UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

JUN HONG SILVA

DESENVOLVIMENTO DE APLICATIVO PARA FOOD TRUCKS

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2018

JUN HONG SILVA

DESENVOLVIMENTO DE APLICATIVO PARA FOOD TRUCKS.

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Especialista em Desenvolvimento para Dispositivos Móveis".

Orientador: Prof.ª Paulo Mauricio de Lucchi Bordin.

CURITIBA
2018

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

“Desenvolvimento de Aplicativo para Food Trucks”

por

“Jun Hong Silva”

Este Trabalho de Conclusão de Curso foi apresentado às 19:22 do dia 20 de fevereiro de 2018 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

_____ Prof. Paulo Mauricio de Lucchi Bordin **(Presidente/Orientador - externo)**	_____ Profa. Maria Claudia Figueiredo Pereira Emer **(Avaliador 1 – UTFPR/Curitiba)**
_____ Prof. Robson Ribeiro Linhares **(Avaliador 2 – UTFPR/Curitiba)**	

“A Ata de Aprovação assinada encontra-se na Coordenação do Curso.”

RESUMO

SILVA, Jun Hong. Desenvolvimento de aplicativo para Food Trucks. 2018. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2018

Com a explosão da tendência de food trucks desde sua introdução no mercado, alguns países se mantiveram forte com crescimento esporádico, enquanto outros sofreram um declínio significativo pela introdução de legislações mais rigorosas. Porém, por se tratar de veículos automotivos, surge a necessidade de identificar suas localizações.

Esta pesquisa apresenta uma abordagem teórico-conceitual e as etapas de desenvolvimento de um aplicativo para o mercado de food trucks (veículos de médio à grande porte equipados para cozinhar e comercializar alimentos), fornecendo um mapeamento de seus veículos, informações detalhadas de seus estabelecimentos, horário de funcionamento, cardápios, galeria do estabelecimento e busca por regiões.

Apresenta conceitos de criação, desde apresentação de diagramas, fluxogramas ao desenvolvimento de um protótipo funcional. Discute os conceitos acerca do mercado, obstáculos, quadro comparativos de aplicações similares existentes. Traz como resultado do estudo um panorama de uma aplicação mais moderna, explorando carências de concorrentes e integrando serviços mais atuais.

Palavras-chaves: Food truck, Aplicativos Mobile, Swift.

ABSTRACT

SILVA, Jun Hong. Desenvolvimento de Aplicativo para Food Trucks. 2018. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2018

With the explosion of the food trucks trend since its introduction in the market, some countries have remained strong with sporadic growth, while others have suffered a significant decline by the introduction of stricter legislation. However, because they are automotive vehicles, the need arises to identify their locations.

This research presents a theoretical-conceptual approach and the steps of developing an application for the food trucks market, providing a mapping of their vehicles, detailed information of their establishments, schedules of operation, menus, gallery of the establishment and search by regions.

It presents concepts of creation, from presentation of diagrams, flowcharts to the development of a functional prototype. Discusses the concepts about market, obstacles, comparative chart of existing similar applications. It brings as a result of the study a panorama of a more modern application, exploring shortages of competitors and integrating more current services.

Keywords: Food Truck, Mobile Application, Swift.

LISTA DE FIGURAS

LISTA DE SIGLAS

APIs Aplication Programming Interface (Interface de Programação de Aplicativos).
UML Unified Modeling Language (Linguagem de Modelagem Unificada).
APP Aplicativo
POD Arquivo podfile
BAAS Backend-as-a-Service

LISTA DE ACRÔNIMOS

JSON JavaScript Object Notation (Notação de Objeto JavaScript)

SUMÁRIO

1 Introdução

1.1 Contextualização

Segundo pesquisa (FoodTruckEmpire, 2017) realizada em abril de 2017 nos EUA, composta pelas cidades da California, Florida, New York, Texas e Oregon, foi coletada dados de 300 donos de food trucks com mais de 2 anos de atuação para mensurar a média de lucro anual. Para tal pesquisa, foi oferecido 5 opções de valores e orientados a escolherem o mais próximo do lucro atingido. De acordo com a pesquisa, mais de 85% geraram um lucro superior à 100.000,00 dólares.

Já no Brasil, a legislação de food truck tem se multiplicado significativamente. O estado do Rio de Janeiro, recentemente, criou uma política estadual de incentivo às feiras gastronômicas e comércio de alimentos em food trucks, trailers, vans, caminhões e outros. A ideia, é incentivar parcerias com municípios, criando eventos conjuntos com a iniciativa privada nos quais possam trabalhar os food trucks (ECOMANDA, 2016). Outras capitais também aderiram rapidamente à tendência. As peculiaridades das leis podem, contudo, variar.

Em Porto Alegre, a lei de food truck proíbe os veículos de estacionarem em frente a danceterias e casas noturnas, algo que na verdade ocorria antes da lei (PREFEITURA DE PORTO ALEGRE, 2016).

Na cidade de Curitiba, com a definição da legislação aprovada em 2015 (PREFEITURA DE CURITIBA, 2015), há uma distância mínima regulamentar em relação a outras feiras gastronômicas e pontos de comércio de alimentos. Os caminhões não podem ficar a menos de 200 metros desses concorrentes, salvo em horários ou dias nos quais esses outros estabelecimentos não funcionem.

Outras grandes cidades espalhadas pelo Brasil, algumas delas onde caminhões e food trucks já operam, incluíram leis e determinam prazos para que proprietários desses food trucks e barracas possam se adequar.

1.2 Motivação e justificativa

No Brasil, houve um declínio significativo desde a explosão no início de 2014 para 2017 (Folha, 2017), consequência da diminuição do interesse do público e legislações específicas para food

trucks que dificultam sua circulação e comercialização. Apesar das controvérsias em alguns países, outros seguem com um crescimento contínuo, como é o caso dos Estados Unidos. Segundo o site Economist (Economist, 2017), apenas na região americana, há mais de 4.000 food trucks cadastrados e esse número tende a crescer esporadicamente, resultado na qual vem causando transtornos e ameaçando os negócios dos restaurantes.

Com o enfraquecimento do mercado de food trucks com os impactos causados pelas legislações atuais, este trabalho visa propor uma possível alternativa, de expor e facilitar sua localização e oferecer serviços de ordem de pedidos, uma alternativa para rebater dificuldade de locomoção dos food trucks.

1.3 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um aplicativo de mapeamento consolidado de food trucks por cidades, estados e países e oferecer serviços de ordem de pedidos.

1.4 Objetivos específicos

Os objetivos específicos deste trabalho são os seguintes:

- Pesquisar alternativas de mapeamento de food trucks.
- Criar uma ferramenta de integração dos dados salvos pelo usuário
- Pesquisar alternativas de identificação e mapeamento dos cardápios
- Pesquisar alternativas de comércio pelo aplicativo
- Desenvolver o aplicativo responsável por mapear os pontos de food trucks
- Desenvolver meios de pedidos

1.5 Método

As etapas de desenvolvimento do trabalho são descritas a seguir:

- Levantamento de tecnologias – As tecnologias utilizadas no desenvolvimento deste projeto foram optadas com base em um levantamento de tecnologias fortes no mercado atual, considerando as necessidades do projeto.
- Levantamento de requisitos – O levantamento de requisitos foi realizado considerando características e funcionalidades da visão do autor e estudos de trabalhos relacionados.
- Modelar UML – Os casos de uso e o diagrama de casos de uso foram desenvolvidos a partir dos requisitos levantados para guiar o desenvolvimento de outros diagramas da UML e também do aplicativo. Os diagramas desenvolvidos foram: Diagrama de atividades, Diagrama de classes e diagrama de caso de uso.
- Desenvolver o projeto - Com os diagramas desenvolvidos, foi feito o desenvolvimento do aplicativo. O aplicativo consome diversos WebServices da API do Google Maps, onde o mesmo fornece informações de localização, pontos de food trucks, fotos dos pontos de food trucks e auxílio no cálculo e mapeamento de rotas até o local. Além do consumo de WebServices, é realizado uma integração com o Firebase para auxiliar no armazenamento dos dados do usuário e suas operações. O aplicativo tem o objetivo de apresentar os pontos de food trucks no mapa e permitir detalhes de sua localização, listagem de cardápio, cálculo de rotas e ordem de pedidos.

1.6 Estrutura do trabalho

O trabalho é dividido em quatro capítulos, compostos pela Introdução, Fundamentação teórica, Desenvolvimento do projeto e Conclusão. A Introdução apresenta a motivação, justificativa, objetivos gerais, objetivos específicos e metodologia deste trabalho. A Fundamentação teórica é composta pelo levantamento das tecnologias utilizadas para o desenvolvimento do trabalho e um comparativo de aplicativos relacionados. No Desenvolvimento do projeto, descreve como o desenvolvimento foi realizado, com apresentação de protótipos, requisitos do sistema, especificações e implementação. Na Conclusão é exposto as conclusões obtidas no desenvolvimento deste trabalho, pendências e futuras implementações.

2 Fundamentação Teórica

Neste capítulo, são apresentadas um estudo das tecnologias presentes no mercado e que serão utilizados para o desenvolvimento deste projeto (seção 2.1), e trabalhos relacionados à categoria do aplicativo desenvolvido neste projeto (seção 2.2).

2.1 Tecnologias

O estudo das necessidades que o aplicativo teria que sanar, alinhados com o conhecimento prévio das partes envolvidas, levaram a uma lista de tecnologias que serão utilizadas no desenvolvimento do aplicativo aqui proposto, elas são:

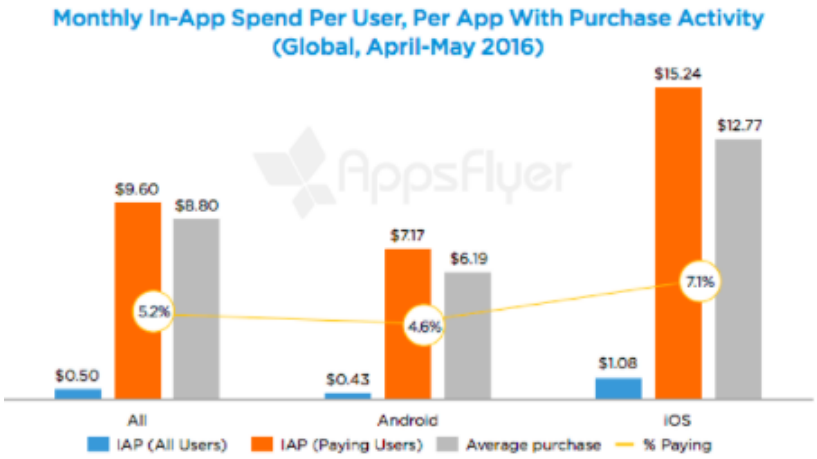
2.1.1 Escolha da tecnologia

Para o desenvolvimento do aplicativo foi considerado as duas linguagens nativas do mercado atual: Android e iOS. Para o processo de criação desta monografia, foi escolhido o iOS, conforme pontos abaixo:

Usuários pagantes:

Apesar do Android atrair maiores consumidores pela sua variedade de dispositivos e opções de preços, Apple ainda se mantém uma forte concorrente pelos seu mercado amplo composto por aplicativos pagos (APPSFLYER, 2016). Desenvolver em iOS tem um grande potencial de retorno financeiro para aplicativos que oferecem serviços pagos, sendo que usuários de iPhone consomem 2,5 vezes à mais que usuários Android, conforme figura 1:

Figura 1- Comparativo de lucro em compras de aplicativos no mundo (Abril – Maio 2016)



Fonte: AppsFlyer - The State of in app spending (2016)

Tempo de Desenvolvimento:

Uma pesquisa realizada em 2017 (INFINUM, 2017), foi organizada para coletar dados e distinguir qual das tecnologias predominantes do mercado mobile tem maior custo de projeto. Utilizando os parâmetros abaixo, foram escolhidos e abstraídos 6 projetos para compor a pesquisa:

- Projeto desenvolvido em iOS e Android
- Projetos sem código legado

Parâmetros de mensuração:

- Linhas de código
- Horas trabalhadas

Conforme exibido na figura 2, a pesquisa demonstrou uma média de 40% mais código em aplicações Android do que iOS:

Figura 2 - Comparativo linhas de código entre projetos Android x Swift

Lines of Code

	iOS	Android	Difference in %
Project A	6,829	15,323	124%
Project B	48,671	50,756	4%
Project C	15,807	28,449	80%
Project D	5,148	14,893	189%
Project E	21,698	25,501	18%
Project F	6,956	10,347	49%
Total	105,109	145,269	38%

Fonte: Infinum - Android vs iOS Development trends for 2017 (2017)

Além do diferencial em linhas de código, na figura 3 é demonstrado o comparativo de horas trabalhadas entre as plataformas Android e Swift:

Figura 3 - Comparativo horas trabalhadas entre projetos Android x Swift

	iOS	Android	Difference in %
Project A	241	440	83%
Project B	1,586	1,613	2%
Project C	822	1,157	41%
Project D	295	755	156%
Project E	602	647	7%
Project F	244	257	5%
Total	3,790	4,869	28%

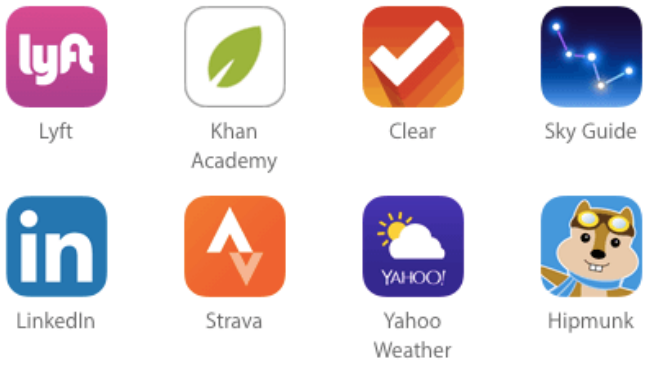
Fonte: Infinum - Android vs iOS Development trends for 2017 (2017)

Através do gráfico, é possível notar uma média de 30% mais tempo para desenvolver aplicações Android. Apesar dos dados coletados na pesquisa acima, é evidente que podem haver parâmetros que dificultem a mensuração. Porém, 2 itens foram apontados como fatores que reforçam

maior lentidão no desenvolvimento em Android: maior número de aparelhos para testes e emulador mais lento.

Um exemplo que comprova os dados resultantes da pesquisa foi a estratégia da empresa Lyft app (FASTCOMPANY, 2015), na qual a mesma reescreveu sua aplicação iOS utilizando Swift. Outras empresas também adotaram a mesma estratégia de migração. Na figura 4 é demonstrado um levantamento realizado pelo site SpaceOTechnologies em 2016, de aplicativos convertidos de Objective-C para Swift:

Figura 4 - Exemplo de aplicativos migrados de Objective-C para Swift



Fonte: 6 Reasons Why Startups & Enterprises Choose Swift For Their Custom iOS App Development (2016)

Lyft, uma empresa de transportes, demonstrou que enquanto sua antiga base de dados era composta por aproximadamente 75,000 linhas de código, a versão Swift adotando as mesmas funcionalidades resultou em menos de 1/3 da versão antiga. Além disso, enquanto a versão antiga foi necessária mais de um mês para diversos engenheiros implementarem, a nova versão com Swift foi finalizada em apenas 1 semana com apenas 1 engenheiro.

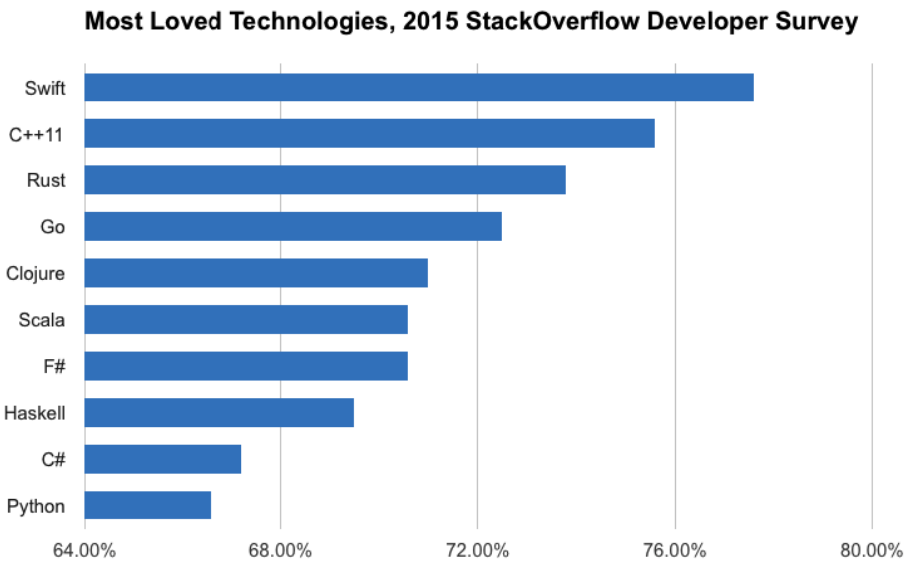
2.1.2 Swift

Para o desenvolvimento de um aplicativo Apple, é possível optar entre duas linguagens de programação, sendo eles: Objective-C e Swift. O Objective-C (Apple, 2014) é uma linguagem baseada na linguagem C e suporta uma programação orientada à objeto e capacidade de uma runtime

de linguagem dinâmica (DLR, facilita o desenvolvimento de linguagens dinâmicas para execução no .NET Framework e para adicionar recursos dinâmicos a linguagens de tipos). O mesmo herda a sintaxe, tipos primitivos e controle de fluxo da linguagem C e sintaxes com capacidade de definição de classes e métodos. Em contrapartida, o Swift é uma linguagem moderna e criada seguindo 3 regras: Seguro, Rápido e Expressivo. Rápido por apresentar uma sintaxe limpa, expressiva e simplificada, facilitando sua leitura e escrita. Por ser concisa, resulta em menos linhas de código comparado com o Objective-C. Criado para ser o substituto das linguagens baseadas em C (C, C++ e Objective-C), se tornou open source em 2015 e desde então houve um crescimento significativo em cada versão lançada pela empresa, apesar das controvérsias de como seria sua aceitação no mercado desde seu lançamento.

Conforme figura 5, a aceitação da linguagem Swift tem se destacado em grande escala através de uma pesquisa realizada pelo StackOverflow em 2015:

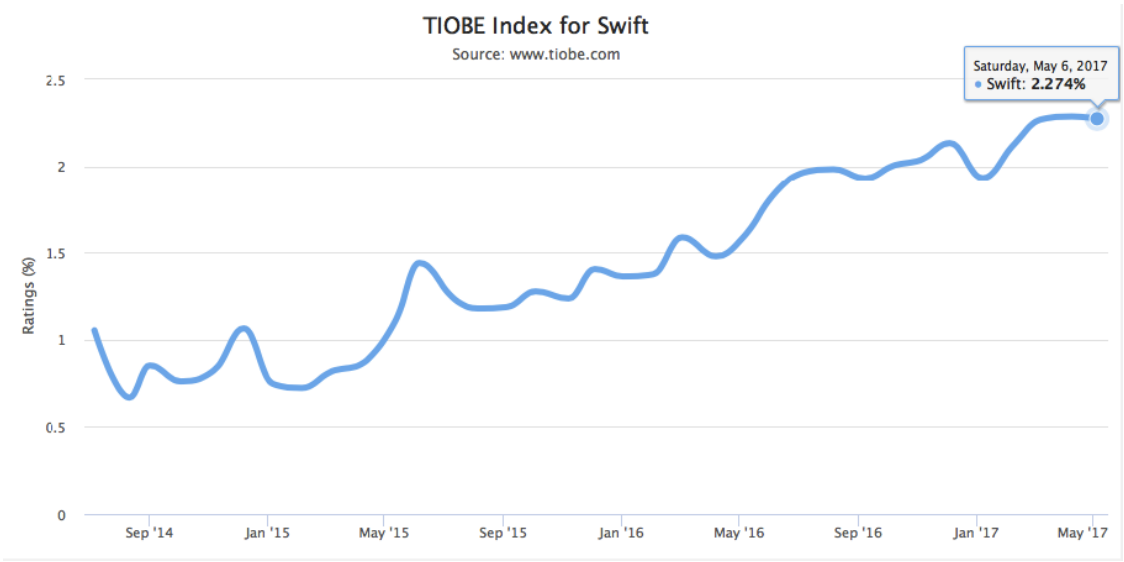
Figura 5 - Tecnologias com melhores índices de aceitação no mercado



Fonte: Stackoverflow Developer Survey (2015)

Após se tornar open source, o Swift se tornou a linguagem com maior crescimento na história, de acordo com dados do TIOBE Index, conforme figura 6:

Figura 6 - Crescimento da linguagem Swift de 2014 até 2017



Fonte: Tiobe The Swift Programming Language (2017)

Atualmente a linguagem Swift permanece no ranking das linguagens mais utilizadas, conforme dados do TIOBE, na figura 7:

Figura 7 - Lista de tecnologias mais utilizadas no mercado

Jan 2018	Jan 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.215%	-3.06%
2	2		C	11.037%	+1.69%
3	3		C++	5.603%	-0.70%
4	5	▲	Python	4.678%	+1.21%
5	4	▼	C#	3.754%	-0.29%
6	7	▲	JavaScript	3.465%	+0.62%
7	6	▼	Visual Basic .NET	3.261%	+0.30%
8	16	⇈	R	2.549%	+0.76%
9	10	▲	PHP	2.532%	-0.03%
10	8	▼	Perl	2.419%	-0.33%
11	12	▲	Ruby	2.406%	-0.14%
12	14	▲	Swift	2.377%	+0.45%
13	11	▼	Delphi/Object Pascal	2.377%	-0.18%
14	15	▲	Visual Basic	2.314%	+0.40%
15	9	⇊	Assembly language	2.056%	-0.65%
16	18	▲	Objective-C	1.860%	+0.24%

Fonte: Tiobe The Swift Programming Language (2017)

2.1.2.1 Vantagens da Linguagem Swift

De acordo com uma nota oficial liberada pela Apple, "Swift combina performance e eficiência de linguagens compiladas com a simplicidade e interações populares das linguagens de script" (APPLE, 2014), o que incide em uma linguagem superior ao Objective-C em vários aspectos.

- Processo de desenvolvimento rápido: Limpo, simplificado e conciso, seu desenvolvimento requer menos linhas de código. Apresenta ARC (Automatic Reference Counting), que realiza todo o trabalho de manter registros e gerenciar o uso da memória (Reduz o tempo de build de aplicativos em Swift além de facilitar a vida do desenvolvedor, evitando gerenciamento manual)
- Fácil de escalonar o produto: Além do rápido tempo de desenvolvimento, a linguagem se beneficia pelo fato de ter suporte contínuo pela Apple com o tempo, sem uma garantia para o Objective-C
- Segurança e performance aprimoradas: Swift é considerado superior no quesito performance comparado ao Objective-C e outras linguagens, segundo testes de benchmark (BENCHMARKGAME, 2017) realizados por diversos desenvolvedores
- Interoperabilidade com a linguagem Objective-C: Swift é perfeitamente compatível com Objective-C, podendo ser desenvolvido com ambas linguagens no mesmo projeto
- Potencial Full stack e suporte Cross-device:Server-side, Swift integra com as tecnologias backends mais populares.

Além dos benefícios da linguagem, com o surgimento da nova linguagem Swift e o crescimento contínuo de sua popularidade, gerou uma maior demanda de desenvolvedores iOS. (THE VERGE, 2014)

2.1.2.2 Desvantagens da Linguagem Swift

Apesar de possuir várias vantagens, a linguagem ainda é relativamente nova e está continuamente crescendo, sendo que a cada versão lançada gera inúmeras incompatibilidades por apresentar um grande número de mudanças na linguagem. Enquanto Xcode (IDE para

desenvolvimento de aplicativos iOS) apresenta uma ferramenta que auxilia migrar a versão atual para a versão mais recente, o mesmo não corrige todos os erros gerados. Consequentemente força os desenvolvedores a reescrever o projeto para a última versão da linguagem.

2.1.2.3 Versões

As versões mais populares e utilizadas atualmente, são: 2.0, 3.0 e 4.0. Lançado recentemente, a versão 4 apresenta poucas mudanças drásticas comparadas a migração do 2 ao 3. Para este trabalho será adotado a nova versão, para evitar uma possível migração futura.

