema valida se o usuário possui créditos suficientes. (4) O sistema autoriza a renovação. (5) O cliente pressiona o botão de finalizar e finaliza seu período.

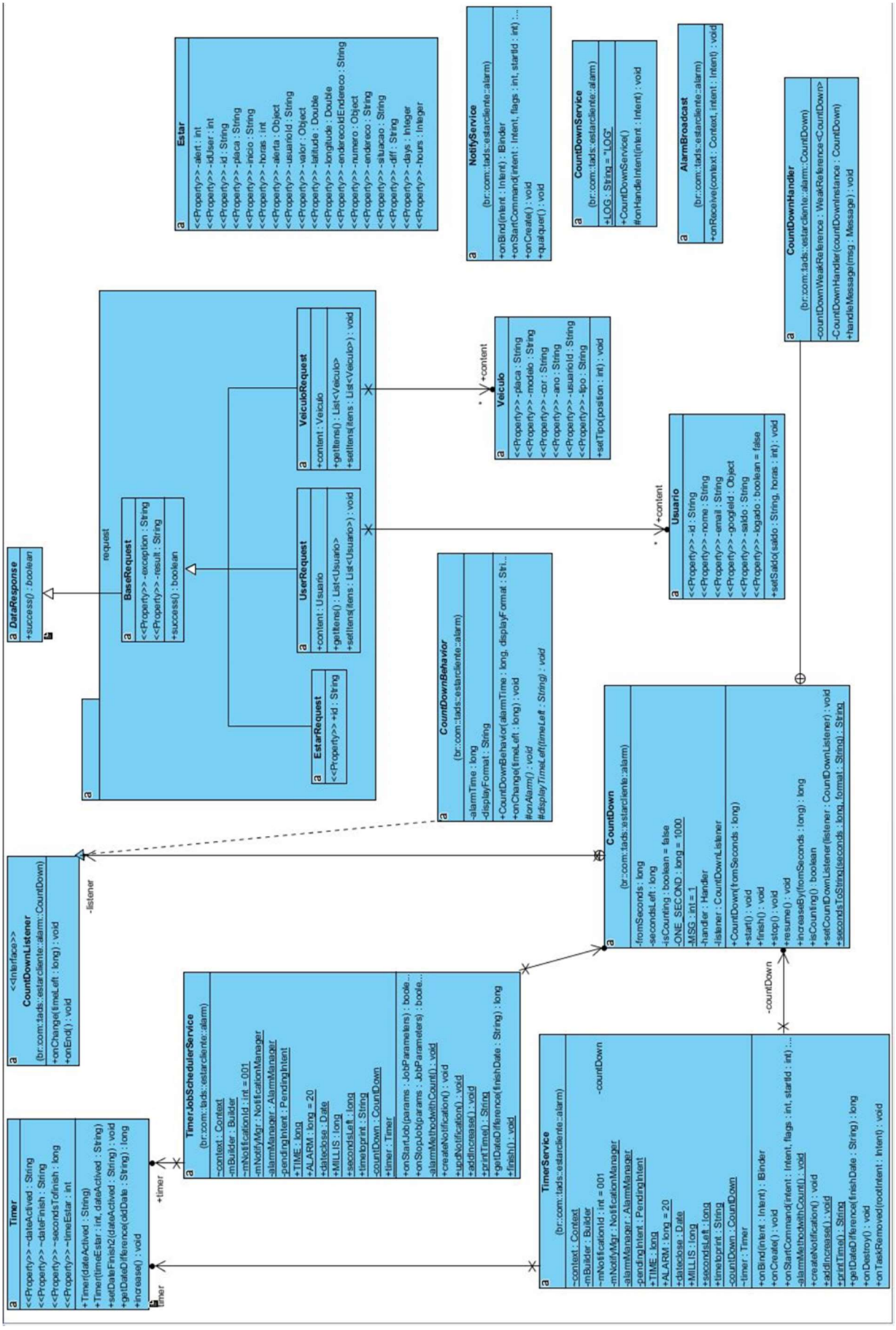
Comprar créditos (APÊNDICE 8.7): Esse diagrama define o funcionamento da compra de créditos. (1) o sistema exibe as opções de compras disponíveis. (2) O usuário seleciona e confirma a compra desejada. (3) O sistema processa a compra e confirma o status. (4) O saldo do usuário é atualizado na tela.