Por ter sido lançado em junho de 2017, há muitas bibliotecas terceiras que ainda não foram migradas, o que gera várias incompatibilidades durante a compilação do projeto. Para contornar esse possível problema, utilizando o gerenciador de dependências CocoaPods e adicionando uma condição, é possível forçar a compilação de bibliotecas em versões específicas, conforme demonstrado na figura 8:

Figura 8 - Exemplo de declaração no PodFile

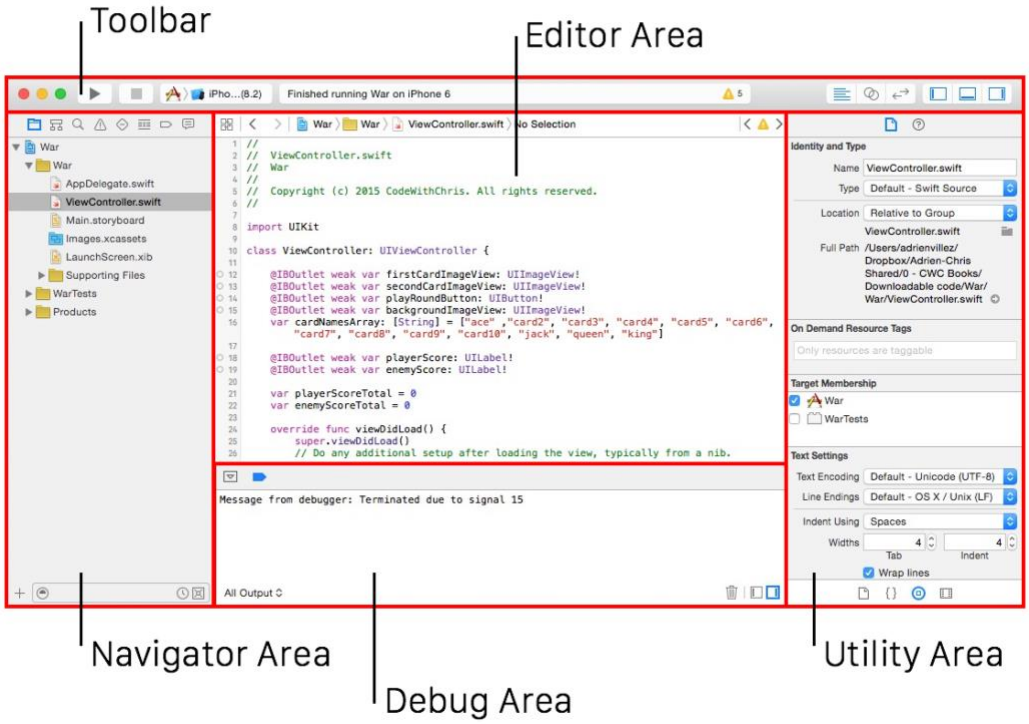
```
old_swift_3_pods = [
    'PodName1',
    'PodName2',
]

post_install do |installer|
    installer.pods_project.targets.each do |target|
        if old_swift_3_pods.include? target.name
            target.build_configurations.each do |config|
                config.build_settings['SWIFT_VERSION'] = '3.2'
            end
        end
      end
end
```

2.1.2.4 Estrutura de um projeto

Para o desenvolvimento de aplicativos iOS, é fornecido uma IDE pela Apple, o Xcode, conforme figura 9 abaixo:

Figura 9 - Representação da IDE XCode



2.1.2.5 Geração de Arquivo IPA

Um arquivo .ipa (GENEXUS, 2016) é utilizado para empacotar aplicativos iOS e redistribuído à usuários finais, tanto internamente para realização de testes durante fases de desenvolvimento ou para disponibilizar na plataforma de store quando o mesmo já pode ser disponibilizado em produção. Cada arquivo inclui um binário para a arquitetura ARM e só pode ser instalado em um dispositivo iOS. A maioria dos arquivos .ipa não podem ser instalados no iPhone simulator porque eles não possuem um binário para arquitetura x86.

2.1.2.6 Regras para publicação na Store

Apple possui diversas regras antes de permitir que um aplicativo seja publicado em sua store. Para validar o aplicativo, é necessário atentar a 2 pontos como: Teste e Regras.

- Testes em todos os aparelhos na qual se pretende disponibilizar o aplicativo
- Seguir as diretrizes de desenvolvimento de softwares, através das documentações disponibilizadas pelo site da Apple:
    - iOS Human Interface Guidelines
    - App Store Review Guidelines

Algumas das regras cruciais mencionadas na documentação reforçados constantemente através dos itens abaixo:

- a aplicação pode não quebrar
- não utilizar API privados
- não replicar funcionalidades de aplicações nativas
- não utilizar câmera ou microfone sem consentimento do usuário
- utilizar apenas imagens com direitos autorais
- links quebrados (todos os links do aplicativo devem estar funcionais)
- Descrições claras do que o app oferece sem passar a impressão de que realiza outra funcionalidade

Além das regras, é necessário atender a alguns pré-requisitos:

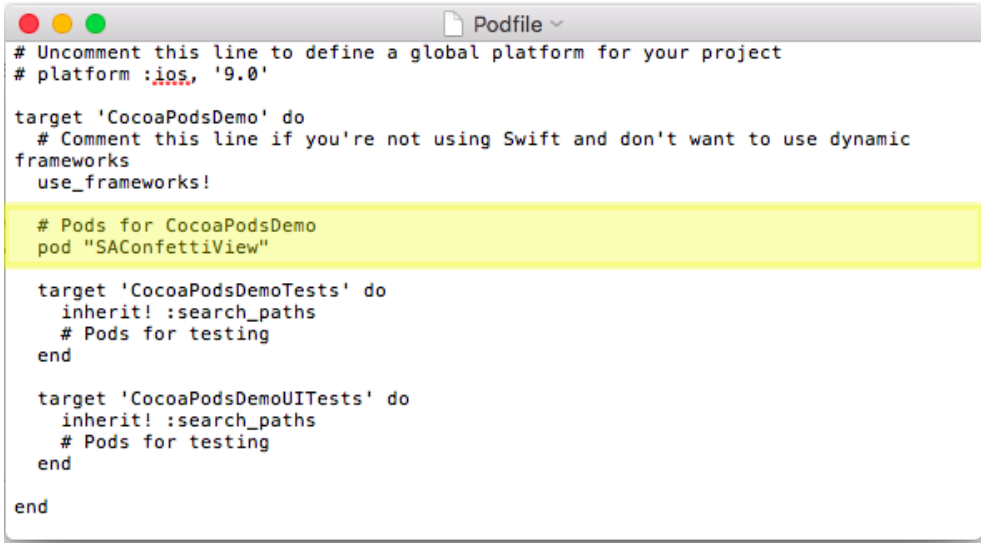
- Possuir um Apple ID: conta Apple
- Certificado de distribuição válido: certificado para identificação do time e/ou organização e permite submeter apps à apple store.
- Perfil de provisão válido: Coleção de entidades digitais que vincula exclusivamente desenvolvedores e dispositivos a uma equipe de desenvolvimento de iPhone autorizada e permite que um dispositivo seja usado para testes.

2.1.3 CocoaPods

Aplicação de gerenciamento de dependências, é aplicável tanto para Objective-C quanto Swift. Possui suporte de mais de 38 mil bibliotecas , permite integração automática com projetos Xcode, o que facilita projetos escalarem de forma mais elegante, sendo manuseados através de linhas de comandos (RAYWENDERLICH, 2017). Para sua utilização, é necessário a criação de um arquivo

Podfile no projeto, para que a mesma seja utilizado para descrever as dependências do projeto alvo. O objetivo do cocoapods é melhorar a descoberta e o envolvimento em bibliotecas de fonte aberta de terceiros criando um ecossistema mais centralizado.

Figura 10 - Exemplo de um arquivo PodFile com dependências



Na figura 10 é demonstrado um exemplo de arquivo podfile. Após a criação do Podfile, é necessário instalar todas as dependências através da execução do comando "Pod install". Finalizado, será gerado um projeto com a extensão .workspace, na qual compõe o projeto juntamente com as dependências instaladas.

2.1.4 Firebase

Plataforma que segue o modelo Backend-as-a-Service (BaaS). O modelo fornece uma maneira de vincular aplicativos à backend cloud storage(plataformas clouds) e APIs expostos por aplicativos back-end além de fornecer recursos como gerenciamento de usuários, notificações instantâneas e integração com serviços de rede social. Por se tratar de uma plataforma que oferece o uso de um banco de dados em tempo real na nuvem, não há necessidade de gerenciar servidores. Firebase é o servidor, API e datastore da sua aplicação e ainda permite a possibilidade de alterar as configurações e linhas de código conforme necessidade. (HOW TO FIREBASE, 2017)

Benefícios da API (HOW TO FIREBASE, 2017):

- Utiliza websockets para sincronização de dados (Normalmente é utilizado protocolo HTTP)
- Permite armazenamento de arquivos (binário)
- Firebase oferece um sistema de autenticação email/password além de suporte para OAuth2, Facebook, Google, Twitter e GitHub.
- Autenticação do firebase é integrado diretamente a base de dados do firebase, permitindo controlar os dados de acessos
- Base de dados em tempo real
- Construído em segurança em nível de dados
- Armazenamento de arquivos
- Hospedagem estática de arquivos
- Sem necessidade de se preocupar com a infraestrutura
- Capacidades de consultas ilimitadas

2.1.5 SwiftyJSON

Biblioteca para tratamento de JSON (modelo para armazenamento e transmissão de informações no formato texto.) (SWIFTYJSON, 2018).

Abaixo, uma representação de implementação em swift para tratar um JSON:

```
if let statusesArray = try? JSONSerialization.jsonObject(with: data, options:
.allowFragments) as? [[String: Any]],
    let user = statusesArray[0]["user"] as? [String: Any],
    let username = user["name"] as? String {
    // Finally we got the username
}
```

Fonte: GitHub swiftyJSON (2018)

Mesmo realizando uma reestruturação do código, ainda permanece um pouco trabalhoso:

```
if let JSONObject = try JSONSerialization.jsonObject(with: data, options:
.allowFragments) as? [[String: Any]],
    let username = (JSONObject[0]["user"] as? [String: Any])?["name"] as? String {
        // There's our username
}
```

Fonte: GitHub swiftyJSON (2018)

Aplicando SwiftyJSON, o tratamento se resume em algumas linhas:

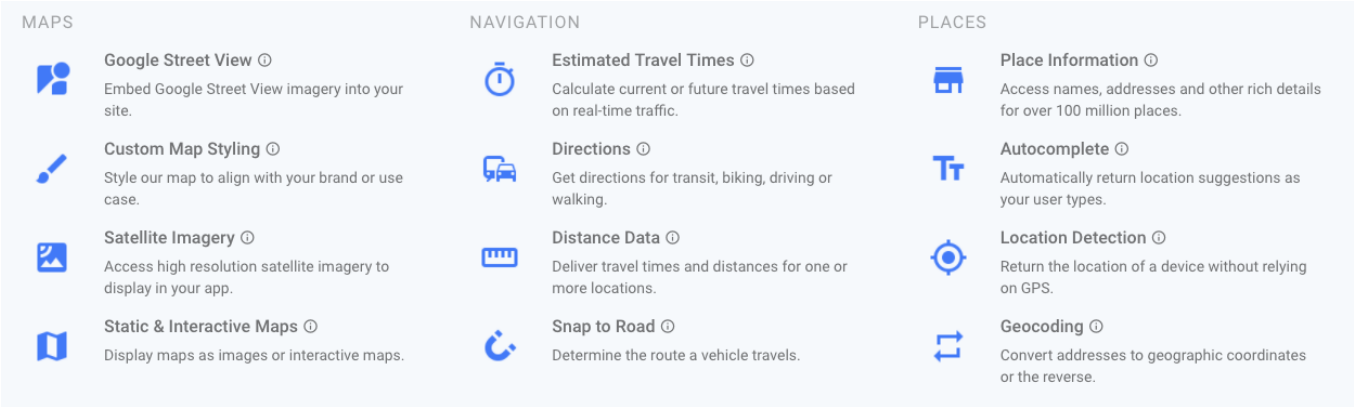
```
let json = JSON(data: dataFromNetworking)
if let userName = json[0]["user"]["name"].string {
  //Now you got your value
}
```

Fonte: GitHub swiftyJSON (2018)

2.1.6 Google Maps API

API poderosa que permite customizações de mapas, visualização de locais, panorâmicas de ruas, lista de web services, que permitem cálculo de distância entre locais, obtenção de localização, visualização de locais específicos e vários outros serviços relacionados à localização (GOOGLE DEVELOPER, 2017). Na figura 11, uma relação de serviços oferecidos pela API:

Figura 11 - Serviços oferecidos pela API Google Maps



Fonte: Google Maps API - Build the next generation of local experiences (2018)

Segundo o site codementor (CODEMENTOR, 2016), alguns dos pontos chaves que incentivam o uso da API:

- Rica em informações
- Múltiplos tipos de transporte
- Suporte para grande escala de marcadores no mapa (10.000+)
- Documentações atualizadas constantemente
- Compartilhamento por redes sociais
- Customização dos mapas

2.1.7 Web Service

Um Web service (OPENSOFT, 2017) é um conjunto de métodos acedidos e invocados por outros programas utilizando tecnologias Web. É utilizado para transferir dados através de protocolos de comunicação para diferentes plataformas, independentemente das linguagens de programação utilizadas nessas plataformas. Os Web services funcionam com qualquer sistema operativo, plataforma de hardware ou linguagem de programação de suporte Web. Estes transmitem apenas informação, ou seja, não são aplicações Web que suportam páginas que podem ser acedidas por utilizadores através de navegadores Web. Para tal, existem protocolos de comunicação como o SOAP (Simple Object Access Protocol) e o REST (Representational State Transfer).

O protocolo SOAP utiliza XML para enviar mensagens e, geralmente, serve-se do protocolo HTTP para transportar os dados. Associado ao protocolo SOAP está o documento WSDL (Web Service Definition Language) que descreve a localização do Web service e as operações que dispõe. Além disso, fornece a informação necessária para que a comunicação entre sistemas seja possível (OPENSOFT, 2017).

O REST é um protocolo de comunicação mais recente que surgiu com o objetivo de simplificar o acesso aos Web services. Este baseia-se no protocolo HTTP e permite utilizar vários formatos para representação de dados, como JSON (um dos mais utilizados), XML, RSS, entre outros (OPENSOFT, 2017).

2.1.7.1 Google Geocoding API

Lista um conjunto de informações referentes à localização do usuário.

Exemplo de chamada:
https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY

Parâmetros obrigatórios (GOOGLE DEVELOPER, 2017):

- key: A chave de API do aplicativo. Essa chave identifica o aplicativo para fins de gerenciamento de cotas e de forma que locais adicionados a partir dele sejam imediatamente disponibilizados para o aplicativo.

- location: A latitude/longitude em torno da qual você deseja recuperar informações de local. Deve ser especificado como latitude,longitude.

Na figura 12, um exemplo de retorno JSON do serviço geocoding:

Figura 12 - Exemplo de retorno JSON Google Geocoding


```
{
   "html_attributions" : [],
   "result" : {
      "address_components" : [
         {
            "long_name" : "5",
            "short_name" : "5",
            "types" : [ "floor" ]
         },
         {
            "long_name" : "48",
            "short_name" : "48",
            "types" : [ "street_number" ]
         },
         {
            "long_name" : "Pirrama Road",
            "short_name" : "Pirrama Rd",
            "types" : [ "route" ]
         },
         {
            "long_name" : "Pyrmont",
            "short_name" : "Pyrmont",
            "types" : [ "locality", "political" ]
         },
         {
            "long_name" : "Council of the City of Sydney",
            "short_name" : "Sydney",
            "types" : [ "administrative_area_level_2", "political" ]
         },
```


Fonte: Google Developers Google Geocoding

2.1.7.2 Google Places API

Após a obtenção de dados referente a localização do usuário, é possível consultar os food trucks da região através da chamada desta requisição.

Exemplo de chamada:

https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=500&type=restaurant&keyword=cruise&key=YOUR_API_KEY

Parâmetros obrigatórios (GOOGLE DEVELOPER, 2017):

- key(obrigatório): A chave da API do seu aplicativo . Esta chave identifica seu aplicativo para fins de gerenciamento de cotas e para que os lugares adicionados do seu aplicativo sejam imediatamente disponibilizados para seu aplicativo.
- query: Especifica o que deseja pesquisar. Para o projeto será utilizado a palavra “food+truck
- location: Latitude e longitude da região onde se deseja obter informações da consulta.

Na figura 13 é possível visualizar um exemplo de retorno JSON do google places:

Figura 13 - Exemplo de retorno JSON Google Places


```
{
   "html_attributions" : [],
   "results" : [
      {
         "geometry" : {
            "location" : {
               "lat" : -33.870775,
               "lng" : 151.199025
            }
         },
         "icon" : "http://maps.gstatic.com/mapfiles/place_api/icons/travel_agent-71.png",
         "id" : "21a0b251c9b8392186142c798263e289fe45b4aa",
         "name" : "Rhythmboat Cruises",
         "opening_hours" : {
            "open_now" : true
         },
         "photos" : [
            {
               "height" : 270,
               "html_attributions" : [],
               "photo_reference" : "CnRnAAAAF-LjFR1ZV93eawe1cU_3QNMCNmaGkowY7CnOf-kcNmPhNnPEG9W979j
               "width" : 519
            }
         ],
         "place_id" : "ChIJyWEHuEmuEmsRm9hTkapTCrk",
         "scope" : "GOOGLE",
         "alt_ids" : [
            {
```


Fonte: Google Developers Google Places

2.1.7.3 Google Maps Direction API

O WebService directions fornece várias opções de busca de direções por tipos de transporte como: bicicleta, veículos automotivos e a pé. A API retorna as rotas mais eficientes ao calcular as direções. O tempo de viagem é o fator principal otimizado, mas a API também pode levar em consideração outros fatores, como distância, número de voltas e muitos mais ao decidir qual rota é a mais eficiente. O cálculo de direções é uma tarefa intensiva em tempo e recursos.

Exemplo de Chamada:
https://maps.googleapis.com/maps/api/directions/json?origin=Toronto&destination=Montreal&key=YOUR_API_KEY

Parâmetros obrigatórios (GOOGLE DEVELOPER, 2017):

- origin - O endereço, o valor da latitude / longitude textual ou o place_ID do lugar a partir do qual você deseja calcular as instruções
- destination - O endereço, o valor da latitude / longitude textual ou o place_ID para o qual você deseja calcular as instruções. As opções para o parâmetro de destino são as mesmas que para o parâmetro de origem.

Parâmetros opcionais:

- mode (padrão de condução): Especifica o modo de transporte a ser usado ao calcular as direções. Valores válidos e outros detalhes de solicitação são especificados em Modos de viagem.
- idioma: idioma para retornar resultados.

Na figura 14, um exemplo de retorno JSON do Google Direction:

Figura 14 - Exemplo de retorno JSON Google Direction


```
{
  "status": "OK",
  "geocoded_waypoints" : [
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ7cv00DwsDogRAMDACa2m4K8",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ69Pk6jdlyIcRDqM1KDY3Fpg",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJgdL4flSKrYcRnTpP0XQSojM",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJE9on3F3HwoAR9AhGJW_fL-I",
        "types" : [ "locality", "political" ]
     }
  ],
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode": "DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },
        "end_location": {
          "lat": 41.8525800,
          "lng": -87.6514100
        },
        "polyline": {
          "points": "a~l~Fjk~uOwHJy@P"
        },
        "duration": {
          "value": 19,
          "text": "1 min"
        },
```


Fonte: Google Developers Google Directions

2.1.7.4 Modelo de Chamada

Para atender a necessidade deste trabalho, será utilizada chamadas REST provenientes do Google Maps API. Será fornecido dados da localização (latitude e longitude) do usuário através do GPS e realizado 3 chamadas através de web services do Google Maps API:

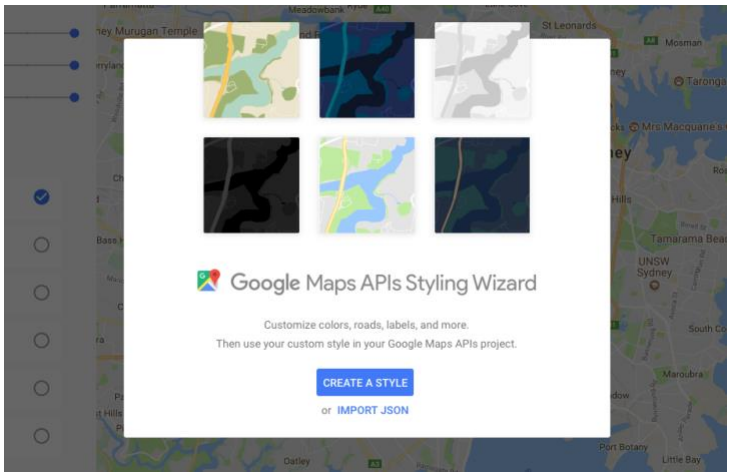
- Google Geocoding API: Através da latitude e longitude, será realizado uma busca através do webservice Geocode para coletar informações da região do usuário
- Google Place API: Após coleta da localização do usuário, será realizado uma chamada pelo webservice Place Details para coletar informações dos food trucks da região (horários de atendimento, avaliações, entre outros)
- Google Maps Directions API: Será utilizado para calcular a rota entre o usuário até o local do food truck

- Google Place Photos: Através do place ID adquirido pelo web service Place Details, o mesmo é utilizado no serviço Place Photos para coletar as imagens do estabelecimento.

2.1.8 Mapa

Para a criação do mapa, será utilizado a ferramenta do Google Maps (GOOGLE DEVELOPER, 2017) que oferece várias customizações que abrangem personalização de ruas, referências, rótulos, alterações de cores e geografia do mapa. É possível visualizar a ferramenta através da imagem 15:

Figura 15 - Ferramenta de personalização de Mapas da Google



Fonte: Google Maps APIs Styling Wizard (2017)

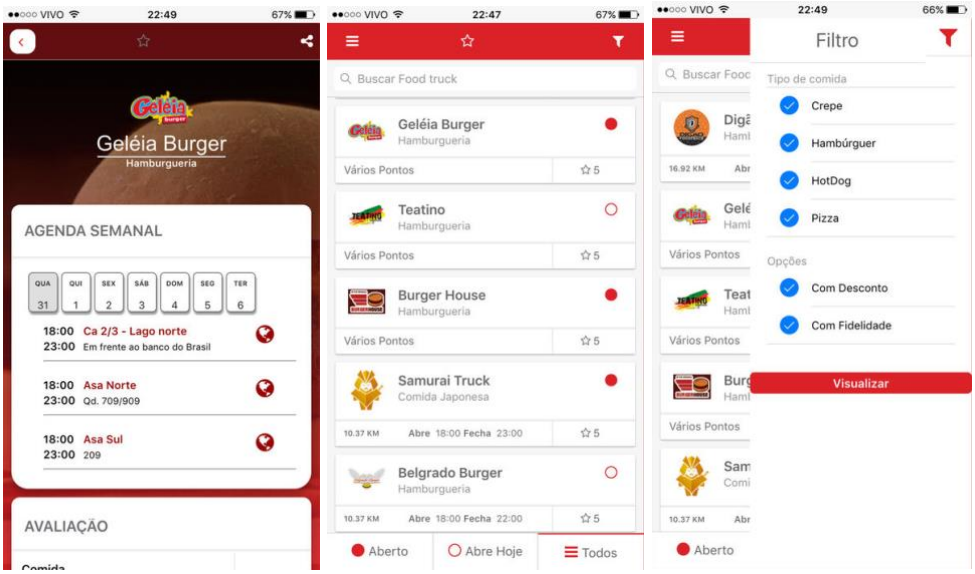
- No trabalho, será utilizado o mapa para disponibilizar:
- Point of Interest (Locais dos food trucks)
- Permitir selecionar o food truck e direcionar em uma outra tela para exibir detalhes sobre o food truck
- Exibir rota até o food truck

2.2 Trabalhos Relacionados

Foi realizado um levantamento de alguns aplicativos que possuem objetivos e requisitos semelhantes ao apresentado neste trabalho. Segue a seguir uma breve descrição e suas principais funcionalidades:

- Central Food Truck: Desenvolvido para android e iOS, possui cobertura apenas na região de Brasília, disponibiliza agenda de eventos, notícias do mundo food truck, integração com redes sociais, catálogo de promoções e descrição da localização. Interface do aplicativo na figura 16.

Figura 16 – Interface do aplicativo Central Food Truck

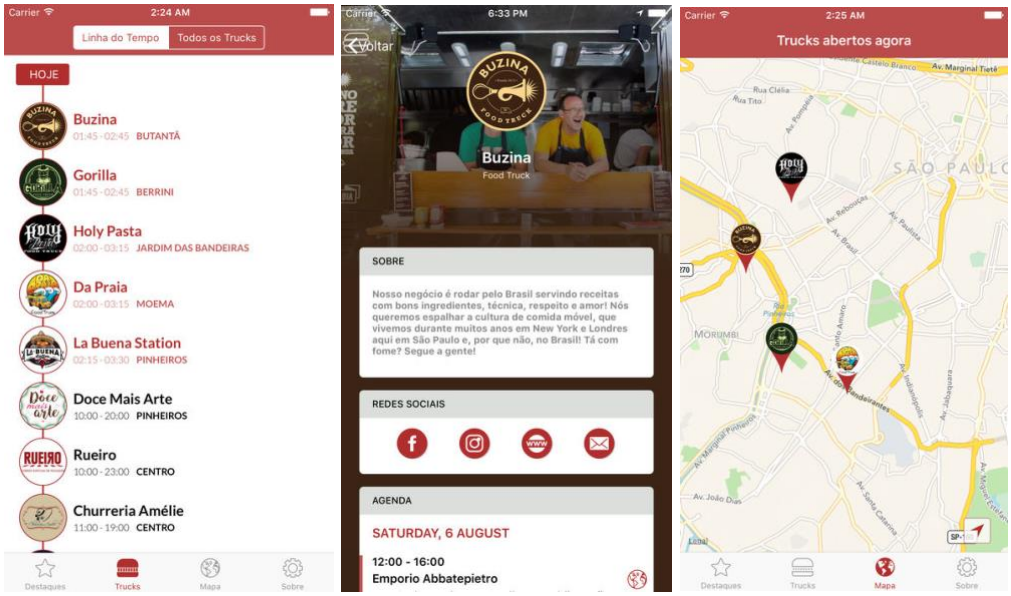


- Food Truck nas Ruas: Desenvolvido para as plataformas android e swift, possui cobertura em 27 cidades, disponibiliza localização de food bikes e carts, cupons de descontos, cartão fidelidade, serviço de delivery e visualização de food trucks com pontos fixos. Interface do aplicativo na figura 17.