APÊNDICE 10 – DESCRIÇÃO DE DIAGRAMA DE SEQUÊNCIA APP FISCAL

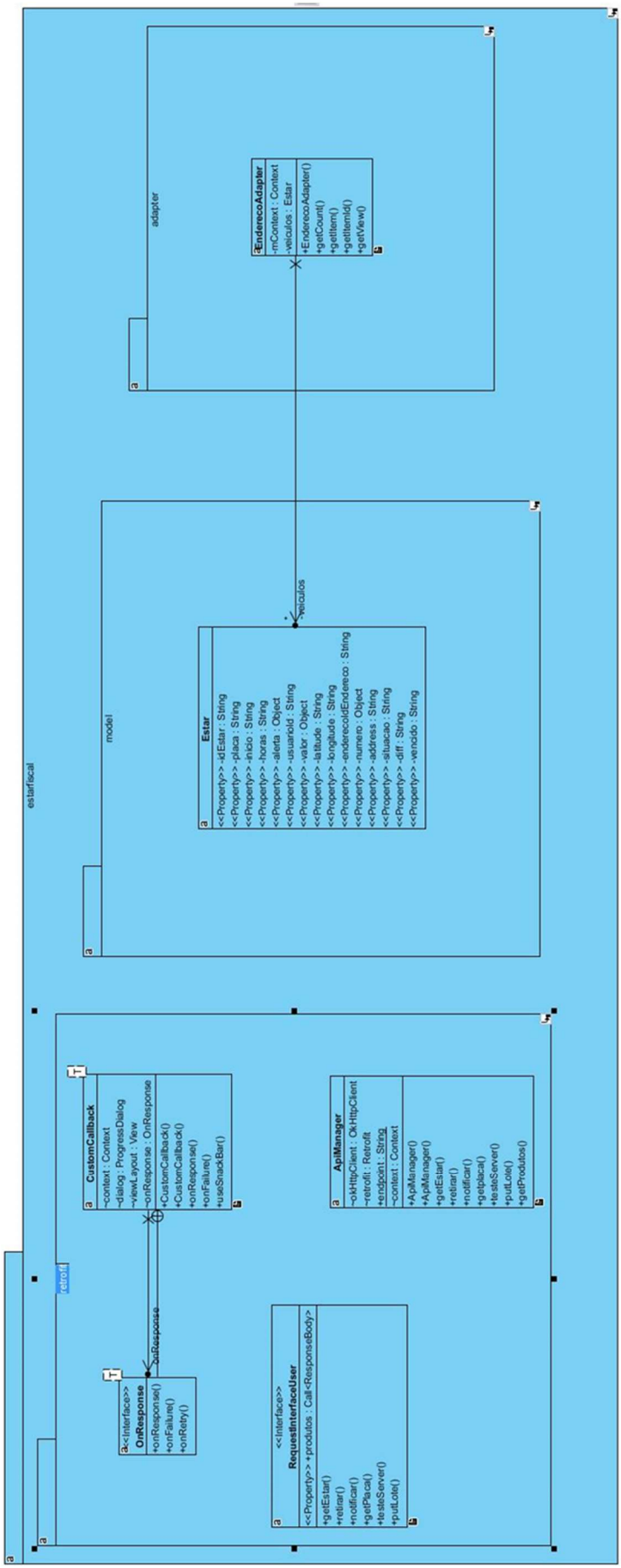
Listar estar (APÊNDICE 8.8): Esse diagrama defina o funcionamento da listagem principal do aplicativo do fiscal. (1) O sistema exibe a tela principal com os períodos abertos em uma distancia de 1km da posição atual do usuário. (2) O usuário define um filtro para apenas vencidos aparecerem. (3) O servidor envia a lista filtrada e o sistema exibe para o usuário. (4) O usuário clica em um item da lista e o sistema redireciona para a tela de visualização.

Visualizar estar (APÊNDICE 8.9): Esse diagrama define a interação entre o usuário e um período de estar. (1) O sistema busca todos os dados do período selecionado no servidor. (2) O sistema apresenta os dados na tela de visualização. (3) O usuário tem a opção de notificar o cliente. (4) O usuário pode retirar o período escolhido de sua lista principal.

APÊNDICE 11- DIAGRAMA DE CLASSE APLICATIVO CLIENTE



APÊNDICE 12- DIAGRAMA DE CLASSE APLICATIVO FISCAL


estarfiscal
adapter
EnderecoAdapter
-mContext : Context
-veiculos : Estar
+EnderecoAdapter()
+getCount()
+getItem()
+getItemId()
+getView()
model
Estar
<<Property>> -idEstar : String
<<Property>> -placa : String
<<Property>> -inicio : String
<<Property>> -horas : String
<<Property>> -alerta : Object
<<Property>> -usuarioId : String
<<Property>> -valor : Object
<<Property>> -latitude : String
<<Property>> -longitude : String
<<Property>> -enderecoldEndereco : String
<<Property>> -numero : Object
<<Property>> -address : String
<<Property>> -situacao : String
<<Property>> -diff : String
<<Property>> -vencido : String
veiculos
retrofit
CustomCallback
~context : Context
~dialog : ProgressDialog
~viewLayout : View
~onResponse : OnResponse
+CustomCallback()
+CustomCallback()
+onResponse()
+onFailure()
+useSnackBar()
OnResponse
+onResponse()
+onFailure()
+onRetry()
onResponse
ApiManager
~okHttpClient : OkHttpClient
~retrofit : Retrofit
+endpoint : String
~context : Context
+ApiManager()
+ApiManager()
+getEstar()
+retirar()
+notificar()
+getplaca()
+testeServer()
+putLote()
+getProdutos()
RequestInterfaceUser
<<Property>> +produtos : Call<ResponseBody>
+getEstar()
+retirar()
+notificar()
+getPlaca()
+testeServer()
+putLote()

APÊNDICE 13 – MODELO FISICO DO BANCO DE DADOS

Banco de Dados: `u952299965_estar`

Estrutura da tabela `Estar`

```
CREATE TABLE IF NOT EXISTS `Estar` (
`idEstar` int(11) NOT NULL AUTO_INCREMENT,
  `placa` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
`inicio` datetime DEFAULT NULL,
  `horas` int(11) DEFAULT NULL,
  `alerta` int(11) DEFAULT NULL,
  `Usuario_id` varchar(11) COLLATE utf8_unicode_ci NOT NULL,
`valor` float DEFAULT NULL,
  `latitude` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `longitude` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `Endereco_idEndereco` int(11) NOT NULL,
  `numero` int(11) DEFAULT NULL,
  `address` varchar(200) COLLATE utf8_unicode_ci NOT NULL,
`situacao` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`idEstar`,`Usuario_id`,`Endereco_idEndereco`),
  KEY `fk_Estar_Usuario1_idx` (`Usuario_id`),
  KEY `fk_Estar_Endereco1_idx` (`Endereco_idEndereco`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=257 ;
```

Estrutura da tabela `Push`

```
CREATE TABLE IF NOT EXISTS `Push` (
  `DeviceId` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `Token` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `Usuario_id` int(11) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Estrutura da tabela `Usuario`

```
CREATE TABLE IF NOT EXISTS `Usuario` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `email` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `senha` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `googleId` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `saldo` int(11) DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=29 ;
```

Estrutura da tabela `Veiculo`

```
CREATE TABLE IF NOT EXISTS `Veiculo` (
`placa` varchar(7) COLLATE utf8_unicode_ci NOT NULL,
  `modelo` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `Usuario_id` varchar(11) COLLATE utf8_unicode_ci NOT NULL,
  `tipo` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`placa`,`Usuario_id`),
KEY `fk_Veiculo_Usuario_idx` (`Usuario_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

APÊNDICE 14 – API APLICATIVO CLIENTE

LOGIN

Nome	Login mobile	URL	http://wsestarapp.esy.es/safe/loginmobile.php
Método	POST	POST	
Input	e-mail	pass	
Output	result	content	exception
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	retorna um array de dados do usuário que acabou de se logar.	em caso de exceção, retorna o motivo da falha

LOGIN COM GOOGLE.

Nome	Login mobile	URL	http://wsestarapp.esy.es/safe/loginGoogle.php
Método	POST	POST	POST
Input	e-mail	Nome	GoogleID

Output	result	content	exception
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	retorna um array de dados do usuário que acabou de se logar.	em caso de exceção, retorna o motivo da falha

CADASTRO DE USUÁRIO.

Nome	addUser	URL	http://wsestarapp.esy.es/setters/addUser.php
Método	POST	POST	POST
Input	Nome	E-mail	Senha
Output	result	exception	
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	

CADASTRO DE VEÍCULO

Nome	addVeiculo	URL	http://wsestarapp.esy.es/setters/addVeiculo.php
Método	POST	POST	POST
Input	idUser	Placa	modelo
Output	result	exception	content
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	retorna, em caso de sucesso, todos os veículos cadastrados pelo usuário, incluindo o mais recente.