Figura 17 – Interface do aplicativo Food Truck nas Ruas

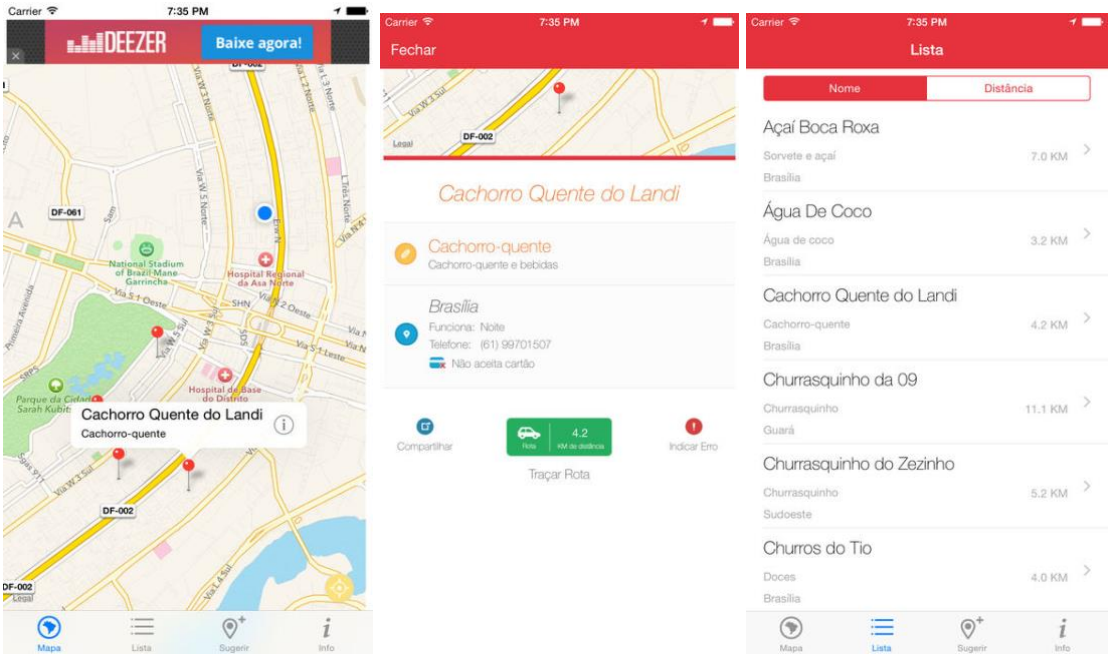
- Guia Food Truck: Desenvolvido em Android e iOS, possui cobertura apenas em SP, disponibiliza agenda de eventos, cardápios dos pontos de food truck e pesquisa por categorias. Interface do aplicativo na figura 18.

Figura 18– Interface do aplicativo Guia Food Truck



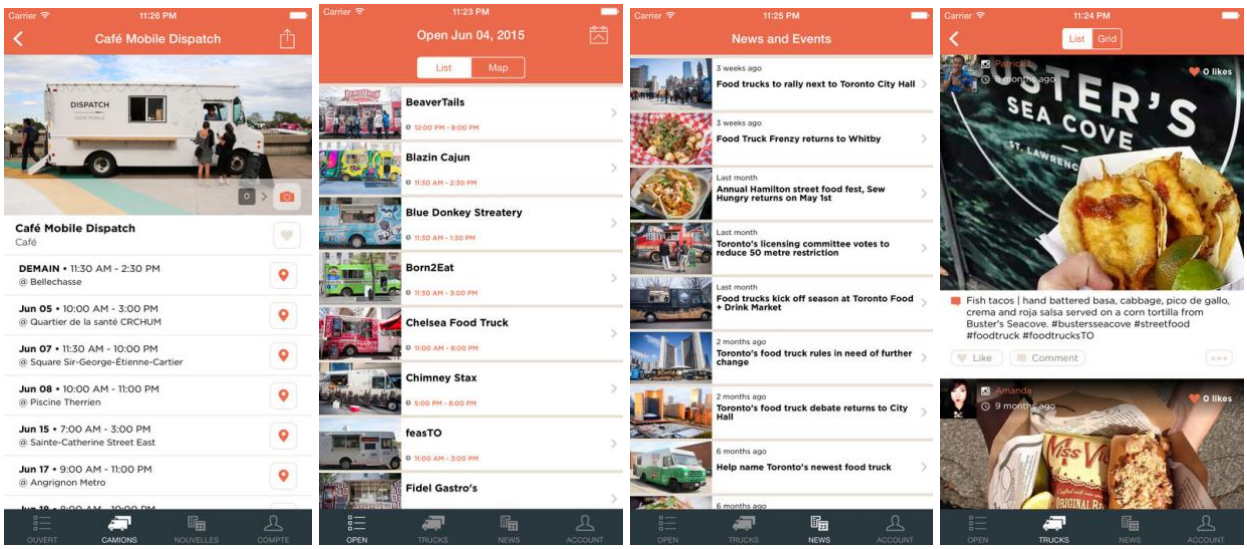
- Comida de Rua: Desenvolvido para Android e iOS, possui cobertura apenas em SP, permite localizar food trucks no mapa ou numa lista. Interface do aplicativo na figura 19.

Figura 19 – Interface do aplicativo Comida de Rua



- Food Trucks: Desenvolvido para Android e iOS, possui cobertura apenas no Canadá, permite localizar food trucks no mapa ou numa lista, possui catálogo de eventos, permite favoritar um food truck e criação de perfil. Interface do aplicativo na figura 20.

Figura 20 – Interface do aplicativo Food Trucks



Analisando as funcionalidades desses aplicativos, percebe-se um foco em cadastros manuais (pois grande maioria dos aplicativos listados possuem cobertura para apenas uma cidade) dos food trucks. Conclui-se que os aplicativos listados não entram em conflito com o objetivo principal do projeto, porém, algumas das funcionalidades que serão desenvolvidas no projeto já estão presentes em alguns dos aplicativos.

3 Desenvolvimento do projeto

Neste capítulo, são apresentadas as etapas do desenvolvimento, sendo elas:

Descrição da aplicação (seção 3.1), levantamento de requisitos (seção 3.2), casos de uso (seção 3.3), diagramas de classe (seção 3.4) e atividades (seção 3.5) e implementação (seção 3.6).

3.1 Descrição da aplicação

A aplicação consiste em um aplicativo desenvolvido na linguagem swift, executando no aparelho do usuário e se comunicando com a plataforma dedicada proveniente da Google, o Firebase.

O aplicativo consiste em treze telas, responsáveis em exibir os food trucks no mapa, prover informações detalhadas do food truck, listagem de cardápios, pesquisa de food trucks por regiões, realização de pedidos, cadastro de cartão de crédito, visualização de galeria de imagens, login, histórico de pedidos, cálculo de rotas e criação de perfil, sendo que o processo de autenticação, criação de perfil, informações da ordem de pedido e cartões de créditos cadastrados serão realizados através da API do Firebase.

O projeto utiliza o Firebase para a realização de três funções principais: A base de dados que armazena os dados salvos pelos usuários, juntamente com as informações do cartão de crédito e ordem de pedidos, serviço de armazenamento de mídia para salvar as imagens do perfil do usuário e o serviço de autenticação de usuário.

3.2 Levantamento de Requisitos

A seguir são apresentados os requisitos funcionais e não funcionais identificados para a aplicação móvel que foi desenvolvida neste trabalho. Estes são os requisitos funcionais necessários ao sistema. Para o mesmo, é considerado um aplicativo futuro para gerenciamento de food trucks para Owners (donos de food trucks).

3.2.1 Requisitos Funcionais

3.2.1.1 Cadastro de Food Truck (Aplicativo food truck para Owners)

RF001 - Visualizar Food Truck (Para ambos aplicativos: Cliente e Owner)

Descrição: Este requisito do sistema permite que o usuário visualize os dados de um determinado food truck (todos os seus atributos, exceto aqueles que são considerados suas propriedades).

Prioridade: Essencial

Entradas e pré-condições: Deve receber como entrada o componente que se deseja visualizar.

Saídas e pós-condições: O usuário visualiza o componente desejado.

RF002 - Cadastrar Food Truck

Descrição: Este requisito do sistema permite que o usuário crie e armazene um novo food truck.

Prioridade: Essencial

Entradas e pré-condições: Não há.

Saídas e pós-condições: Um componente é cadastrado no sistema

RF003 - Editar Food Truck

Descrição: Este requisito do sistema permite que o usuário edite dados de um food truck.

Prioridade: Essencial

Entradas e pré-condições: Recebe como entrada o componente que esteja visualizando.

Saídas e pós-condições: O usuário consegue editar o componente desejado.

RF004 - Excluir Food Truck

Descrição: Este requisito do sistema permite que o usuário exclua o food truck desejado.

Prioridade: Essencial

Entradas e pré-condições: Recebe como entrada o componente que esteja visualizando.

Saídas e pós-condições: O usuário consegue excluir o componente desejado.

3.2.1.2 Cadastro de Cliente

RF005 - Cadastrar Cliente
Descrição: Este requisito do sistema permite que o usuário crie e armazene um novo cliente.
Prioridade: Essencial
Entradas e pré-condições: Não há.
Saídas e pós-condições: Um componente é cadastrado no sistema.

RF006 - Editar Cliente
Descrição: Este requisito do sistema permite que o usuário edite dados de um cliente.
Prioridade: Essencial
Entradas e pré-condições: Recebe como entrada o componente que esteja visualizando.
Saídas e pós-condições: O usuário consegue editar o componente que deseja.

RF007 - Excluir Cliente
Descrição: Este requisito do sistema permite que o usuário exclua o cliente desejado.
Prioridade: Essencial
Entradas e pré-condições: Recebe como entrada o componente que esteja visualizando.
Saídas e pós-condições: O usuário consegue excluir o componente desejado.

3.2.1.3 Cadastro de Gerente (Aplicativo food truck para Owners)

RF008 - Cadastrar Gerente
Descrição: Este requisito do sistema permite que o usuário crie e armazene um novo gerente.
Prioridade: Essencial
Entradas e pré-condições: Não há
Saídas e pós-condições: Um componente é cadastrado no sistema

RF009 - Editar Gerente
Descrição: Este requisito do sistema permite que o usuário edite dados de um gerente.
Prioridade: Essencial

Entradas e pré-condições: Recebe como entrada o componente que esteja visualizando.

Saídas e pós-condições: O usuário consegue editar o componente que deseja.

RF010 - Excluir Gerente

Descrição: Este requisito do sistema permite que o usuário exclua o gerente desejado.

Prioridade: Essencial

Recebe como entrada o componente que esteja visualizando.

Saídas e pós-condições: O usuário consegue excluir o componente desejado.

3.2.1.4 Visualização de Históricos

RF011 - Consultar Histórico

Descrição: Este requisito do sistema permite que o usuário visualize os dados de um determinado componente (todos os seus atributos, exceto aqueles que são considerados suas propriedades).

Prioridade: Essencial

Entradas e pré-condições: Deve receber como entrada o componente que se deseja visualizar.

Saídas e pós-condições: O usuário visualiza o componente desejado.

3.2.1.5 Cadastro de Menus (Aplicativo food truck para Owners)

RF012 - Adicionar Menus

Descrição: Este requisito do sistema permite que o usuário crie e armazene um novo menu.

Prioridade: Essencial

Entradas e pré-condições: Não há.

Saídas e pós-condições: Um componente é cadastrado no sistema.

RF013 - Editar Menus

Descrição: Este requisito do sistema permite que o usuário faça edição de um menu.

Prioridade: Essencial

Entradas e pré-condições: Deve receber como entrada o componente que esteja visualizando.
Saídas e pós-condições: O usuário consegue editar o componente desejado.

RF014 - Consultar Menus (Para ambos aplicativos: Clientes e Owners)
Descrição: Este requisito do sistema permite que o usuário visualize os dados de um determinado menu (todos os seus atributos, exceto aqueles que são considerados suas propriedades)
Prioridade: Essencial
Entradas e pré-condições: Deve receber como entrada o componente que se deseja visualizar.
Saídas e pós-condições: O usuário visualiza o componente desejado.

RF015 - Excluir Menus
Descrição: Este requisito do sistema permite que o usuário exclua um menu.
Prioridade: Essencial
Entradas e pré-condições: Deve receber como entrada o componente que se deseja visualizar
Saídas e pós-condições: O usuário consegue excluir o componente desejado.

3.2.1.6 Ordem de Pedidos

RF016 - Realizar Pedidos
Descrição: Este requisito do sistema permite que o usuário crie uma nova ordem de pedido.
Prioridade: Essencial
Entradas e pré-condições: Deve receber como entrada o componente que esteja visualizando
Saídas e pós-condições: O usuário consegue realizar o pedido no sistema

RF017 - Editar Pedidos
Descrição: Este requisito do sistema permite que o usuário faça a edição de um pedido.
Prioridade: Essencial
Entradas e pré-condições: Deve receber como entrada o componente que esteja visualizando.
Saídas e pós-condições: O usuário consegue editar o componente desejado

RF018 - Cancelar Pedidos
Descrição: Este requisito do sistema permite que o usuário cancele a ordem de pedido.

Prioridade: Essencial

Entradas e pré-condições: Deve receber como entrada o componente que esteja visualizando.

Saídas e pós-condições: O usuário consegue cancelar o componente desejado.

RF019 - Visualizar Pedidos

Descrição: Este requisito do sistema permite que o usuário visualize

Prioridade: Essencial

Entradas e pré-condições: Deve receber como entrada o componente que se deseja visualizar.

Saídas e pós-condições: O usuário visualiza o componente desejado.

3.2.1.7 Visualização de Food Truck

RF020 - Pesquisar Food Truck

Descrição: Este requisito do sistema permite ao usuário pesquisar pela região em que deseja visualizar os food trucks.

Prioridade: Essencial

Entradas e pré-condições: Deve receber como entrada o componente que se deseja visualizar.

Saídas e pós-condições: O usuário visualiza o componente desejado.

3.2.2 Requisitos não funcionais

Estes são os requisitos não funcionais necessários ao sistema. Eles descrevem as restrições sobre os serviços e/ou funções, ou ainda no processo de desenvolvimento.

NF001 - Usabilidade

A interface com o usuário é de vital importância para o sucesso do sistema, para isso o aplicativo mobile deve contar com um layout intuitivo e com profundidade máxima de dois níveis para menus e seleção de opções.

NF002 - Desempenho

Embora não seja um requisito essencial ao sistema, deve ser considerada por corresponder a um fator de qualidade de software. O aplicativo mobile deve rodar com hardware de no mínimo um gigabyte de memória RAM.

NF003 - Hardware e Software

Avaliando a necessidade de agilidade e confiabilidade do sistema, se faz necessário um aplicativo cliente que rode sobre plataforma mobile e use conectividade 3/4G e wi-fi. A solução deve ser capaz de trabalhar de forma assíncrona, permitindo o envio dos dados coletados de forma automática. Além disso os dados armazenados no dispositivo móvel devem ser criptografados para impedir acesso indevido no caso de perda, por exemplo. Quanto ao Software de base, ou seja, o sistema operacional, é imprescindível que o sistema rode sobre em versões android 4.1 e IOS 5 ou superior.

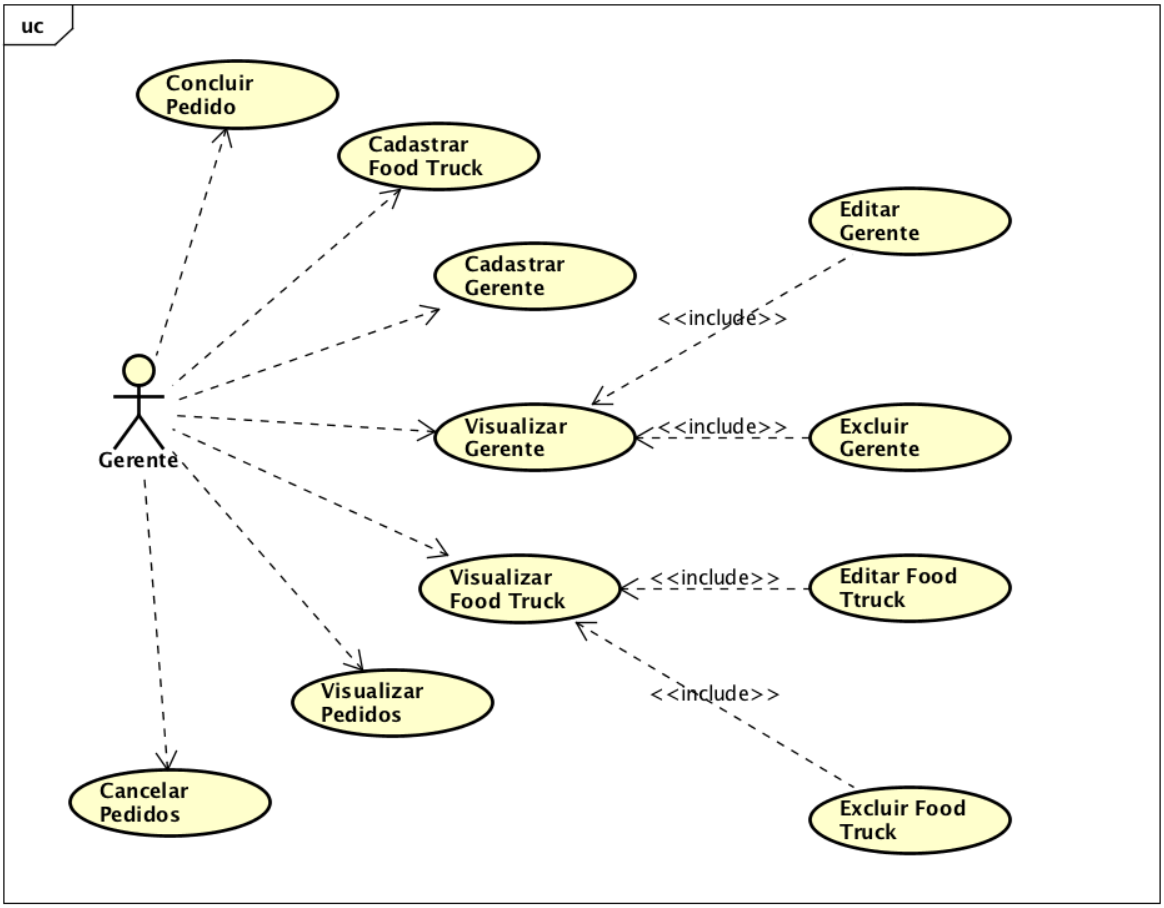
3.3 Casos de uso

Como parte do desenvolvimento do sistema, diagramas de UML foram desenvolvidos para apoiar o desenvolvimento do aplicativo.

3.3.1.1 Caso de Uso – Gerente

A Figura 21 ilustra o diagrama de casos de uso do Gerente. A descrição dos casos de uso é apresentada a seguir.

Figura 11 - Diagrama de casos de uso (Aplicativo Food Truck para Owners)



3.3.1.2 Descrição Detalhada – Gerente

Nome do Caso de Uso: Cadastrar Food Truck

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema cadastrar um ponto de food truck.

Eventos:

- Usuário solicita um novo cadastro
- Sistema solicita os dados do food truck
- Usuário informa as informações do food truck
- Usuário solicita o cancelamento do cadastro

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Um ponto de food truck cadastrado no sistema.

2. Conclusões sem sucesso:

- Não cadastrado, falta de informações.

Fluxo básico:

1. O gerente pressiona a opção de “Cadastrar Food Truck”.
2. O sistema retorna formulário de cadastro solicitando informações.
3. O gerente insere as informações do food truck.
4. O sistema valida os dados.
5. O gerente pressiona o botão “Salvar”.
6. O sistema insere e armazena o ponto de food truck.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O gerente informa dado válido ou cancela o cadastro (caso não tenha o dado correto).

Nome do Caso de Uso: Editar Food Truck

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema editar food truck.

Eventos:

- Gerente solicita editar food truck;
- Gerente informa novas informações do food truck;

- Gerente solicita o cancelamento da edição;

Atores:
- Gerente

Pré-Condições:
- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O Gerente deve encontrar o food truck que deseja editar.

Pós-Condições:
1. Conclusões com sucesso:
- As informações do food truck são editadas.
2. Conclusões sem sucesso:
- Não cadastrado, falta de informações ou informações inválidas.

Fluxo básico:
1. O Gerente pressiona o botão “Editar” referente ao food truck que deseja editar.
2. O sistema retorna as informações do food truck.
3. O gerente insere os novos dados do food truck.
4. O sistema valida os dados.
5. O Gerente pressiona o botão “Salvar”.
6. O sistema insere e atualiza a food truck.

Fluxos alternativos:
A1: Alternativa do passo 5 – Dados não validados.
A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido (s).
A1.2. O gerente informa o dado válido ou cancela a edição (caso não tenha o dado correto).

A2: Alternativa do passo 4 - " Gerente cancela edição"

A1.1. O Gerente clica em cancelar edição

A1.2. Sistema retorna ao passo 1 do fluxo principal.

Nome do Caso de Uso: Visualizar Food Truck

Descrição:

Este caso de uso tem por objetivo permitir ao Gerente do sistema visualizar um food truck.

Eventos:

- Usuário solicita visualizar food truck
- Sistema retorna food truck

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- Um food truck é visualizado.

2. Conclusões sem sucesso:
- Sem cadastro de food truck.

Fluxo básico:

1. O Gerente pressiona o botão “Visualizar”.
2. O sistema retorna todos as informações do food truck exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum cadastro de food truck.

A1.1. O sistema não encontra o food truck cadastrado e exibe a mensagem “Nenhum cadastro de food truck.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Excluir Food Truck

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema excluir um ponto de food truck.

Eventos:

- Usuário solicita exclusão do ponto de food truck
- Sistema solicita confirmação

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O usuário deve encontrar o ponto de food truck que deseja excluir.

Pós-Condições:

1. Conclusões com sucesso:
- Ponto de food truck excluído.
2. Conclusões sem sucesso:
- Não excluído.

Fluxo básico:

1. O gerente pressiona o botão “Excluir” referente ao ponto de food truck que deseja excluir.

2. O sistema solicita a confirmação da ação e exibe na tela a mensagem “Deseja excluir o ponto de Food Truck “X” ?”.

3. O gerente pressiona o botão “Sim”.

4. O sistema exclui o ponto de food truck do sistema.

Fluxos alternativos:

A1: Alternativa do passo 3 – Confirmação da exclusão.

A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

Nome do Caso de Uso: Concluir Pedido

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema concluir pedido realizado pelo cliente.

Eventos:

- Gerente solicita concluir pedido
- Sistema retorna mensagem de conclusão do pedido

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Um pedido é concluído.

2. Conclusões sem sucesso:

- Nenhum pedido concluído.

Fluxo básico:

1. O ator pressiona o botão “Concluir pedido”.

2. O sistema retorna mensagem "Pedido concluído" exibido na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum pedido concluído.

A1.1. O sistema não encontra o pedido realizado e exibe a mensagem “Nenhum pedido realizado.”

A1.2. Retorna ao passo 1.

Fluxos alternativos:

A1: Alternativa do passo 1 – Confirmação da conclusão do pedido.

A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

Nome do Caso de Uso: Cadastrar Gerente

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema cadastrar perfil de gerente.

Eventos:

- Usuário solicita um novo cadastro
- Sistema solicita os dados do gerente
- Usuário informa as informações do gerente
- Usuário solicita o cancelamento do cadastro

Atores:

- Usuário

Pré-Condições:

- O sistema deve estar em execução.

Pós-Condições:

1. Conclusões com sucesso:

- Um perfil de gerente cadastrado no sistema.

2. Conclusões sem sucesso:

- Não cadastrado, falta de informações.

Fluxo básico:

1. O usuário pressiona a opção de “Cadastrar Gerente”.

2. O sistema retorna formulário de cadastro solicitando informações.

3. O usuário insere as informações do perfil.

4. O sistema valida os dados.

5. O usuário pressiona o botão “Salvar”.

6. O sistema insere e armazena o perfil de gerente.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O usuário informa dado válido ou cancela o cadastro (caso não tenha o dado correto).

Nome do Caso de Uso: Editar Gerente

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema editar perfil de gerente.

Eventos:

- Gerente solicita editar perfil;
- Gerente informa novas informações do perfil;
- Gerente solicita o cancelamento da edição;

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- As informações do perfil são editadas.
2. Conclusões sem sucesso:
- Não cadastrado, falta de informações ou informações inválidas.

Fluxo básico:

1. O Gerente pressiona o botão “Editar” referente ao perfil que deseja editar.
2. O sistema retorna as informações do perfil do gerente.
3. O gerente insere os novos dados do perfil.
4. O sistema valida os dados.
5. O Gerente pressiona o botão “Salvar”.
6. O sistema insere e atualiza o perfil.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.
A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido (s).
A1.2. O gerente informa o dado válido ou cancela a edição (caso não tenha o dado correto).
A2: Alternativa do passo 4 - " Gerente cancela edição"
A1.1. O Gerente clica em cancelar edição
A1.2. Sistema retorna ao passo 1 do fluxo principal.

Nome do Caso de Uso: Visualizar Gerente

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema visualizar perfil do gerente.

Eventos:
- Usuário solicita visualizar perfil
- Sistema retorna o perfil do gerente

Atores:
- Gerente

Pré-Condições:
- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:
1. Conclusões com sucesso:
- Um perfil de gerente é visualizado.

2. Conclusões sem sucesso:
- Sem cadastro no sistema.

Fluxo básico:
1. O gerente pressiona o botão “Visualizar”.
2. O sistema retorna todos as informações do perfil exibidos na tela.

Fluxos alternativos:
A1: Alternativa do passo 2 – Nenhum cadastro no sistema.
A1.1. O sistema não encontra o cadastro e exibe a mensagem “Nenhum cadastro no sistema.”
A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Excluir Gerente
Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema excluir perfil de gerente.