COMPRAR CRÉDITOS

Nome	addSaldo	URL	http://wsestarapp.esy.es/setters/addSaldo.php
Método	POST	GET	
Input	horas	idUser	
Output	result	exception	content
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	retorna, em caso de sucesso, todos os dados cadastrados pelo usuário logado.

ATIVAR PERÍODO.

Nome	addAtivar	URL	http://wsestarapp.esy.es/setters/addAtivar.php			
Método	GET	POST	POST			
Input	idUser	placa	address	longitude	latitude	horas
Output	result	exception	id			
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	retorna, em caso de sucesso, o id inserido no banco da ativação realizada.			

BUSCAR VEÍCULOS

Nome	getVeiculos	URL	http://wsestarapp.esy.es/getters/getVeiculos.php
Método	GET		
Input	idUser		
Output	result	exception	content
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	retorna, em caso de sucesso, todos os veículos cadastrados pelo usuário.

HISTÓRICO DE USUÁRIO

Nome	getHistorico	URL	http://wsestarapp.esy.es/getters/getHistorico.php
Método	GET		
Input	idUser		
Output	result	exception	content
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	retorna, em caso de sucesso, todos os períodos antigos do usuário.

APÊNDICE 15 – API APLICATIVO FISCAL

LISTA PRINCIPAL

Nome	getEstar	URL	http://wsestarapp.esy.es/setters/getEstarFiltro.php?filter =0
Método	POST		
Input	filter		
Output	result	exception	
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	

RETIRAR DA LISTA

Nome	getEstar	URL	http://wsestarapp.esy.es/setters/updStatus.php?IdEstar =0
Método	GET		
Input	idEstar		
Output	result	exception	
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	

NOTIFICAR USUÁRIO

Nome	push	URL	http://wsestarapp.esy.es/setters/push.php?UserId =0
Método	GET		
Input	UserId		
Output	result	exception	
Descrição	retorna um valor boolean para	em caso de exceção, retorna o	

	indicar se houve sucesso ou falha na requisição.	motivo da falha.	

DETALHES PERÍODO

Nome	getDetalhes	URL	http://wsestarapp.esy.es/setters/getDetalhes.php?idEstar =0
Método	GET		
Input	idEstar		
Output	result	exception	content
Descrição	retorna um valor boolean para indicar se houve sucesso ou falha na requisição.	em caso de exceção, retorna o motivo da falha.	retorna, em caso de sucesso, todos os detalhes do período selecionado.

APÊNDICE 16 – GLOSSÁRIO DE DADOS DAS CLASSES

USUÁRIO

Atributo	Tipo	Descrição
id	String	Código sequencial que identifica os usuários.
nome	String	Nome do usuário cadastrado.
email	String	Email do usuário cadastrado.
googleId	String	Código de identificação caso o usuário esteja utilizando sua conta Google.
saldo	String	Seu saldo disponível para compra.
logado	boolean	Atributo que identifica se o usuário se manterá logado.

ESTAR

Atributo	Tipo	Descrição
id	String	Código sequencial que identifica os períodos.
idUser	String	Código sequencial que identifica os usuários.
alert	Int	Tempo escolhido pelo usuário para soar um alerta.
placa	String	Placa do veículo utilizado para ativação.
inicio	String	Inicio da ativação.
horas	Int	Quantidade de horas escolhida pelo usuário para validade do período.
valor	String	Valor do estar ativado (Não usado no protótipo).
latitude	Double	Posição do local onde foi ativado o serviço.
longitude	Double	Posição do local onde foi ativado o serviço.
idEndereco	String	Código sequencial que identifica os endereços (Não usado no protótipo).

numero	String	Complemento do endereço.
endereco	String	Endereço do local de ativação.
situacao	String	Se foi ou não finalizado pelo usuário.
diff	String	Diferença entre o dia da ativação e o dia atual, em minutos.
days	Int	Diferença entre o dia da ativação e o dia atual, em dias.
hours	Int	Diferença entre o dia da ativação e o dia atual, em horas.

VEICULO

Atributo	Tipo	Descrição
placa	String	
modelo	String	
cor	String	
ano	String	
usuarioId	String	
tipo	String	

COUNTDOWN

Atributo	Tipo	Descrição
fromSeconds	Long	`Atributo com o valor inicial de segundos em que trabalhará o contador.`
secondsLeft	Long	Atributo com o valor do contador sendo decrementado a cada segundo.
handler	Handler	Thread que executa o funcionamento do contador.
listener	CountDownListener	Notificado quando o contador fizer um decremento ou incremento.
`MSG`	Int	Message usado pelo Handler na sua regra de negocio.