Eventos:

- Usuário solicita exclusão do perfil de gerente
- Sistema solicita confirmação

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O usuário deve encontrar o perfil que deseja excluir.

Pós-Condições:

1. Conclusões com sucesso:
- Perfil de gerente excluído.
2. Conclusões sem sucesso:
- Não excluído.

Fluxo básico:

1. O gerente pressiona o botão “Excluir” referente ao perfil de gerente que deseja excluir.
2. O sistema solicita a confirmação da ação e exibe na tela a mensagem “Deseja excluir o perfil “X” ?”.
3. O gerente pressiona o botão “Sim”.
4. O sistema exclui o perfil do sistema.

Fluxos alternativos:

A1: Alternativa do passo 3 – Confirmação da exclusão.

A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

Nome do Caso de Uso: Cadastrar Menu

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema cadastrar um menu.

Eventos:

- Usuário solicita um novo cadastro
- Sistema solicita os dados do menu
- Usuário informa as informações do menu
- Usuário solicita o cancelamento do cadastro

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- Um menu cadastrado no sistema.

2. Conclusões sem sucesso:
- Não cadastrado, falta de informações.

Fluxo básico:

1. O gerente pressiona a opção de “Cadastrar Menu”.
2. O sistema retorna formulário de cadastro solicitando informações.
3. O Gerente insere as informações do menu.
4. O sistema valida os dados.
5. O gerente pressiona o botão “Salvar”.
6. O sistema insere e armazena o menu.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O Gerente informa dado válido ou cancela o cadastro (caso não tenha o dado correto).

Nome do Caso de Uso: Editar Menu

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema editar menu.

Eventos:

- Gerente solicita editar menu
- Gerente informa novas informações do menu
- Gerente solicita o cancelamento da edição;

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O Gerente deve encontrar o menu que deseja editar.

Pós-Condições:

1. Conclusões com sucesso:
- As informações do food truck são editadas.
2. Conclusões sem sucesso:
- Não cadastrado, falta de informações ou informações inválidas.

Fluxo básico:

1. O Gerente pressiona o botão “Editar” referente ao menu que deseja editar.

2. O sistema retorna as informações do menu.
3. O gerente insere os novos dados do menu.
4. O sistema valida os dados.
5. O Gerente pressiona o botão “Salvar”.
6. O sistema insere e atualiza o menu.

Fluxos alternativos:
A1: Alternativa do passo 5 – Dados não validados.
A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido (s).
A1.2. O gerente informa o dado válido ou cancela a edição (caso não tenha o dado correto).
A2: Alternativa do passo 4 - " Gerente cancela edição"
A1.1. O Gerente clica em cancelar edição
A1.2. Sistema retorna ao passo 1 do fluxo principal.

Nome do Caso de Uso: Excluir menu
Descrição:
Este caso de uso tem por objetivo permitir ao usuário do sistema excluir um menu.

Eventos:
- Usuário solicita exclusão do menu
- Sistema solicita confirmação

Atores:
- Gerente

Pré-Condições:
- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O usuário deve encontrar o menu que deseja excluir.

Pós-Condições:

1. Conclusões com sucesso:

- Menu excluído.

2. Conclusões sem sucesso:

- Não excluído.

Fluxo básico:

1. O gerente pressiona o botão “Excluir Menu” referente ao menu que deseja excluir.

2. O sistema solicita a confirmação da ação e exibe na tela a mensagem “Deseja excluir o menu “X” ?”.

3. O gerente pressiona o botão “Sim”.

4. O sistema exclui o menu do sistema.

Fluxos alternativos:

A1: Alternativa do passo 3 – Confirmação da exclusão.

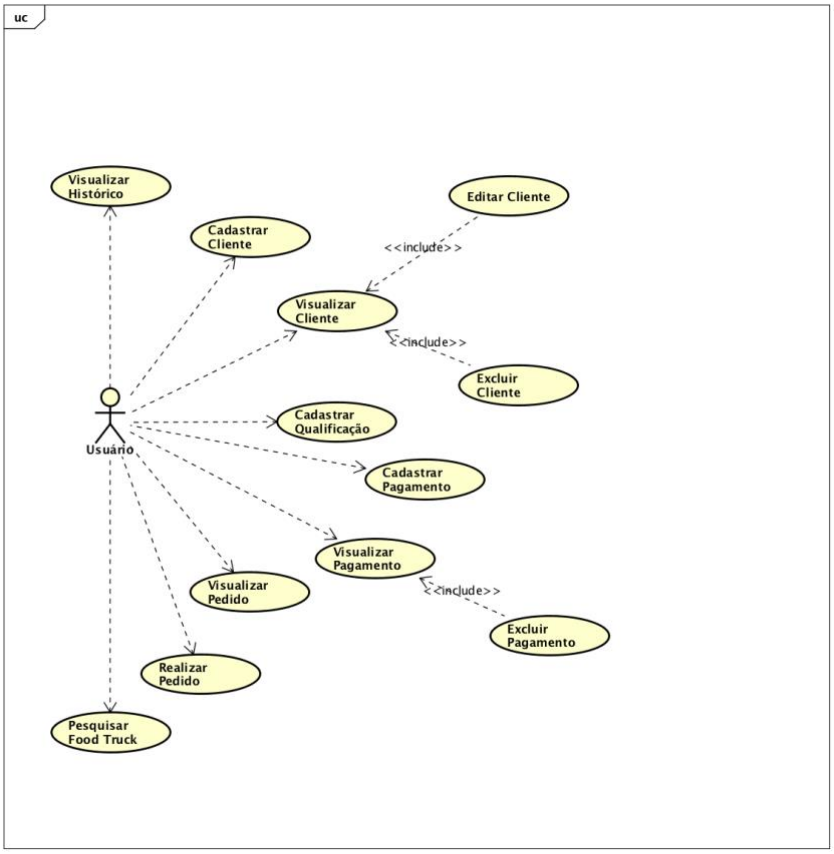
A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

3.3.1.3 Caso de Uso – Cliente

A Figura 22 ilustra o diagrama de casos de uso do Cliente. A descrição dos casos de uso é apresentada a seguir.

Figura 22 - Diagrama de casos de uso (Aplicativo Food Truck para Cliente)



3.3.1.4 Descrição Detalhada – Cliente

Nome do Caso de Uso: Cadastrar Cliente

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema cadastrar um perfil de cliente.

Eventos:

- Usuário solicita um novo cadastro
- Sistema solicita os dados do cliente
- Usuário informa as informações do cliente
- Usuário solicita o cancelamento do cadastro

Atores:

- Usuário

Pré-Condições:

- O sistema deve estar em execução.

Pós-Condições:

1. Conclusões com sucesso:

- Um cliente cadastrado no sistema.

2. Conclusões sem sucesso:

- Não cadastrado, falta de informações.

Fluxo básico:

1. O usuário pressiona a opção de “Cadastrar”.
2. O sistema retorna formulário de cadastro solicitando informações.
3. O usuário insere as informações do cliente.
4. O sistema valida os dados.
5. O usuário pressiona o botão “Salvar”.
6. O sistema insere e armazena um cliente.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O usuário informa dado válido ou cancela o cadastro (caso não tenha o dado correto).

Nome do Caso de Uso: Editar Cliente

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema editar perfil de cliente.

Eventos:

- Cliente solicita editar perfil cliente;
- Cliente informa novas informações do perfil cliente;
- Cliente solicita o cancelamento da edição;

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- As informações do perfil são editadas.
2. Conclusões sem sucesso:
- Não cadastrado, falta de informações ou informações inválidas.

Fluxo básico:

1. O Cliente pressiona o botão “Editar Perfil” referente ao perfil que deseja editar.
2. O sistema retorna as informações do perfil.
3. O Cliente insere os novos dados do perfil.
4. O sistema valida os dados.
5. O Cliente pressiona o botão “Salvar”.
6. O sistema insere e atualiza o perfil do cliente.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido (s).

A1.2. O cliente informa o dado válido ou cancela a edição (caso não tenha o dado correto).

A2: Alternativa do passo 4 - " Cliente cancela edição"

A1.1. O cliente clica em cancelar edição

A1.2. Sistema retorna ao passo 1 do fluxo principal.

Nome do Caso de Uso: Visualizar Cliente

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema visualizar perfil do cliente.

Eventos:

- Usuário solicita visualizar perfil
- Sistema retorna o perfil do cliente

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- Um perfil de cliente é visualizado.

2. Conclusões sem sucesso:
- Sem cadastro no sistema.

Fluxo básico:

1. O cliente pressiona o botão “Visualizar”.
2. O sistema retorna todos as informações do perfil exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum cadastro no sistema.

A1.1. O sistema não encontra o cadastro e exibe a mensagem “Nenhum cadastro no sistema.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Excluir Cliente

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema excluir perfil de cliente.

Eventos:

- Usuário solicita exclusão do perfil de cliente
- Sistema solicita confirmação

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O usuário deve encontrar o perfil que deseja excluir.

Pós-Condições:

1. Conclusões com sucesso:
- Perfil de cliente excluído.
2. Conclusões sem sucesso:
- Não excluído.

Fluxo básico:

1. O cliente pressiona o botão “Excluir” referente ao perfil de cliente que deseja excluir.
2. O sistema solicita a confirmação da ação e exibe na tela a mensagem “Deseja excluir o perfil “X” ?”.
3. O cliente pressiona o botão “Sim”.
4. O sistema exclui o perfil do sistema.

Fluxos alternativos:

A1: Alternativa do passo 3 – Confirmação da exclusão.

A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

Nome do Caso de Uso: Consultar Histórico

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema visualizar o histórico de ações do usuário.

Eventos:

- Ator solicita visualizar histórico
- Sistema retorna o histórico do ator

Atores:

- Cliente
- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- Histórico visualizado.

2. Conclusões sem sucesso:
- Sem histórico no sistema.

Fluxo básico:

1. O ator pressiona o botão “Visualizar histórico”.
2. O sistema retorna todos as informações do histórico exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum histórico no sistema.

A1.1. O sistema não encontra o histórico e exibe a mensagem “Nenhum histórico no sistema.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Visualizar Menu

Descrição:

Este caso de uso tem por objetivo permitir ao Gerente do sistema visualizar um menu.

Eventos:

- Usuário solicita visualizar menu
- Sistema retorna menu

Atores:

- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- Um menu é visualizado.

2. Conclusões sem sucesso:
- Sem cadastro de food truck.

Fluxo básico:

1. O Gerente pressiona o botão “Visualizar Menu”.
2. O sistema retorna todos as informações do menu exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum cadastro de menu.

A1.1. O sistema não encontra o menu e exibe a mensagem “Nenhum cadastro de menu.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Realizar Pedido

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema realizar um ordem de pedido.

Eventos:

- Usuário solicita um novo pedido
- Sistema solicita os dados do pedido
- Usuário informa as informações do pedido
- Usuário solicita o cancelamento do pedido

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Um pedido realizado no sistema.

2. Conclusões sem sucesso:

- Operação não realizada, falta de informações.

Fluxo básico:

1. O cliente pressiona a opção de “Realizar Pedido”.
2. O sistema retorna formulário de pedido solicitando informações.

3. O cliente insere as informações do pedido.

4. O sistema valida os dados.

5. O cliente pressiona o botão “finalizar”.

6. O sistema insere e armazena o pedido.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O cliente informa dado válido ou cancela o pedido (caso não tenha o dado correto).

Nome do Caso de Uso: Visualizar Pedido

Descrição:

Este caso de uso tem por objetivo permitir ao ator do sistema visualizar um pedido.

Eventos:

- Usuário solicita visualizar pedido
- Sistema retorna pedido

Atores:

- Gerente
- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Um pedido é visualizado.

2. Conclusões sem sucesso:

- Nenhum pedido realizado.

Fluxo básico:

1. O ator pressiona o botão “Visualizar pedido”.

2. O sistema retorna todos as informações do pedido exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum pedido realizado.

A1.1. O sistema não encontra o pedido e exibe a mensagem “Nenhum pedido realizado.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Cadastrar Pagamento

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema cadastrar um método de pagamento.

Eventos:

- Usuário solicita um novo cadastro de método de pagamento
- Sistema solicita os dados do cadastro de método de pagamento
- Usuário informa as informações do cadastro de método pagamento
- Usuário solicita o cancelamento da cadastro de método de pagamento

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Um método de pagamento cadastrado no sistema.

2. Conclusões sem sucesso:

- Não cadastrado, falta de informações.

Fluxo básico:

1. O cliente pressiona a opção “Pagamentos”.

2. O sistema retorna formulário de pagamento solicitando informações.

3. O cliente insere as informações do pagamento.

4. O sistema valida os dados.

5. O Cliente pressiona o botão “salvar”.

6. O sistema insere e armazena o método de pagamento.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O cliente informa dado válido ou cancela o cadastro (caso não tenha o dado correto).

Nome do Caso de Uso: Visualizar Pagamento

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema visualizar todas os métodos de pagamentos do usuário.

Eventos:

- Ator solicita visualizar pagamentos
- Sistema retorna método de pagamento

Atores:

- Cliente
- Gerente

Pré-Condições:

- O sistema deve estar em execução.

- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Método de pagamento visualizado.

2. Conclusões sem sucesso:

- Sem cadastro de método de pagamento.

Fluxo básico:

1. O ator pressiona o botão “Visualizar”.

2. O sistema retorna todos as informações de pagamento exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum cadastro de pagamento.

A1.1. O sistema não encontra o método de pagamento e exibe a mensagem “Nenhum cadastro de método de pagamento.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Excluir Pagamento

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema excluir método de pagamento.

Eventos:

- Usuário solicita exclusão do método de pagamento
- Sistema solicita confirmação

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O usuário deve encontrar a método de pagamento que deseja excluir.

Pós-Condições:

1. Conclusões com sucesso:
- Método de pagamento excluída.
2. Conclusões sem sucesso:
- Não excluído.

Fluxo básico:

1. O cliente pressiona o botão “Remover pagamento” referente ao método de pagamento que deseja excluir.
2. O sistema solicita a confirmação da ação e exibe na tela a mensagem “Deseja excluir o método de pagamento “X” ?”.
3. O cliente pressiona o botão “Sim”.
4. O sistema exclui o pagamento do sistema.

Fluxos alternativos:

A1: Alternativa do passo 3 – Confirmação da exclusão.

A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

Nome do Caso de Uso: Cadastrar Qualificação

Descrição: Este caso de uso tem por objetivo permitir ao usuário do sistema realizar uma qualificação do food truck.

Eventos:

- Usuário solicita uma nova qualificação
- Sistema solicita os dados para qualificação
- Usuário informa as informações da qualificação

- Usuário solicita o cancelamento da qualificação

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema.

Pós-Condições:

1. Conclusões com sucesso:
- Uma qualificação cadastrada no sistema.

2. Conclusões sem sucesso:
- Não cadastrado, falta de informações.

Fluxo básico:

1. O cliente pressiona a opção de “Qualificar Food Truck”.
2. O sistema retorna formulário de qualificação solicitando informações.
3. O cliente insere as informações do qualificação.
4. O sistema valida os dados.
5. O Cliente pressiona o botão “enviar”.
6. O sistema insere e armazena a qualificação do food truck.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido(s).

A1.2. O cliente informa dado válido ou cancela o cadastro (caso não tenha o dado correto).

Nome do Caso de Uso: Editar Qualificação

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema editar qualificação.

Eventos:

- Cliente solicita editar qualificação
- Cliente informa novas informações da qualificação;
- Cliente solicita o cancelamento da edição;

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O ator deve localizar a qualificação que deseja editar.

Pós-Condições:

1. Conclusões com sucesso:
- As informações da qualificação são editadas.
2. Conclusões sem sucesso:
- Não cadastrado, falta de informações ou informações inválidas.

Fluxo básico:

1. O Cliente pressiona o botão “Editar” referente à qualificação que deseja editar.
2. O sistema retorna as informações da qualificação.
3. O cliente insere os novos dados da qualificação.
4. O sistema valida os dados.
5. O Gerente pressiona o botão “Salvar”.
6. O sistema insere e atualiza a qualificação.

Fluxos alternativos:

A1: Alternativa do passo 5 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido (s).

A1.2. O cliente informa o dado válido ou cancela a edição (caso não tenha o dado correto).

A2: Alternativa do passo 4 - " Cliente cancela edição"

A1.1. O cliente clica em cancelar edição

A1.2. Sistema retorna ao passo 1 do fluxo principal.

Nome do Caso de Uso: Visualizar Qualificação

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema visualizar todas as qualificações do food truck.

Eventos:

- Ator solicita visualizar qualificações do food truck
- Sistema retorna todas as qualificações

Atores:

- Cliente
- Gerente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- Qualificações visualizadas.

2. Conclusões sem sucesso:

- Sem cadastro de qualificação.

Fluxo básico:

1. O ator pressiona o botão “Visualizar”.

2. O sistema retorna todos as informações de qualificações exibidos na tela.

Fluxos alternativos:

A1: Alternativa do passo 2 – Nenhum cadastro de qualificação.

A1.1. O sistema não encontra a qualificação e exibe a mensagem “Nenhum cadastro de qualificação.”

A1.2. Retorna ao passo 1.

Nome do Caso de Uso: Excluir Qualificação

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema excluir uma qualificação de food truck.

Eventos:

- Usuário solicita exclusão da qualificação
- Sistema solicita confirmação

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- O ator deve estar logado no sistema
- Deve haver dados cadastrados no sistema.
- O usuário deve encontrar a qualificação que deseja excluir.

Pós-Condições:

1. Conclusões com sucesso:

- Qualificação excluída.

2. Conclusões sem sucesso:

- Não excluído.

Fluxo básico:

1. O cliente pressiona o botão “Excluir” referente à qualificação que deseja excluir.

2. O sistema solicita a confirmação da ação e exibe na tela a mensagem “Deseja excluir qualificação “X” ?”.

3. O cliente pressiona o botão “Sim”.

4. O sistema exclui a qualificação do sistema.

Fluxos alternativos:

A1: Alternativa do passo 3 – Confirmação da exclusão.

A1.1. O usuário pressiona o botão “Não”.

A1.2. O sistema retorna para a tela anterior.

Nome do Caso de Uso: Pesquisar Food Truck

Descrição:

Este caso de uso tem por objetivo permitir ao usuário do sistema pesquisar food truck em outras regiões

Eventos:

- Cliente solicita pesquisar food truck;
- Cliente informa nome do food truck;
- Sistema retorna food truck;

Atores:

- Cliente

Pré-Condições:

- O sistema deve estar em execução.
- Deve haver dados cadastrados no sistema.

Pós-Condições:

1. Conclusões com sucesso:

- As informações do food truck são visualizadas.

2. Conclusões sem sucesso:

- Não cadastrado, falta de informações ou informações inválidas.

Fluxo básico:

1. O cliente pressiona o botão “pesquisar” referente à região em que deseja pesquisar.

2. O sistema solicita as informações para pesquisa.

3. O cliente insere os dados da região que deseja visualizar.

4. O sistema retorna o food truck.

5. O cliente pressiona o botão visualizar.

6. O sistema retorna todas as informações do food truck exibidos na tela

Fluxos alternativos:

A1: Alternativa do passo 2 – Dados não validados.

A1.1. O sistema encontra uma divergência ou falta dos dados e exibe uma mensagem na tela informando o(s) dado(s) inválido (s).

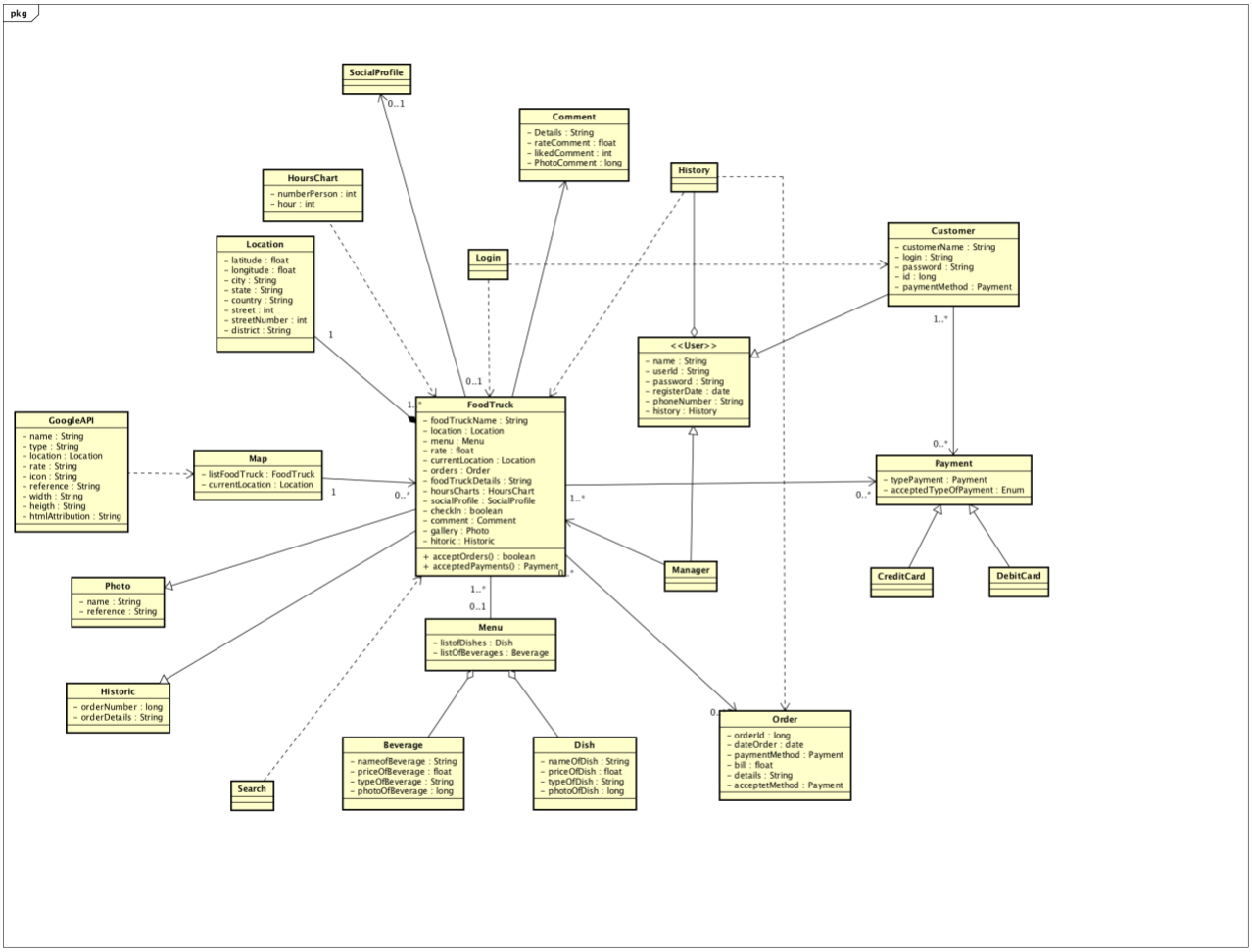
A1.2. O cliente informa o dado válido ou cancela a pesquisa (caso não tenha o dado correto).

A1.3 Sistema retorna ao passo 3 do fluxo principal.

3.4 Diagrama de Classes

A Figura 23 ilustra o diagrama de classe do trabalho proposto:

Figura 23 - Diagrama de classes do aplicativo



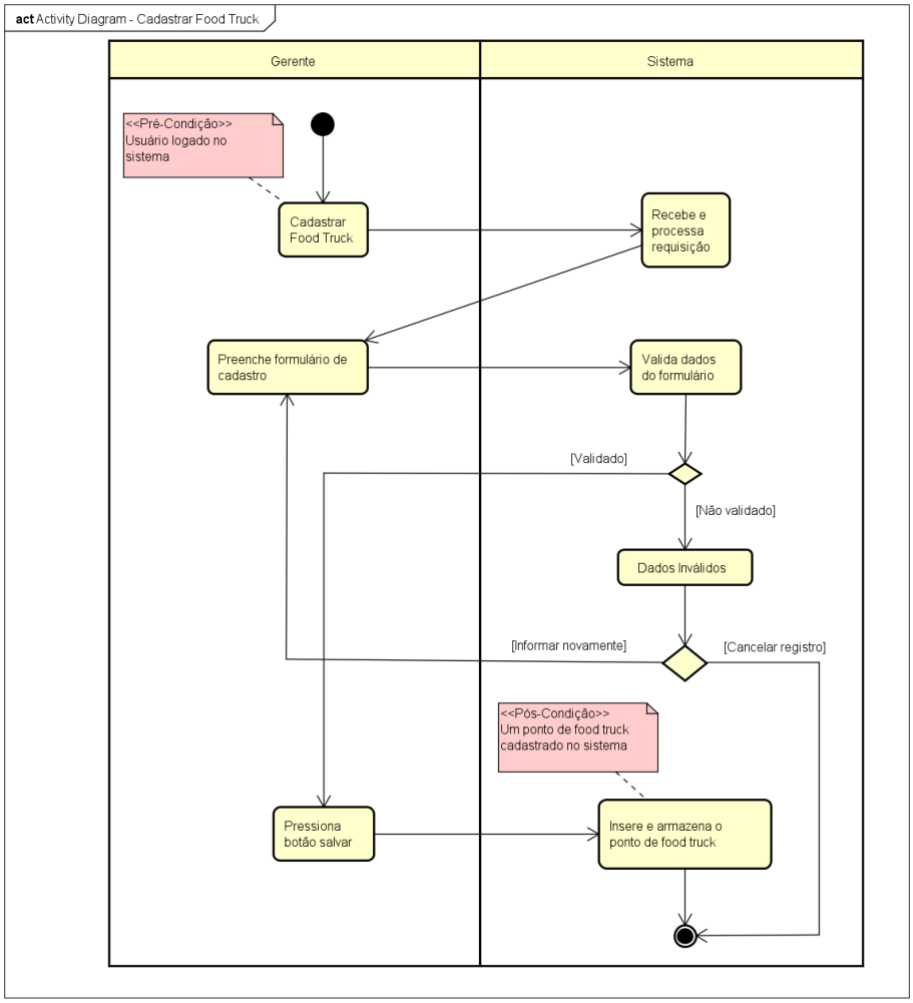
A Figura 23 apresenta a representação das classes e as interações entre si. No projeto foi adotado uma arquitetura MVC. O Model principal é o FoodTruck que é composto por outros modelos de dados como Menu, Historic, Order, Photo, Payment, Comments e Location. Os Controllers, responsáveis em servir de interface entre o View e o Model, utilizam algumas classes como auxiliares (FoodTruck, Profile, Login, Photo, Menu, Historic, Map e Payment). Todas as informações são apresentadas pelo View, através das telas desenvolvidas no trabalho.

3.5 Diagrama de atividades

Para representar o comportamento do sistema, foram desenvolvidos diagramas apresentados nesta seção.

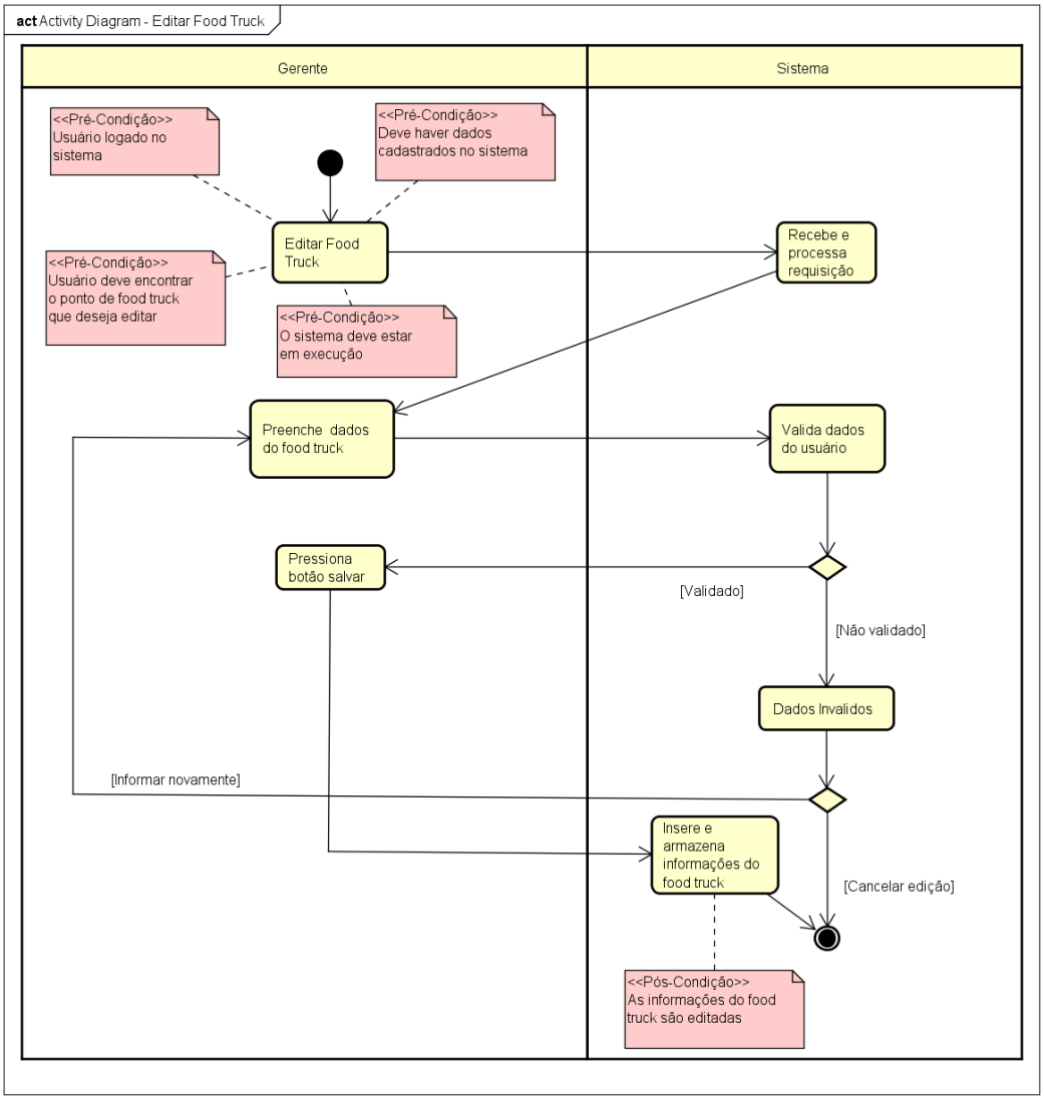
O diagrama de atividades representado na Figura 24 mostra o fluxo do cadastro do food truck.

Figura 24 - Diagrama de atividades, referente ao cadastro do food truck.



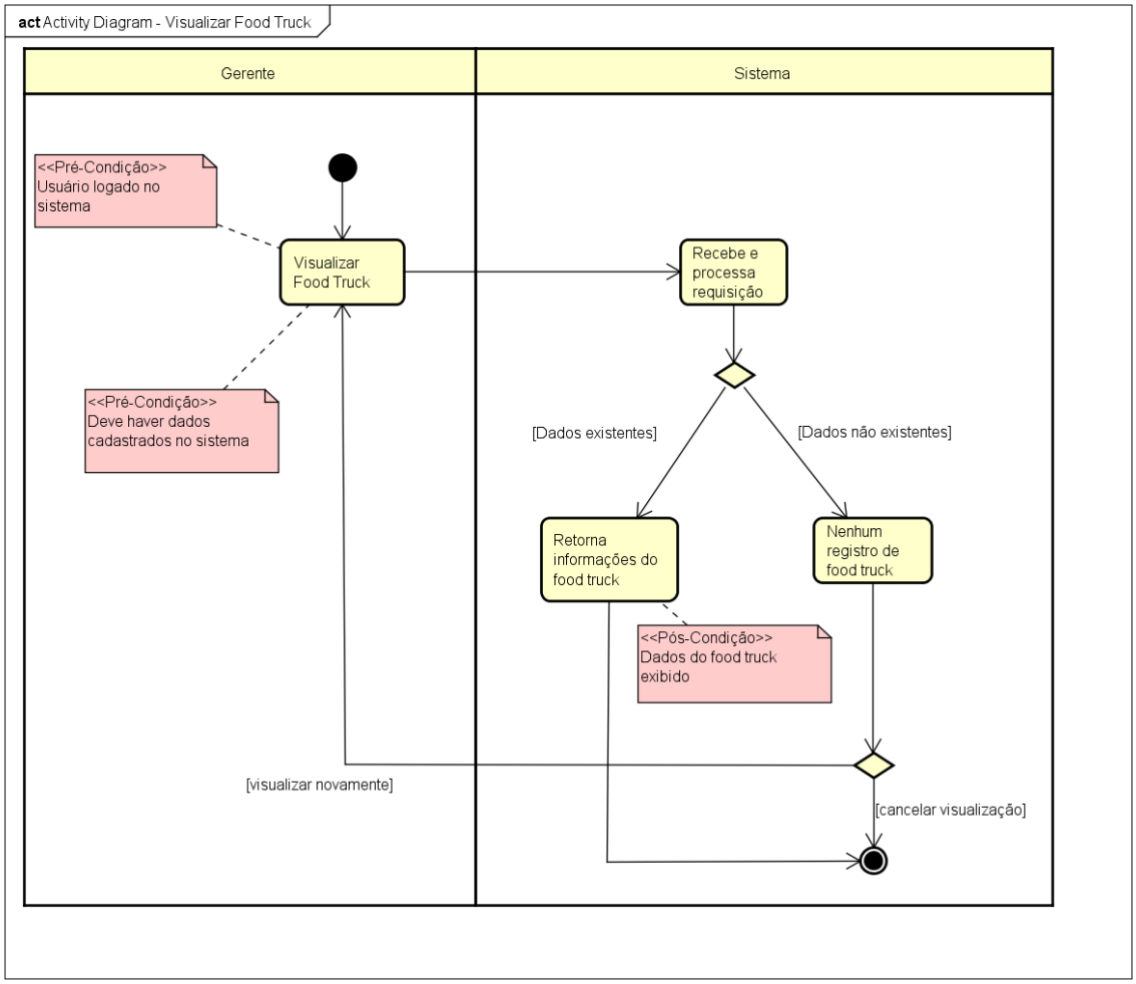
O diagrama de atividades representado na Figura 25 demonstra o fluxo da edição do food truck.

Figura 25 - Diagrama de atividades, referente ao fluxo Editar food truck.



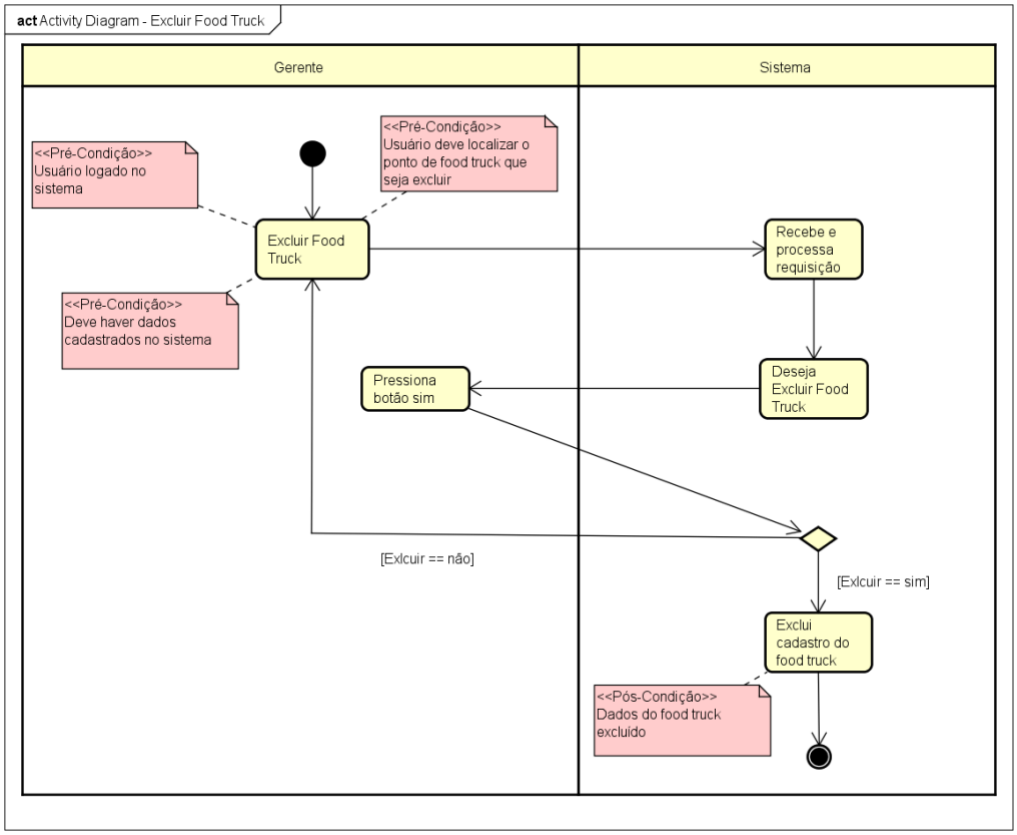
O diagrama de atividades representado na Figura 26 demonstra o fluxo visualizar food truck.

Figura 26 - Diagrama de atividades, referente ao fluxo visulizar food truck.



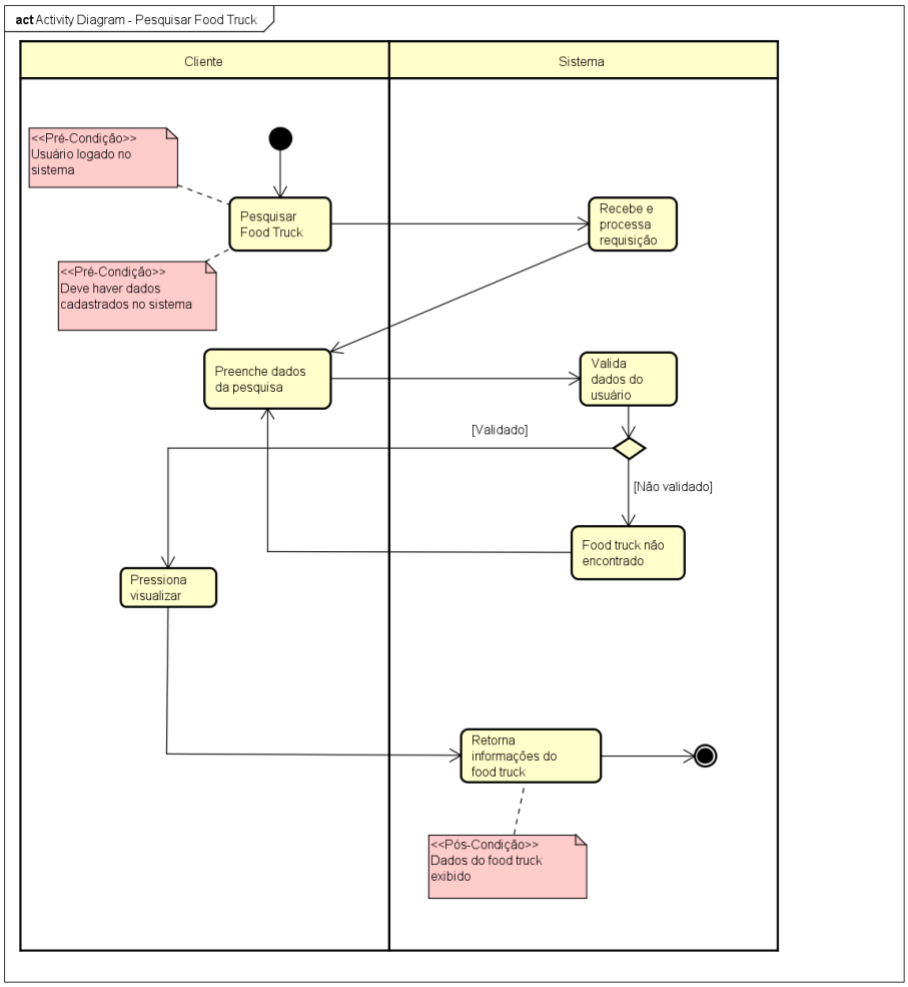
O diagrama de atividades representado na Figura 27 demonstra o fluxo exlcuir food truck.

Figura 27 - Diagrama de atividades, referente ao fluxo exlcuir food truck.



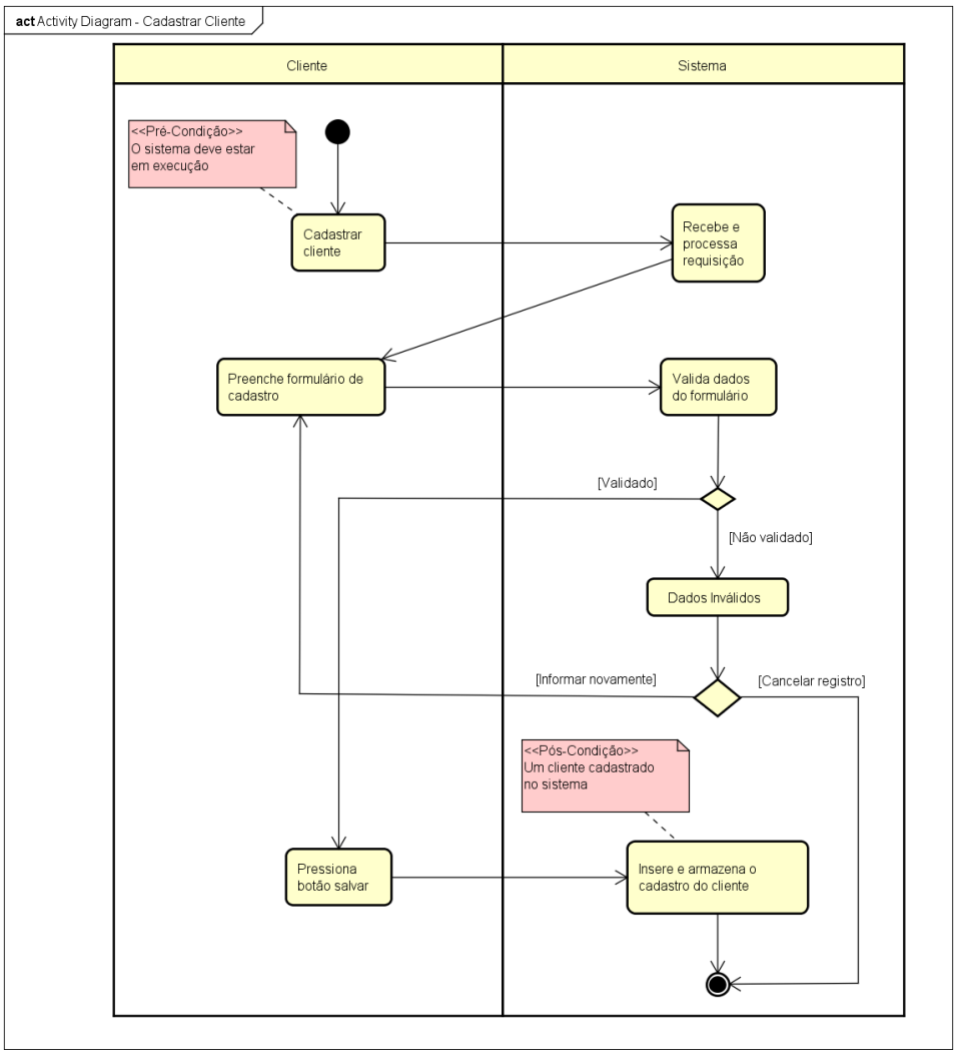
O diagrama de atividades representado na Figura 28 demonstra o fluxo pesquisar food truck.

Figura 28 - Diagrama de atividades, referente ao fluxo pesquisar food truck.



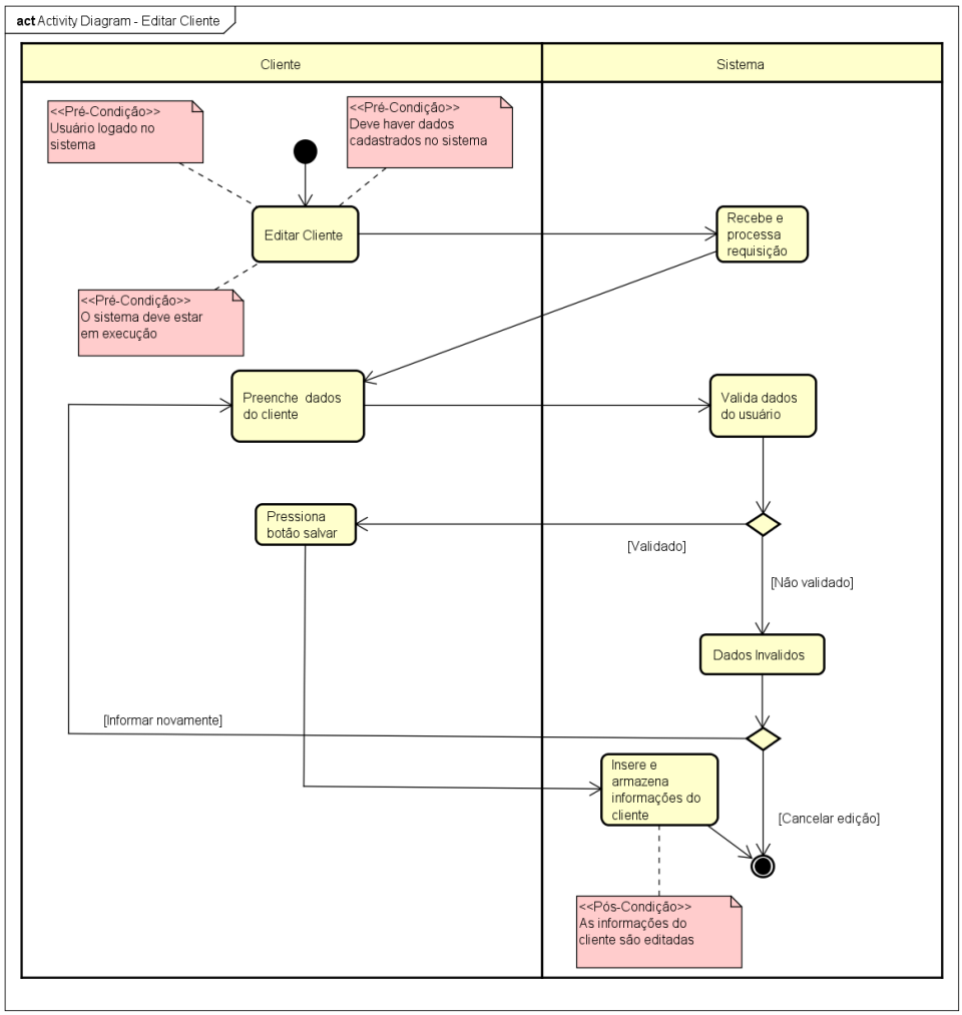
O diagrama de atividades representado na Figura 29 demonstra o fluxo cadastrar gerente.

Figura 29 - Diagrama de atividades, referente ao fluxo Editar food truck.



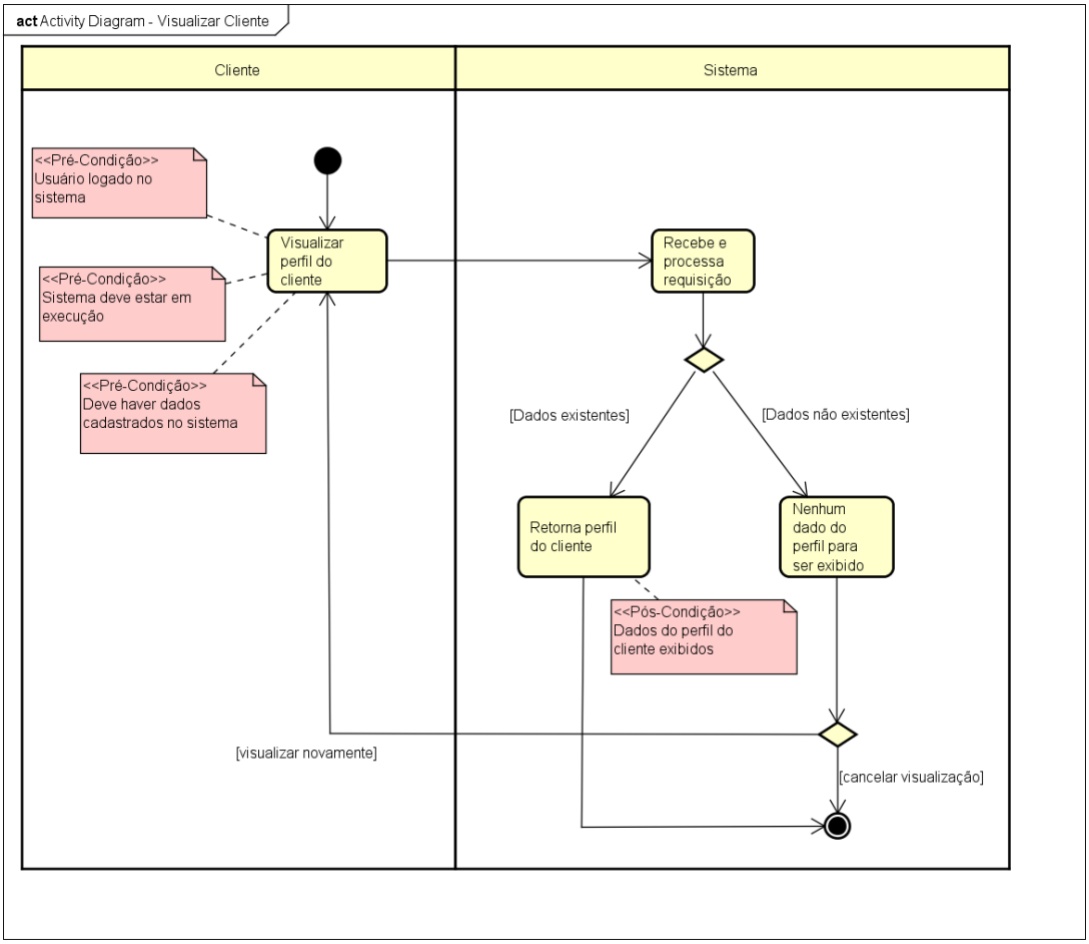
O diagrama de atividades representado na Figura 30 demonstra o fluxo editar cliente.

Figura 30 - Diagrama de atividades, referente ao fluxo editar cliente.