COUNTDOWNBEHAVIOR

Atributo	Tipo	Descrição
alarmTime	Long	Tempo escolhido pelo usuário para soar o alarme.
displayFormat	String	Modo como será exibido o contador.

TIMER

Atributo	Tipo	Descrição
dateActived	String	Data de ativação, para fazer a validação de tempo quando o usuário fechar a activity.
dateFinish	String	Data que irá finalizar. É feito uma conta com o tempo comprado e a data de ativação. É usado para controlar o contador caso haja problemas.
secondsToFinish	long	Segundos para finalizar, é usado para administrar as regras de negocio da activity de gerenciamento de estar.
timeEstar	int	Tempo restante.

TIMESERVICE

Atributo	Tipo	Descrição
context	Context	Contexto onde estará localizado.
mBuilder	Builder	Objeto que constrói uma *notification*.
mNotificationId	Int	Código de identificação da *notification*.
mNotifyMgr	NotificationManager	Gerenciador da *notification*.
alarmManager	AlarmManager	Responsável pela criação do alarme que será tocado.
pendingIntent	PendingIntent	Responsável pela ação que irá gerar ao pressionar a notification.
TIME	Long	Tempo, em segundos, utilizado pelo contador.

ALARM	Long	Tempo escolhido pelo usuário para tocar o alarme.
countDown	CountDown	Objeto que gera o contador.
timer	Timer	Objeto que salva os dados do periodo para que o contador funcione de maneira correta.
secondsLeft	Long	Tempo restante para finalizar.

APÊNDICE 17 – GLOSSÁRIO DE DADOS MODELO LÓGICO

Estar

Comentários da tabela: Estar

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
idEstar	int(11)	Não			Código sequencial que identifica os periodos do sistema.	
placa	varchar(45)	Sim	*NULL*		Atributo contendo a placa do veículo utilizado.	
inicio	datetime	Sim	*NULL*		Atributo com a hora em que foi ativado o período.	
horas	int(11)	Sim	*NULL*		Valor de horas comprados pelo usuário.	
alerta	int(11)	Sim	*NULL*		Se o usuário deseja ou nao receber alertas.	
Usuario_id	varchar(11)	Não		Usuario -> id	Código sequencial que liga com usuário do sistema	
valor	float	Sim	*NULL*		Valor de compra gasto pelo usuário.	
latitude	varchar(200)	Sim	*NULL*		Latitude do usuário na hora da ativação.	
longitude	varchar(200)	Sim	*NULL*		Longitude do usuário na hora da ativação.	
Endereco_idEndereco	int(11)	Não			Código sequencial que liga com a tabela de endereços do sistema	
numero	int(11)	Sim	*NULL*			
address	varchar(200)	Não			Endereço do local estacionado.	
situacao	int(11)	Não	0		Atributo que diz respeito ao periodo finalizado ou não.	

Veiculo

Comentários da tabela: Veiculo

Coluna	Tipo	Nulo	Padrão	Links para	Comentários	MIME
placa	varchar(7)	Não			Atributo para a placa do veículo.	
modelo	varchar(45)	Sim	*NULL*		Atributo para o modelo do veículo.	
Usuario_id	varchar(11)	Não		Usuario -> id	Código sequencia que liga com a tabela Usuario.	
tipo	varchar(500)	Não				

Push

Comentários da tabela: Push

Coluna	Tipo	Nulo	Padrão	Comentários	MIME
DeviceId	varchar(100)	Sim	*NULL*	Atributo que possuio o valor de identificação do dispositivo do usuário.	
Token	varchar(500)	Não		Atributo com a chave de dispositivo para notificação.	
Usuario_id	int(11)	Não		Código sequencia que liga com a tabela Usuario.	

Usuario

Comentários da tabela: Usuario

Coluna	Tipo	Nulo	Padrão	Comentários	MIME
id	int(11)	Não		Código sequencial que identifica os usuários no sistema.	
nome	varchar(45)	Sim	*NULL*	Atributo com o nome do usuário cadastrado.	
email	varchar(45)	Sim	*NULL*	Atributo com o email do usuário cadastrado.	
senha	varchar(45)	Sim	*NULL*	Atributo com a senha do usuário cadastrado.	
googleId	varchar(200)	Sim	*NULL*	Código de identificação do usuário na conta Google.	
saldo	int(11)	Sim	0	Saldo disponivel para compra.	

À Deus e ao Bob.