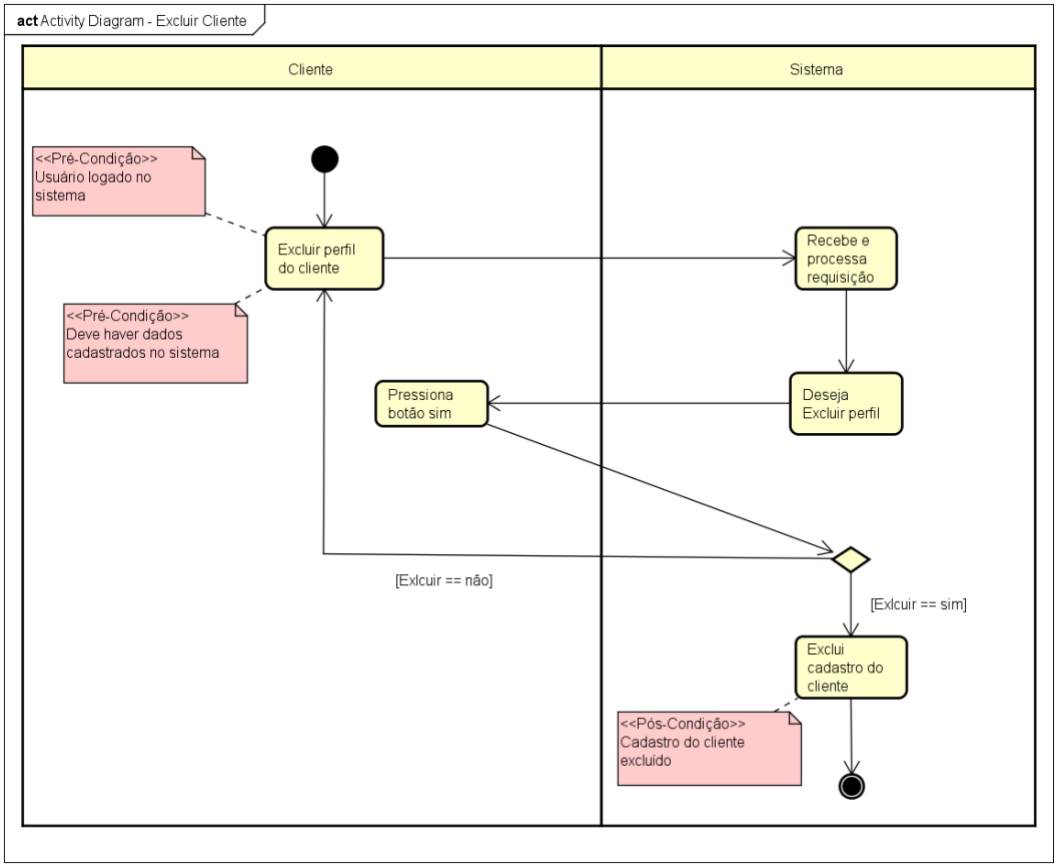
O diagrama de atividades representado na Figura 31 demonstra o fluxo visualizar cliente.

Figura 31 - Diagrama de atividades, referente ao fluxo visualizar cliente.



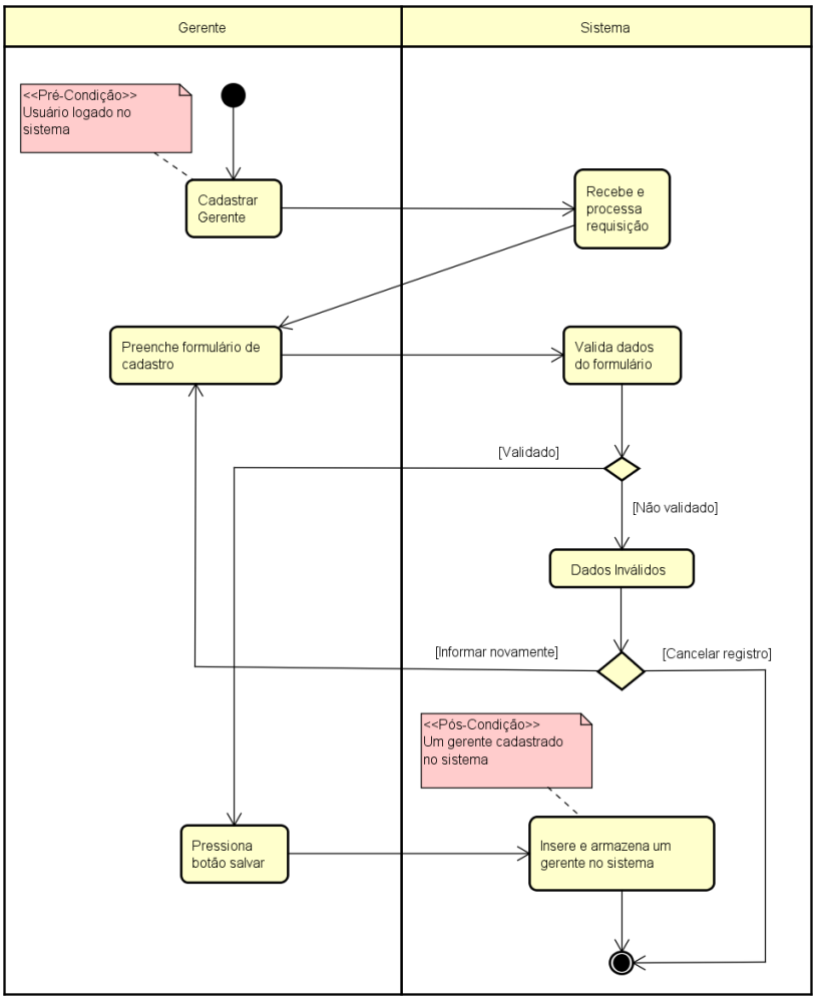
O diagrama de atividades representado na Figura 32 demonstra o fluxo excluir cliente.

Figura 32 - Diagrama de atividades, referente ao fluxo excluir cliente.



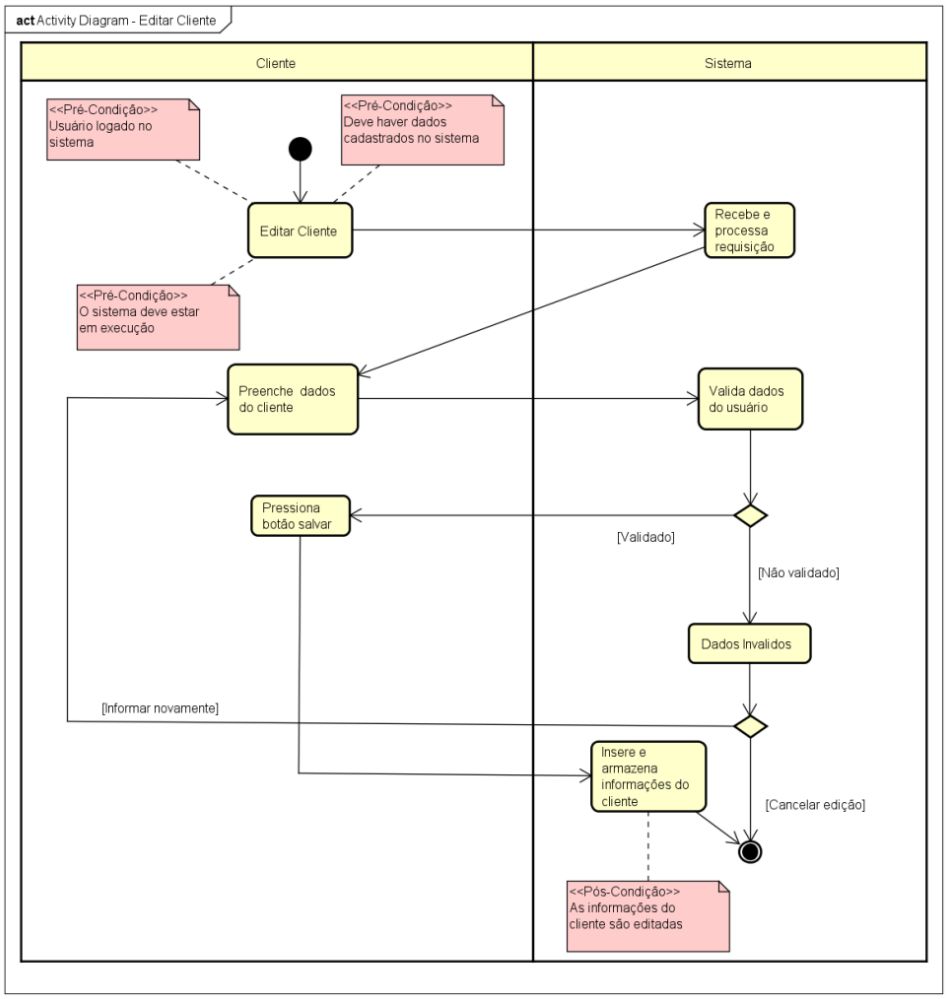
O diagrama de atividades representado na Figura 33 demonstra o fluxo cadastrar gerente.

Figura 33 - Diagrama de atividades, referente ao fluxo cadastrar gerente.



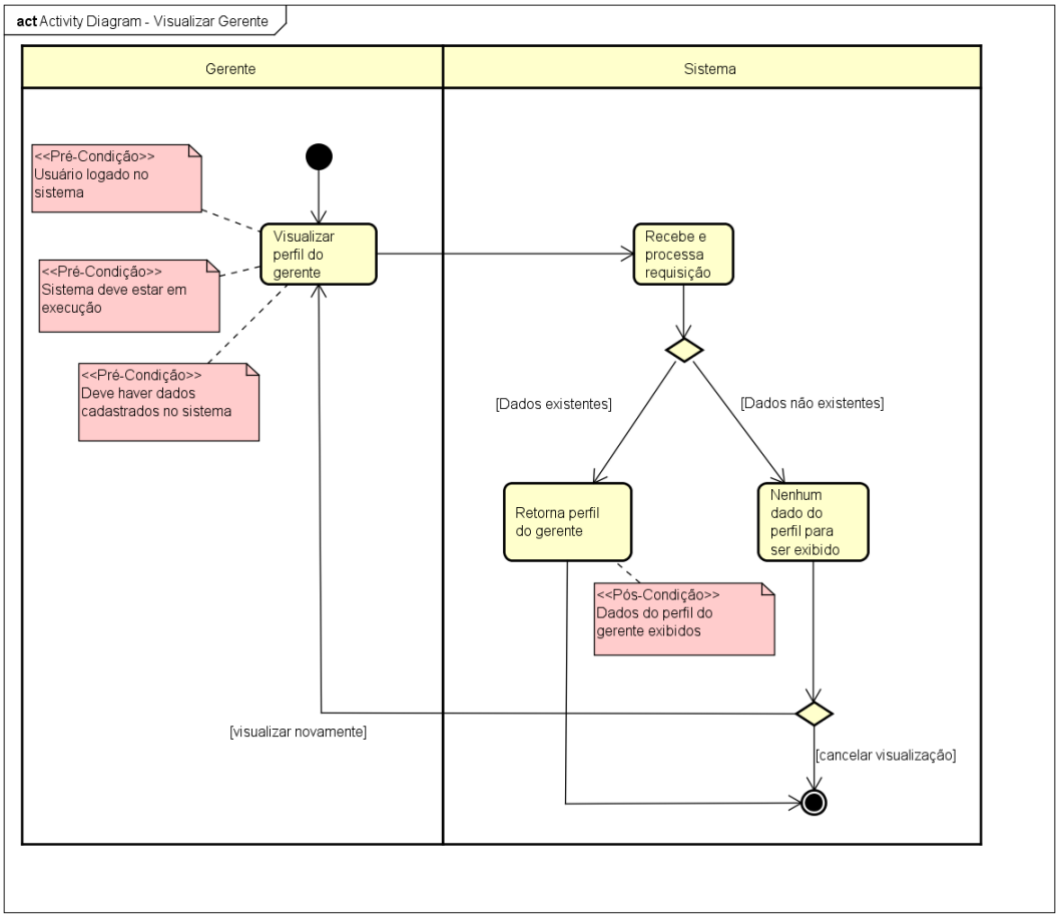
O diagrama de atividades representado na Figura 34 demonstra o fluxo editar gerente.

Figura 34 - Diagrama de atividades, referente ao fluxo editar gerente.



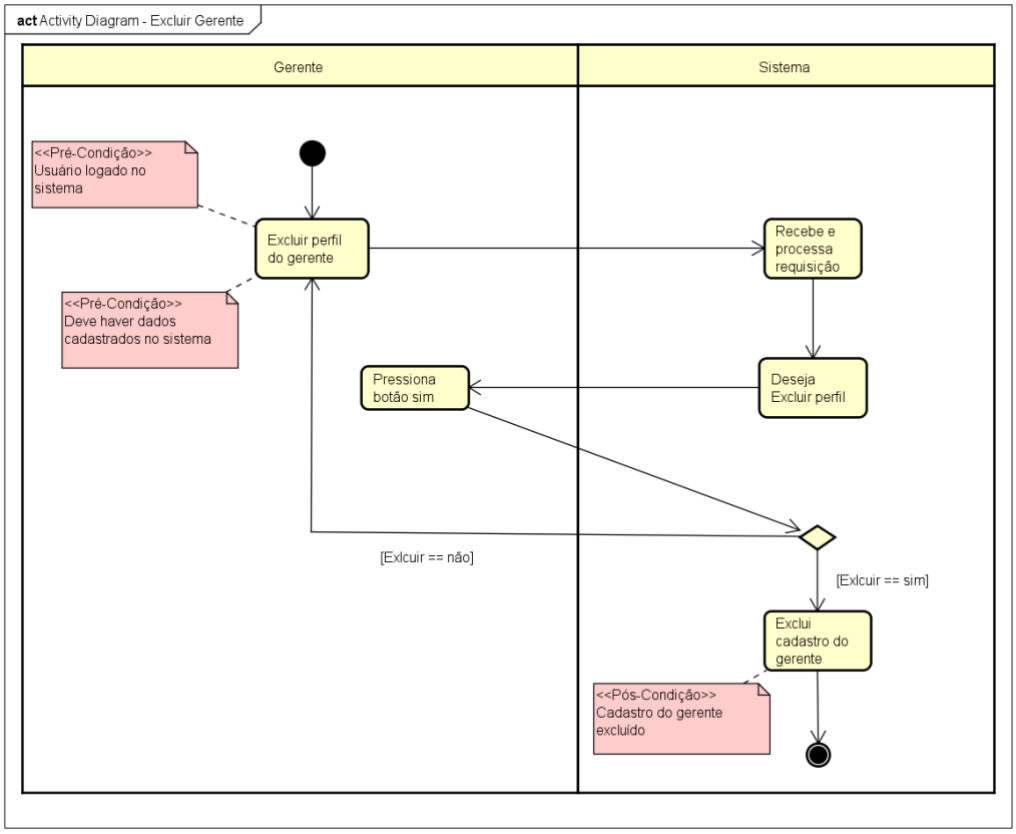
O diagrama de atividades representado na Figura 35 demonstra o fluxo visualizar gerente.

Figura 35 - Diagrama de atividades, referente ao fluxo visualizar gerente.



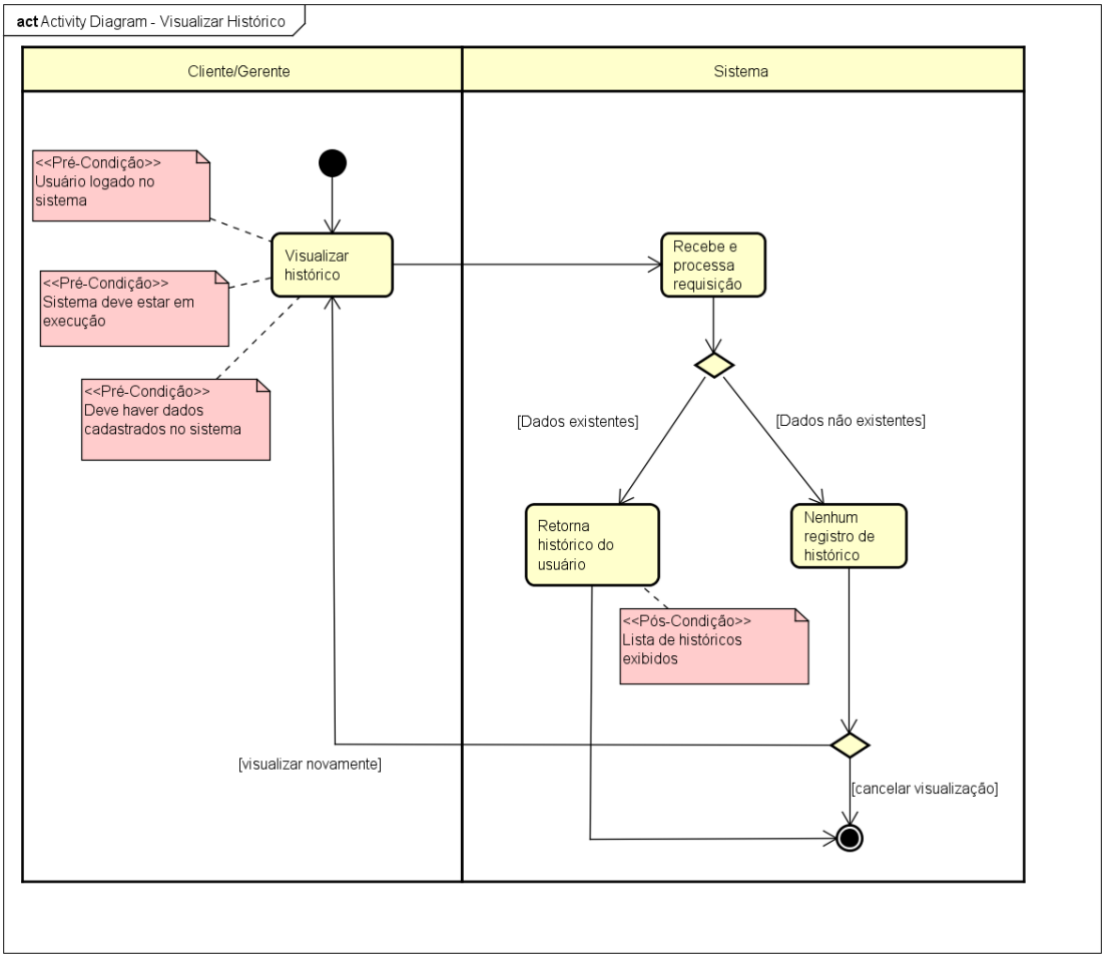
O diagrama de atividades representado na Figura 36 demonstra o fluxo excluir gerente.

Figura 36 - Diagrama de atividades, referente ao fluxo excluir gerente.



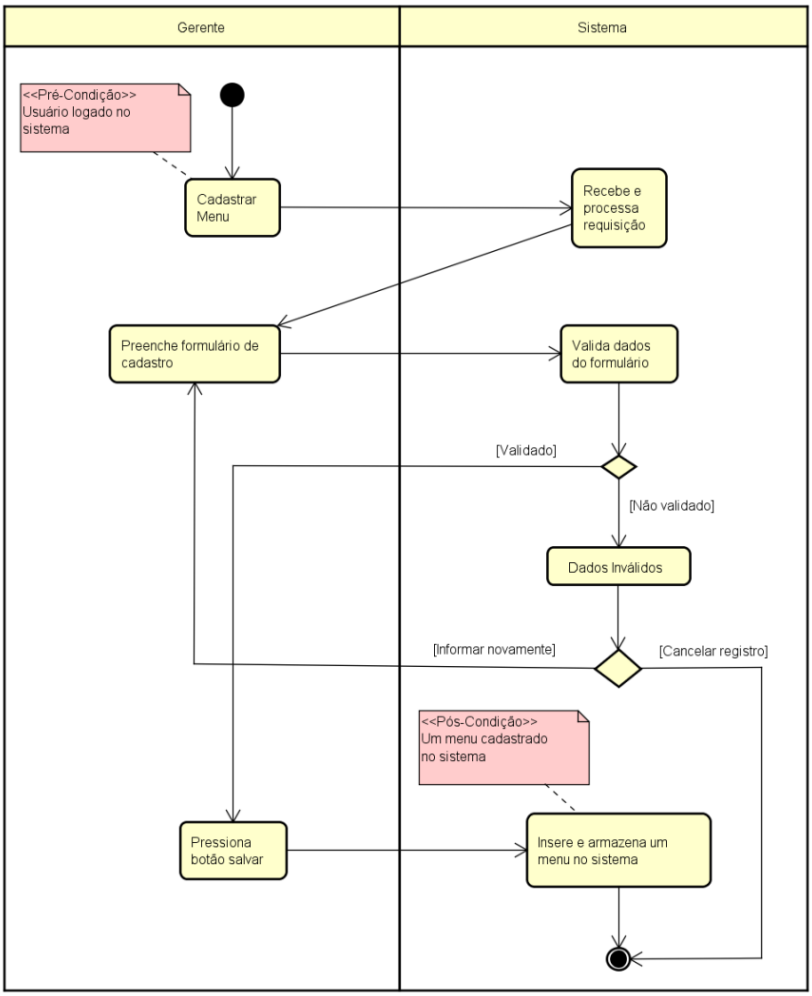
O diagrama de atividades representado na Figura 37 demonstra o fluxo consultar histórico.

Figura 37 - Diagrama de atividades, referente ao fluxo consultar histórico.



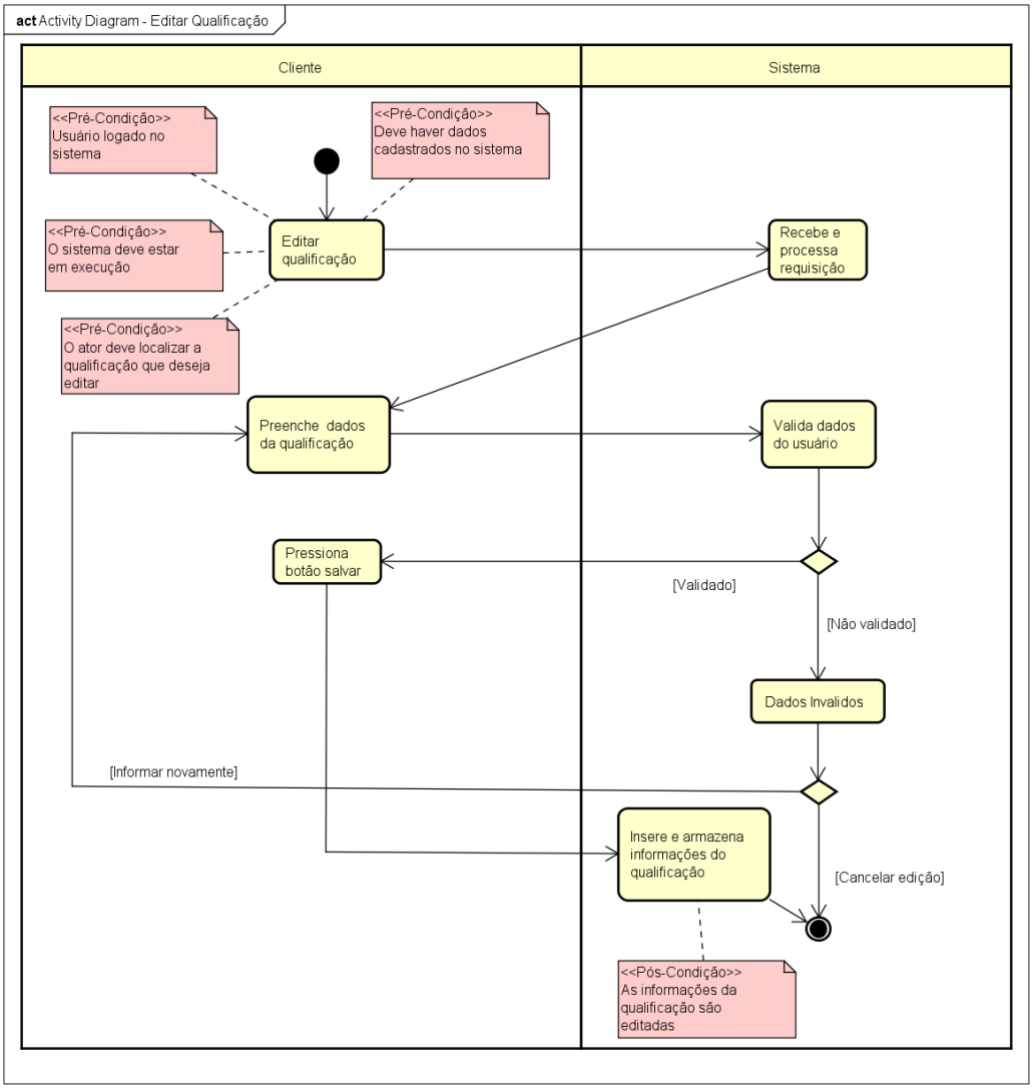
O diagrama de atividades representado na Figura 38 demonstra o fluxo cadastrar qualificação.

Figura 38 - Diagrama de atividades, referente ao fluxo cadastrar qualificação.



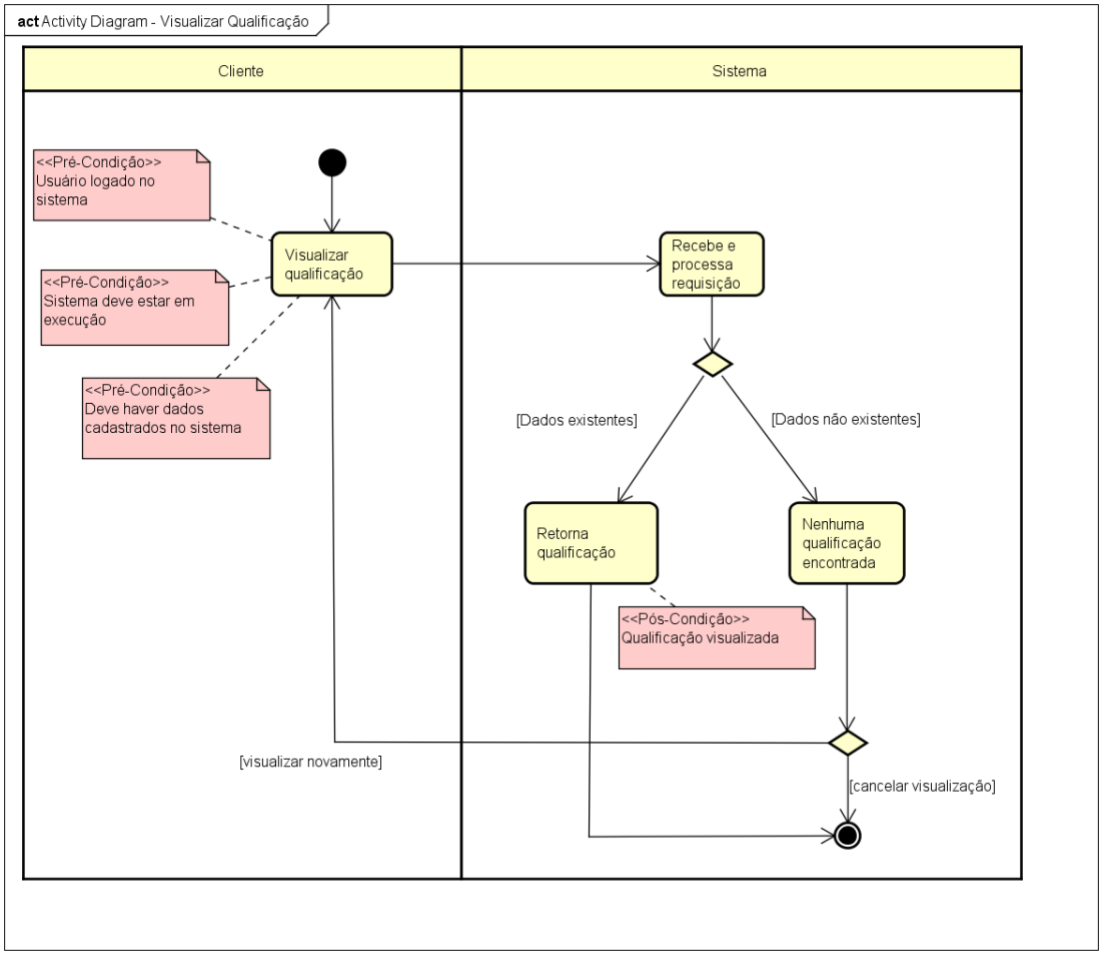
O diagrama de atividades representado na Figura 39 demonstra o fluxo editar qualificação.

Figura 39 - Diagrama de atividades, referente ao fluxo editar qualificação.



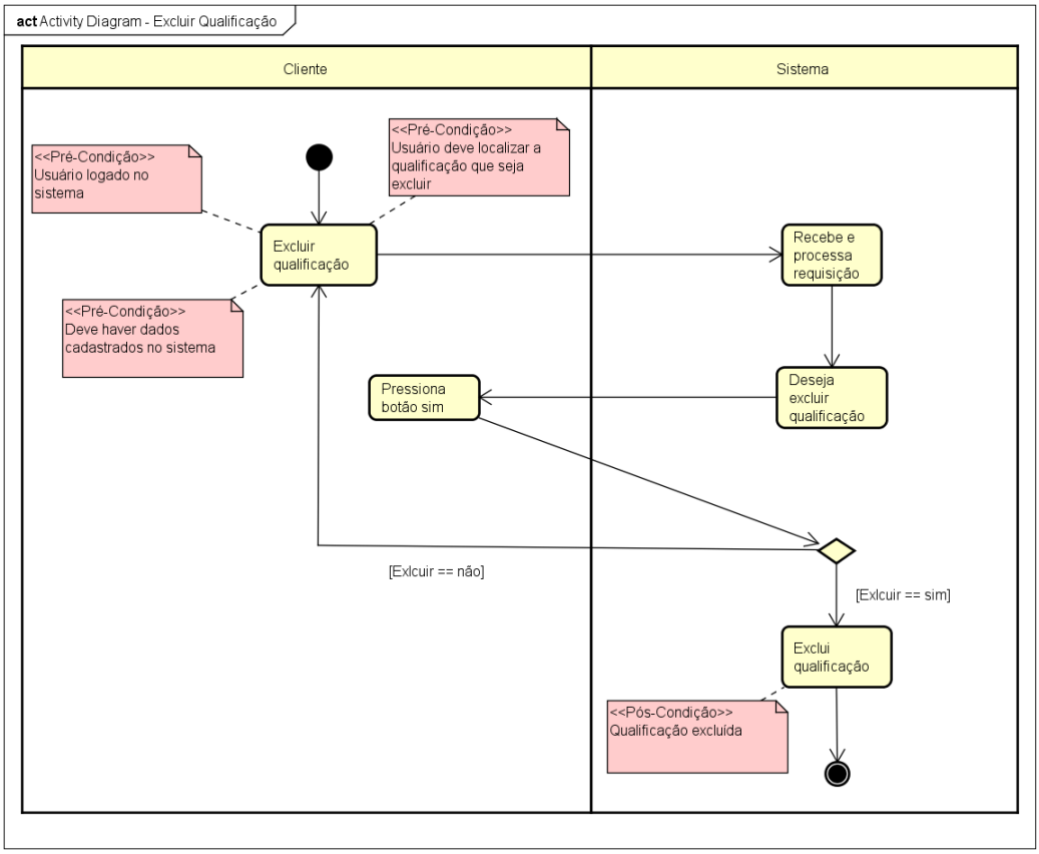
O diagrama de atividades representado na Figura 40 demonstra o fluxo visualizar qualificação.

Figura 40 - Diagrama de atividades, referente ao fluxo visualizar qualificação.



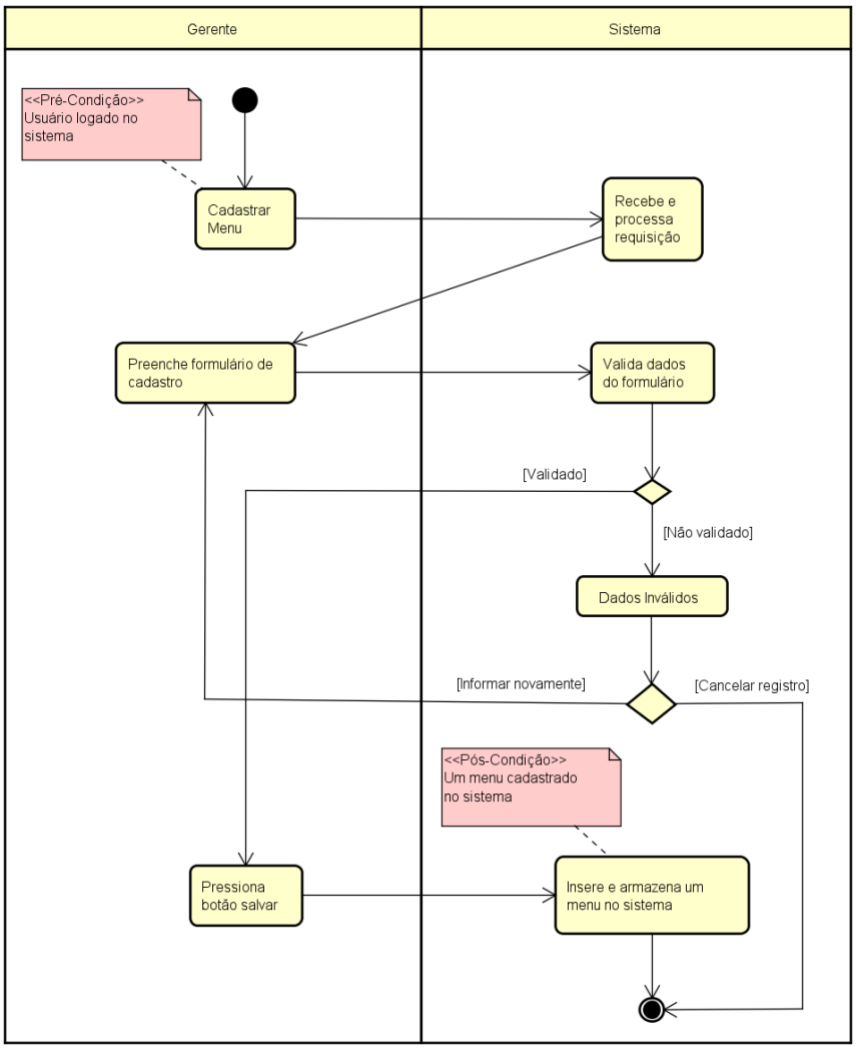
O diagrama de atividades representado na Figura 41 demonstra o fluxo excluir qualificação.

Figura 41 - Diagrama de atividades, referente ao fluxo excluir qualificação.



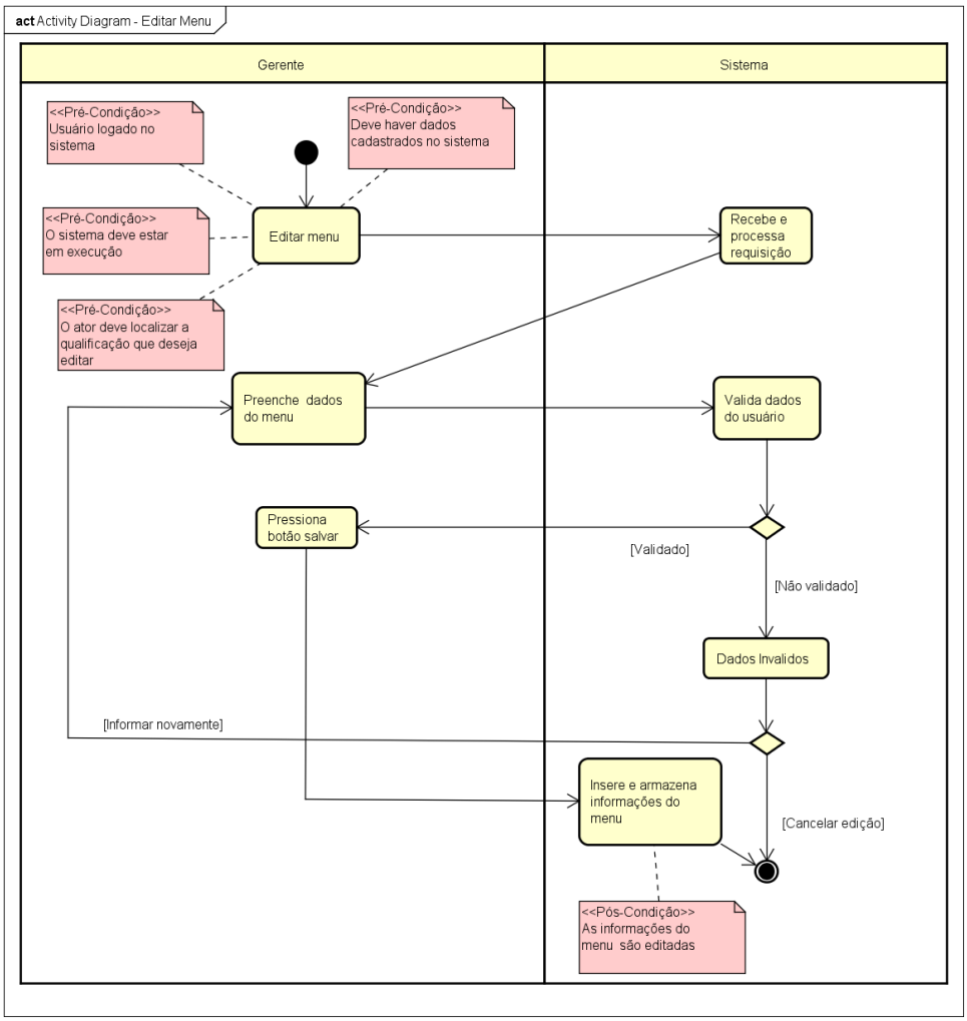
O diagrama de atividades representado na Figura 42 demonstra o fluxo cadastrar menu.

Figura 42 - Diagrama de atividades, referente ao fluxo cadastrar menu.



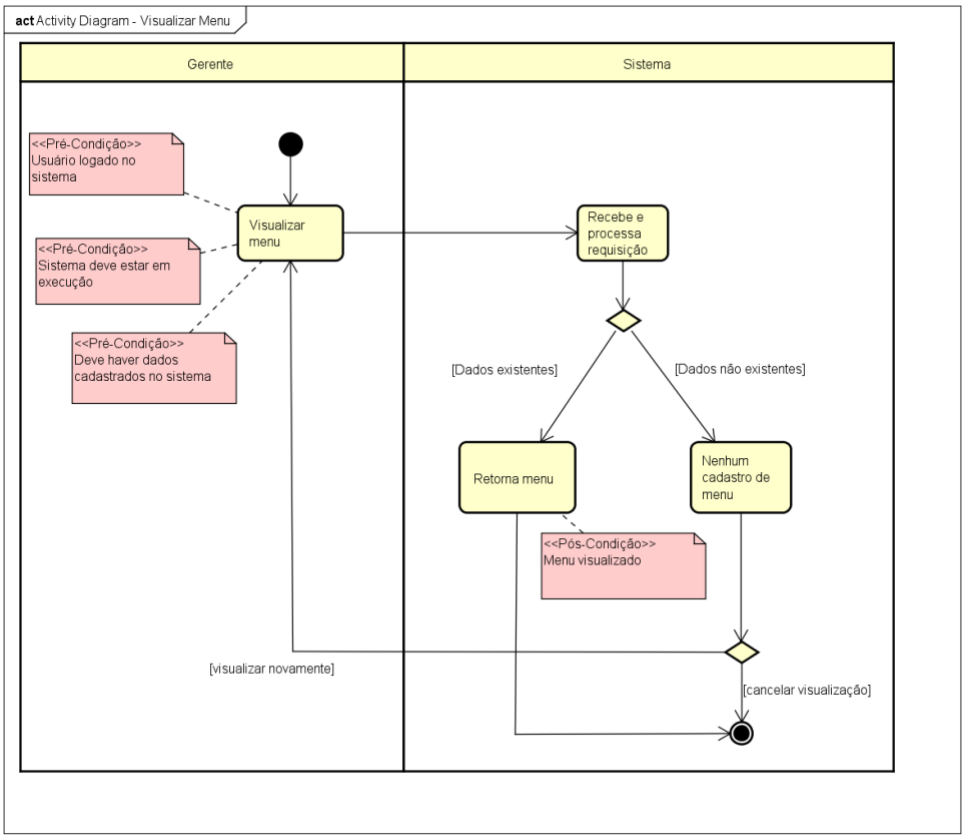
O diagrama de atividades representado na Figura 43 demonstra o fluxo editar menu.

Figura 43 - Diagrama de atividades, referente ao fluxo editar menu.



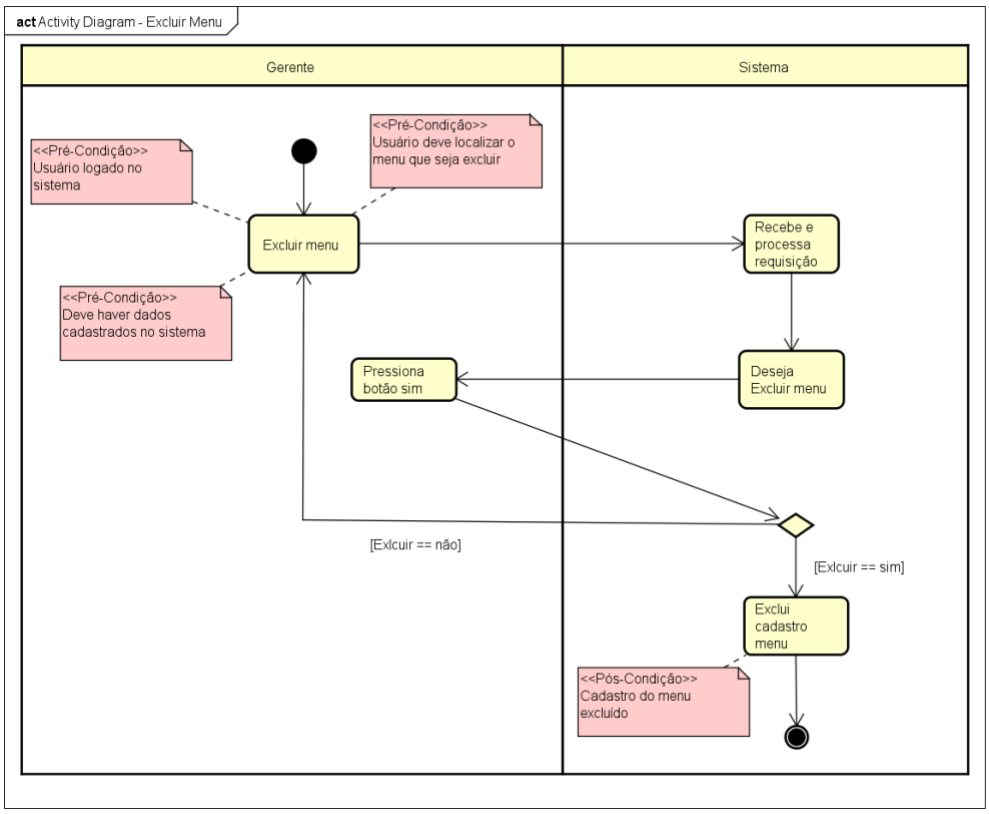
O diagrama de atividades representado na Figura 44 demonstra o fluxo visualizar menu.

Figura 44 - Diagrama de atividades, referente ao fluxo visualizar menu.



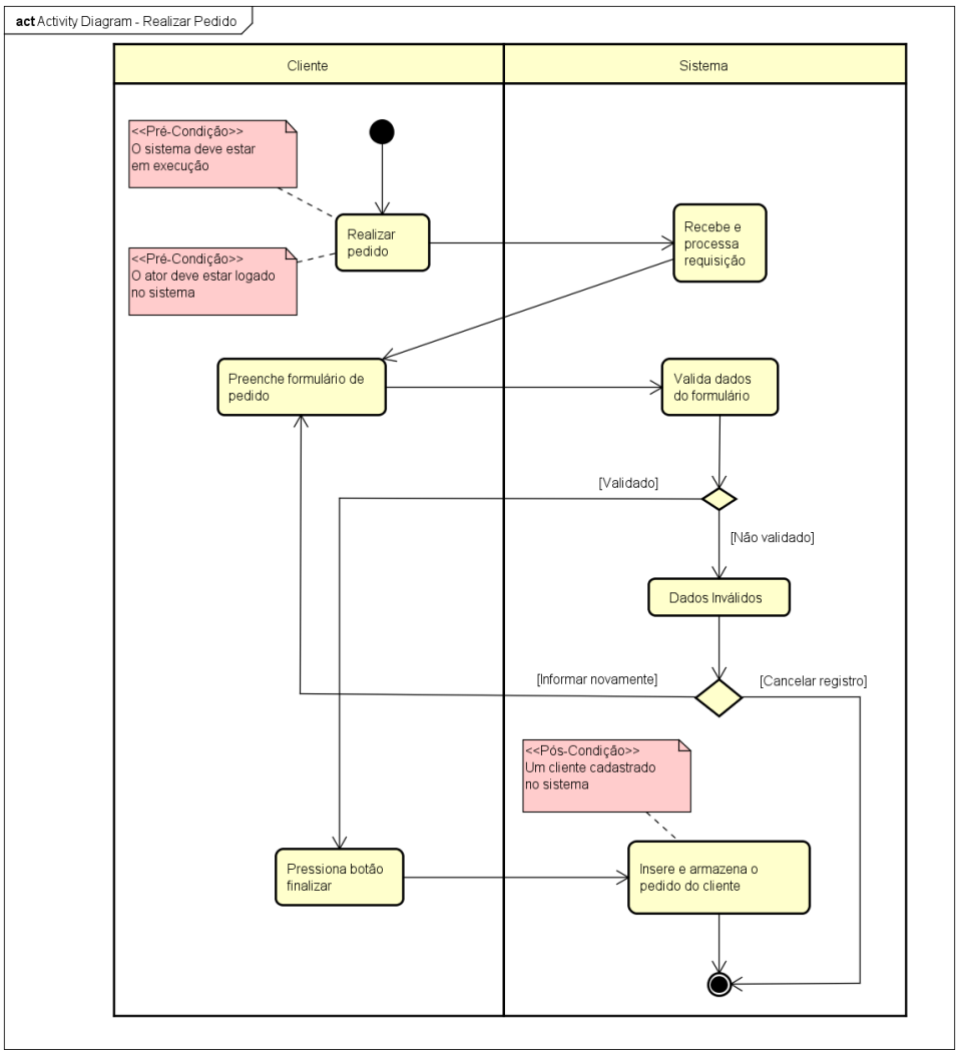
O diagrama de atividades representado na Figura 45 demonstra o fluxo excluir menu.

Figura 45 - Diagrama de atividades, referente ao fluxo excluir menu.



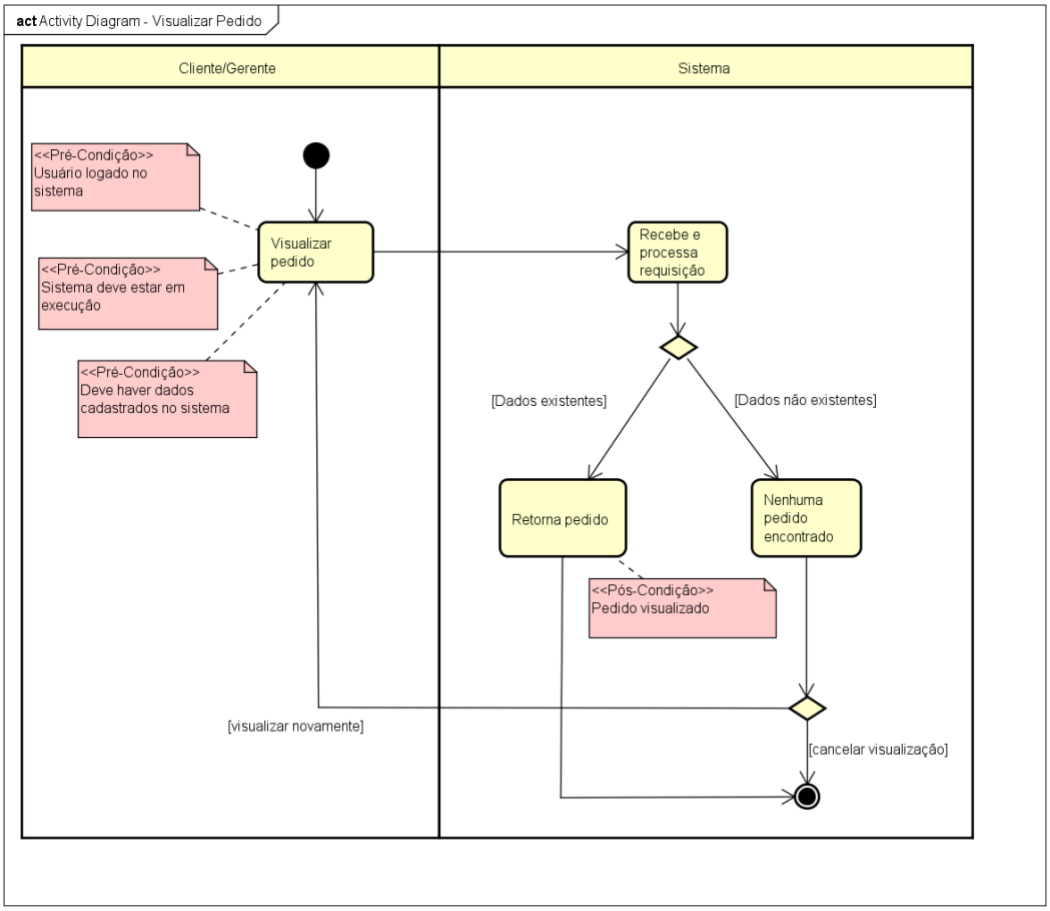
O diagrama de atividades representado na Figura 46 demonstra o fluxo realizar pedido.

Figura 46 - Diagrama de atividades, referente ao fluxo realizar pedido.



O diagrama de atividades representado na Figura 47 demonstra o fluxo visualizar pedido.

Figura 47 - Diagrama de atividades, referente ao fluxo visualizar pedido.



3.6 Implementação

Nesta seção, é demonstrado a implementação das tecnologias e o protótipo do aplicativo.

Tela inicial:

Na figura 48 representa o momento em que o aplicativo é aberto.

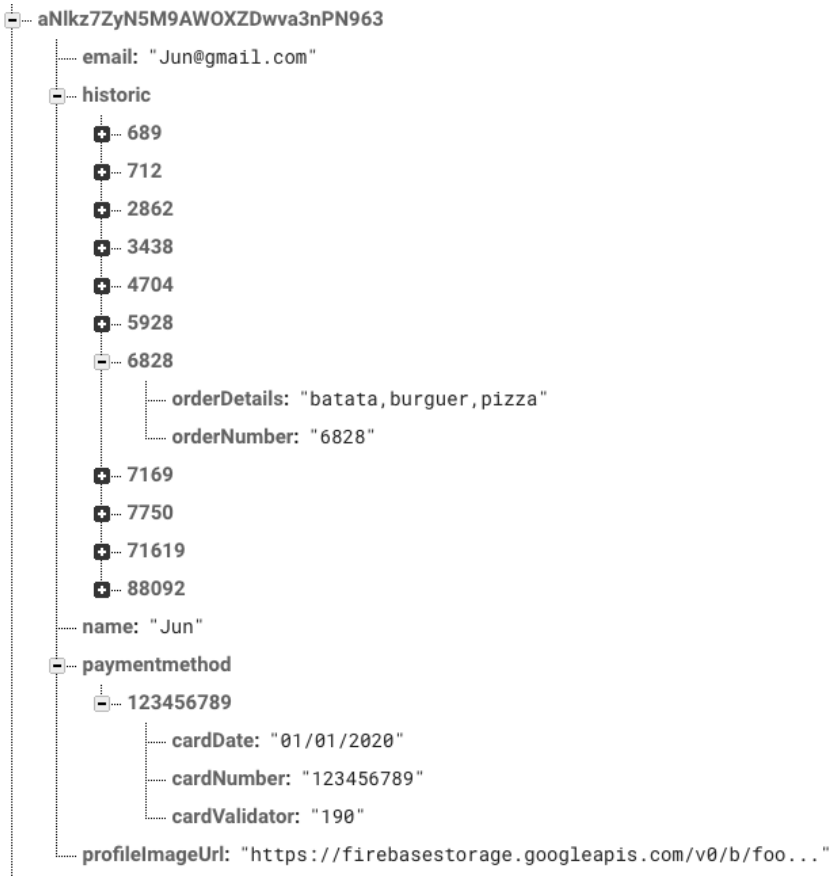
Figura 48 – Tela inicial

Após inicialização, é realizado as seguintes operações:

1. Durante o carregamento do aplicativo, é realizado uma chamada ao Firebase para obter:
   - Dados do usuário;
   - Histórico de pedidos;
   - Cartão de crédito;
   - Dados são armazenados em uma classe Singleton;

Na figura 49 é apresentado um modelo de dados salvos na base de dados do Firebase:

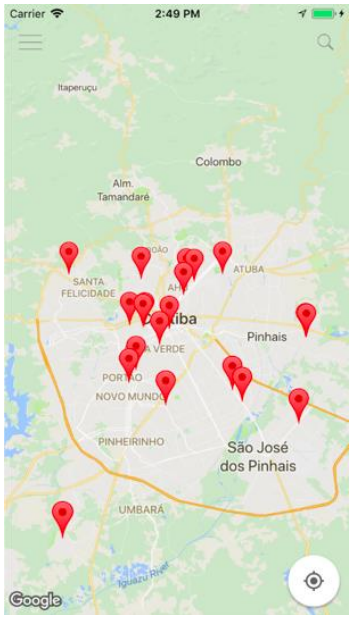
Figura 3 – Console Firebase demosntrando dados de usuário, histórico, método de pagamento e image do perfil



2. Após de realizar a sincronização com o Firebase, a tela é carregada com um mapa da API do Google.
3. Consulta com o WebService GeoLocation para consultar o país de origem e estado do usuário:
   - O JSON retornado pelo GeoLocation é utilizado para limitar a busca por País;
   - O estado é utilizado na consulta no web service Places Details;
4. Com os dados retornados pelo webservice GeoLocation, é realizado uma segunda consulta no WebService Places Details para localizar todos os food trucks da região.
   - Realizado tratamento do retorno JSON e armazendo em um array de objetos FoodTrucks
5. Com o objeto foodTruck carregado, é criado marcadores para cada um no mapa.
6. Ao clicar em um dos marcadores:
   - É salvo o food truck selecionado no Singleton
   - Redirecionado para Tela de Food Truck

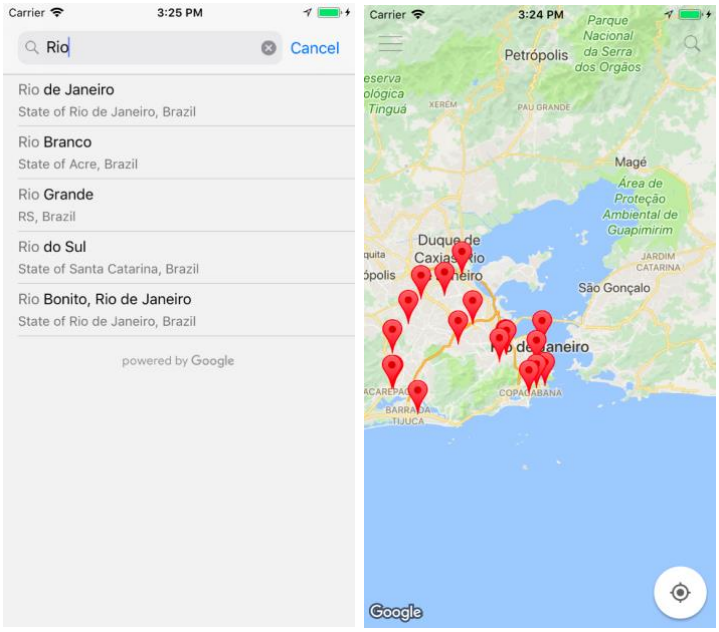
Após finalização do carregamento e sincronização com o Firebase, os dados são exibidos e apresentados na Figura 50, na tela de Localização de food trucks.

Figura 50 – Tela Localização de Food trucks



Ao clicar na lupa na barra de navegação (Figura 50), é apresentado um search bar para consultar food trucks em outras regiões, conforme exibido na Figura 51.

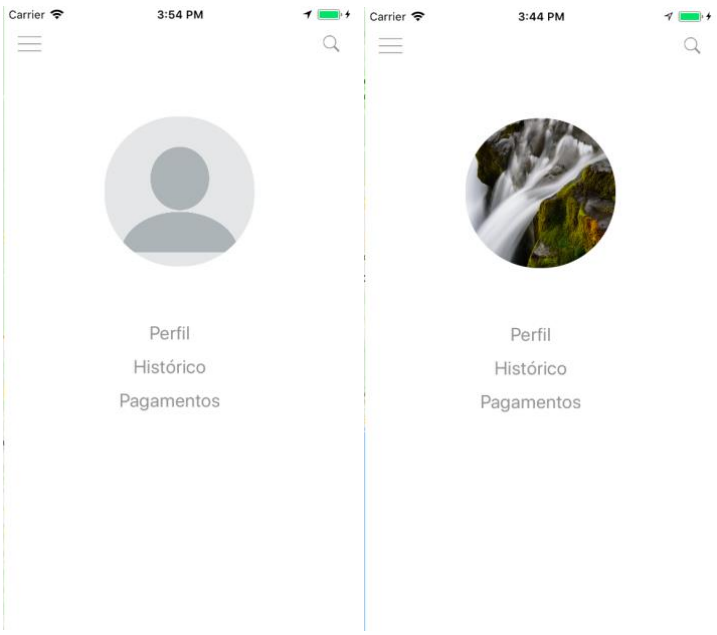
Figura 51 – Exemplo de consulta em outras regiões



A busca é limitada por País, e conforme demonstrado na figura 51, assim que informado a região, é disponibilizada uma lista de cidades de todos os estados do País. Ao selecionar o estado, é realizado uma nova consulta no WebService Place details para obter uma nova lista de food trucks da nova região.

Ao clicar no menu na barra de navegação (Figura 50), é apresentado uma janela de menu, conforme demonstrado na Figura 52:

Figura 52 – Tela de Menu Bar



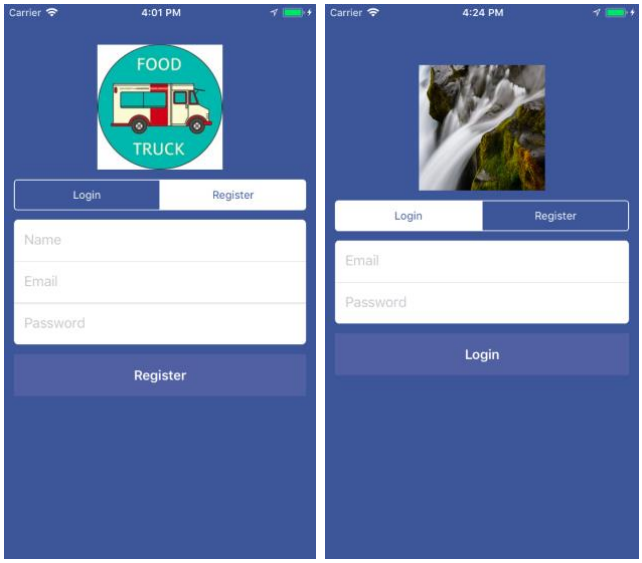
Uma consulta é realizada no singleton para verificar se existe dados do usuário. A imagem do perfil ficará no estado da figura 52 à esquerda quando o usuário não estiver logado, caso o mesmo se encontre logado será apresentado conforme a figura 52 à direita.

Ao clicar na opção Perfil da Figura 52, é realizado uma consulta no firebase para verificar se o usuário está logado:

- Quando um usuário é criado, é gerado um userID único para cada usuário
- Para verificar se o usuário está logado, é realizado uma consulta através do método interno firebase.auth().currentUser.

Caso o usuário não esteja logado, será redirecionado para a tela de Login e registro, conforme Figura 53.

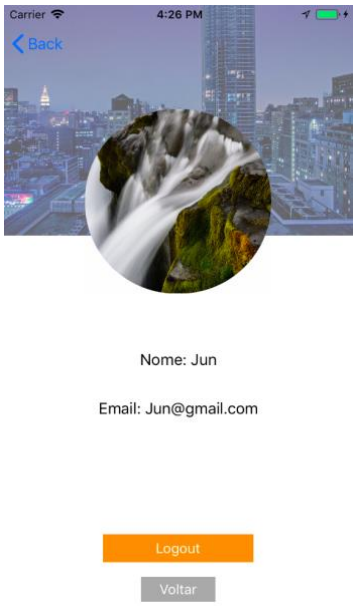
Figura 53 – Tela de Login e Registro



É oferecido duas opções: Registro e Login. Para o Registro, é necessário informar nome, email e uma senha. Também é possível escolher a imagem de perfil ao clicar no ícone de imagem. Ao preencher os dados e escolher a imagem do perfil, as informações são serializadas na base de dados do Firebase e redirecionado para a tela de Perfil, conforme Figura 54.

No Login, com as informações preenchidas, é realizado uma consulta no firebase para verificar se existe um userID que coincidem com o login e senha informado. Todos os dados validados, é redirecionado para a tela de Perfil (Figura 54).

Figura 54 – Tela de Perfil



Supondo que o usuário já estivesse logado, ao entrar na tela de Perfil (Figura 54), seria realizado uma chamada no firebase para obter dados do usuário, histórico e cartão de crédito.

Informações do nome e email do usuário são disponibilizados na tela, juntamente com a imagem do perfil. Ao clicar no botão logout, é desvinculado o userID da aplicação através do método interno do Firebase auth()signOut() e redirecionado para a tela de login (Figura 53).

O Botão Voltar direciona para a tela Principal (Figura 50).

Ao clicar na opção Pagamentos (Figura 52), durante a inicialização da tela de Pagamentos (Figura 55), é realizado uma verificação no singleton se existe um cartão vinculado ao usuário. Caso exista um cartão cadastrado, uma consulta na base de dados do firebase é realizada para carregar as informações do cartão e preencher na tela.

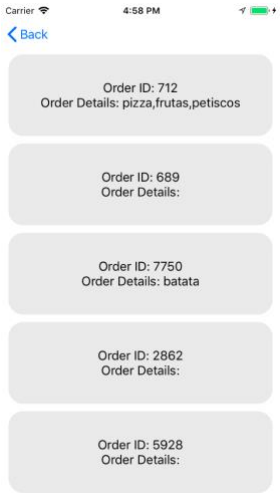
Figura 55 – Tela de Pagamento

Ao pressionar o botão salvar da tela de Pagamento (Figura 55), é verificado no firebase se o número do cartão já existe, impedindo que o mesmo seja cadastrado novamente. O botão remover cartão, deleta da base de dados do firebase as informações do cartão de crédito.

Não havendo um cadastro de cartão de crédito, as informações preenchidas serão cadastradas no firebase, sendo a ID de identificação na base de dados o número do cartão, conforme demonstrado na Figura 49.

Ao clicar na opção Histórico (Figura 52), é apresentado os dados do histórico já carregados previamente no singleton, e a table view é apenas carregada com o array de objetos Históricos do singleton e exibidos na tela, conforme Figura 56.

Figura 56 – Tela de Histórico



Na tela principal (Figura 50), ao clicar em um dos marcadores na tela, é redirecionado para a tela de Tab Bars, sendo a tela principal a descrição do food truck, conforme Figura 57.

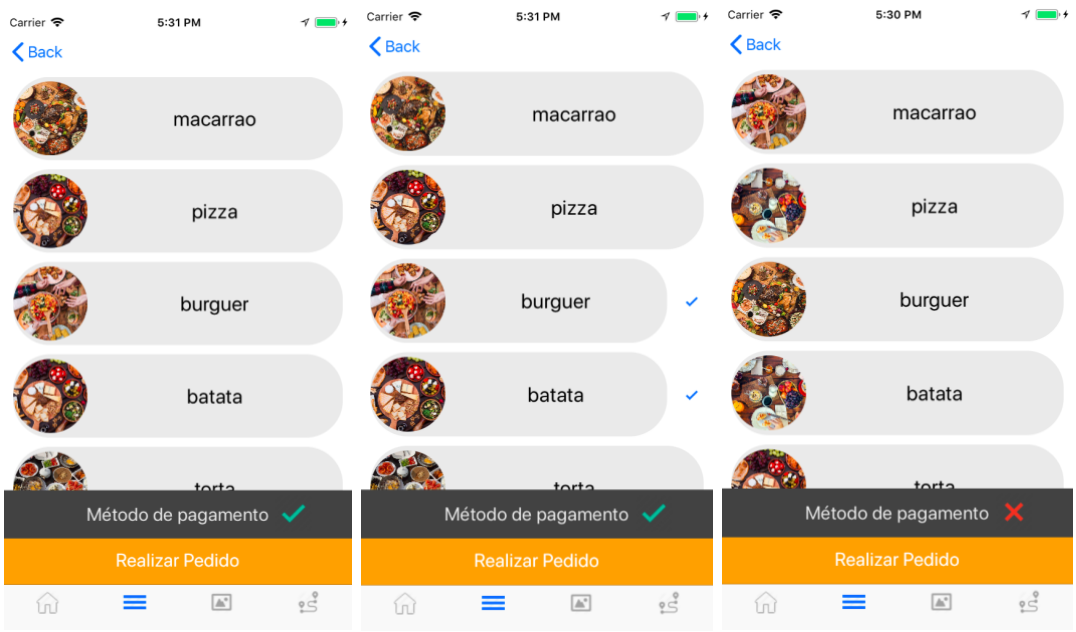
Figura 57 – Tela de Food Truck (descrição)

Durante a chamada da tela de descrição de food truck (Figura 57), é repassada as informações do food truck selecionado na tela principal (através dos marcadores da tela na Figura 50). Com o objeto food truck já populado, é apenas exibido o nome do local, endereço e avaliação do local, conforme demonstrado na Figura 56.

Na segunda Tab bar, é apresentado o Menu cadastrado pelo Food truck. Durante o carregamento da tela é consultado no singleton se o usuário possui ou não um cartão de crédito cadastrado. Será exibido uma imagem de checado (conforme segunda imagem da Figura 58) para cartão de crédito cadastrado e uma imagem em forma de X (terceira imagem da Figura 58). É possível consultar e cadastrar método de pagamento ao clicar no botão, sendo redirecionado à tela de Pagamento (Figura 55).

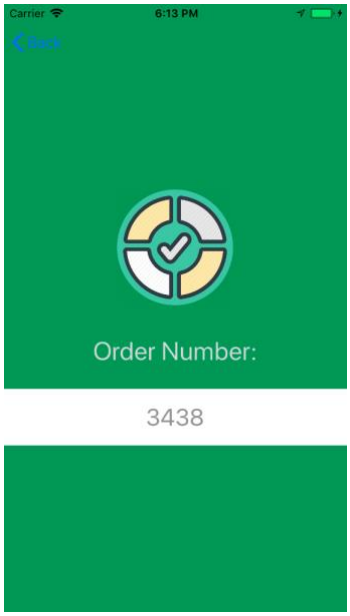
Conforme Figura 58, após selecionar os itens desejados, é possível proceder com o pedido selecionando o botão Realizar Pedido.

Figura 58– Tela de Food Truck (Menu)



Após confirmação de pedido, é gerado um número de pedido que é exibido na tela para o usuário, conforme Figura 59.

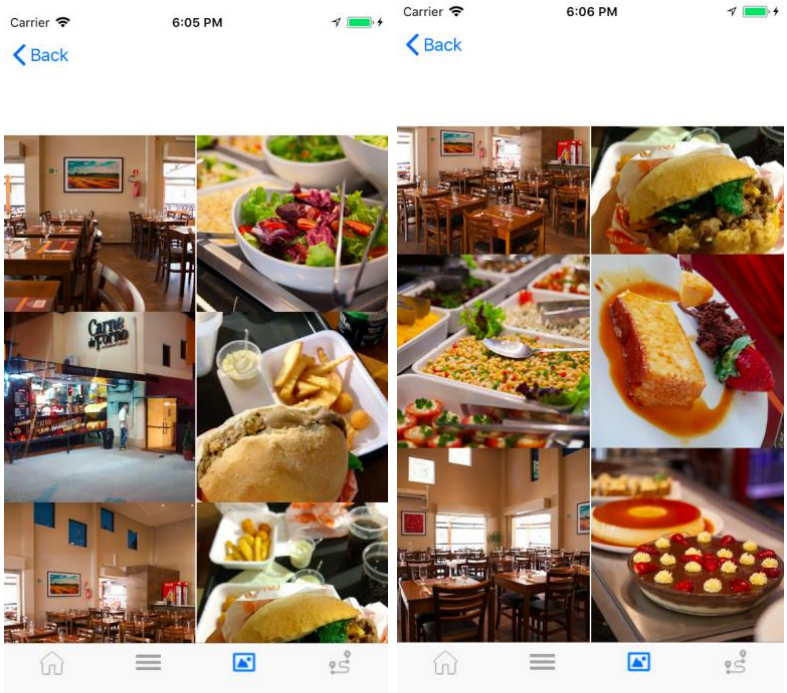
Figura 59 – Tela de Food Truck (confirmação de Pedido)



Com a ordem de pedido gerada, é realizado uma chamada no firebase para armazenar as informações do pedido (número do pedido utilizado como primary key, consultar Figura 49). O detalhe da ordem do pedido poderá ser consultado na tela de históricos (Figura 56).

Na terceira opção do Tab bar, é apresentado a tela de galeria de imagens do Food truck, conforme exibido na Figura 60.

Figura 60 – Tela de Food Truck (Galeria de Imagens)

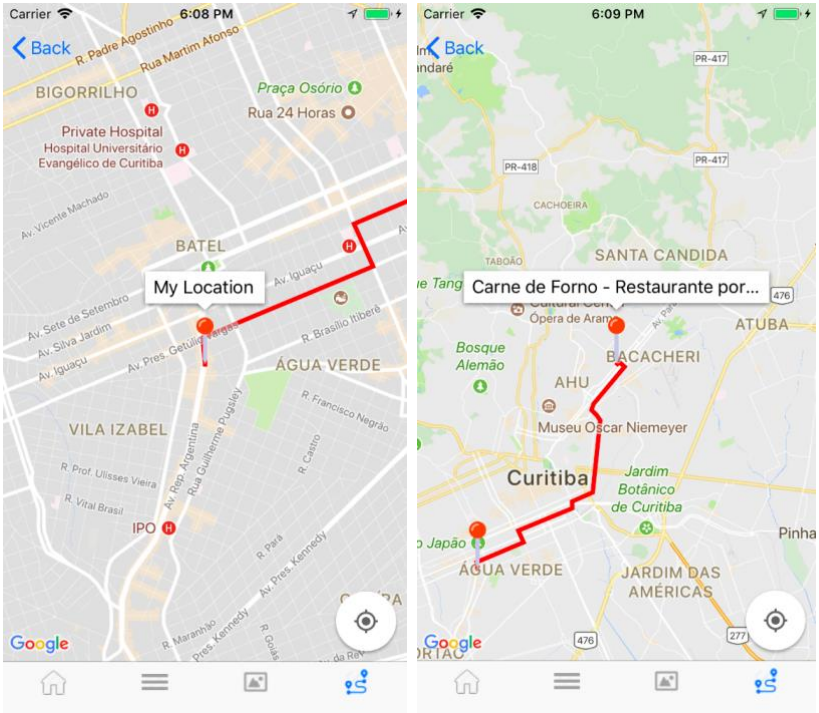


Durante o carregamento da tela de galeria de imagens (Figura 60) é realizado uma consulta pela API do google Places Photos, utilizando como parâmetro o place_id do food truck selecionado. O retorno do serviço é um array de metadados do tipo objeto GMSPlacePhotoMetadata. O array de objetos é repassado para outro serviço google Load Places Photos para converter em objetos UIImages (objeto imagem swift).

Após convertido e armazenado, as imagens são exibidas na collection view da galeria de imagens (Figura 60).

Na última aba do Tab bar, temos a tela de cálculo de rotas até o food truck, conforme Figura 61.

Figura 61 – Tela de Food Truck (Rotas)



Durante carregamento da tela de rotas (Figura 60), são carregados:

- permissões de localização
- obtenção da localização do usuário
- carregado parâmetro foodTruck através do singleton

Após o carregamento, são criado 2 marcadores, sendo eles:

- Localização inicial com latitude e longitude do usuário

- Localização do food truck passando nome do food truck e latitude e longitude do mesmo

Em seguida, ambas localizações são repassadas para um método que define o ponto inicial e ponto final. Com os parâmetros preenchidos, é utilizado o serviço web service google Directions.

O mesmo retorna um JSON com os pontos até o local, sendo traçado no mapa o caminho inicial até o ponto final.

4 Conclusão

Apesar de ainda não finalizado, por se tratar de 2 aplicativos mobile, um voltado para o cliente e outro para o dono do Food truck, o aplicativo já cumpre o objetivo proposto, expondo uma abordagem que visa promover o mercado de food trucks através do mapeamento e disponibilização de serviços que destaquem seu comércio.

Ao realizar o desenvolvimento do aplicativo, um dos pontos que dificultou o foco em UX/UI do projeto, foi a integração de libs não atualizadas para a versão atual da linguagem Swift. Também foi observado dificuldade em integrações que exigem um maior conhecimento da linguagem.

A API do google mostrou-se eficiente na elaboração do projeto, porém, não demonstrou precisão nos resultados na busca por food trucks. Portanto, será considerado em um trabalho futuro a substituição pela API do Foursquare.

4.1 Trabalhos Futuros

Alguns passos se mostram necessários para o avanço e futuro do aplicativo, entre eles:

- Aplicativo Food truck para Owners (donos do food truck) que compõe as seguintes funcionalidades:
    - Cadastro de food trucks
    - Cadastro de menus

Aplicativo Food truck para Clientes:

- Aprimorar UI/UX
- Aprimorar Validações
- Sistemas de notificação entre aplicativos (mensagens de ordens de pedido e conclusão de pedidos)
- Adicionar APIs de pagamentos e implementar no projeto
- Chats entre clientes e owners do food truck (communicação entre Aplicativo food truck com o aplicativo de food truck para Owners)
- Estudar alternativa de utilizar a API do Foursquare para mapeamento dos food trucks

5 Referências

1 ALTEXSOFT. The good and the bad of swift programming language. 2017 Disponível em: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/ > Acesso em: out. 2017

2 APPLE. Programming with Object C. 2014 Disponível em: <https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html> Acesso em: out. 2017

3 APPLE. Apple Releases iOS 8 SDK with over 4.000 new APIs. 2014 Disponível em: <https://www.apple.com/newsroom/2014/06/02Apple-Releases-iOS-8-SDK-With-Over-4-000-New-APIs/> Acesso em: out. 2017

4 APPLE. App Review Guidelines and Resources. 2017 Disponível em: https://developer.apple.com/app-store/review/rejections/

5 APPLE. Human Interface Guidelines. 2017 Disponível em: <https://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html> Acesso em: out. 2017

6 APPLE. App Store Review Guidelines. 2017 Disponível em: <https://developer.apple.com/app-store/review/guidelines/> Acesso em: out. 2017

7 APPSFLYER. The State of in app spending. 2016 Disponível em: <https://www.appsflyer.com/resources/state-app-spending-global-benchmarks-data-study/> Acesso em: out. 2017

8 BENCHMARKGAME. Swift Language vs Java. 2017 Disponível em: <http://benchmarksgame.alioth.debian.org/u64q/swift.html> Acesso em: out. 2017

9 CODEMENTOR. Google Maps API or Leaflet: What`s best for your project? 2016. Disponível em: https://www.techwalla.com/articles/disadvantages-advantages-of-using-the-google-maps-website

10 ECONOMIST. America`s .food-truck industry is growing rapidly despite roadblocks. London, 2017 Disponível em: <https://www.economist.com/blogs/graphicdetail/2017/05/daily-chart-3> Acesso em: out. 2017

11 FASTCOMPANY. Lyft goes swift: How (and why) it rewrote its app from scratch in apples new language. 2015 Disponível em: <https://www.fastcompany.com/3050266/lyft-goes-swift-how-and-why-it-rewrote-its-app-from-scratch-in-apples-new-lang> Acesso em: out. 2017

12 FOLHA. Declínio dos food trucks leva empresário a migrar para loja. São Paulo, 2017 Disponível em: <http://www1.folha.uol.com.br/mercado/2017/07/1903377-declinio-dos-food-trucks-leva-empresario-a-migrar-para-loja.shtml
> Acesso em: out. 2017

13 FOODTRUCKEMPIRE. What is the average income for a food truck vendor?. New York, 2017 Disponível em: <https://foodtruckempire.com/news/survey-income/> Acesso em: out. 2017

14 GENEXUS. How to: Create an .ipa file from XCode. 2016 Disponível em: https://wiki.genexus.com/commwiki/servlet/wiki?34616,HowTo%3A+Create+an+.ipa+file+from+XCode,

15 GOOGLE DEVELOPER. Place Search. 2017 Disponível em: <https://developers.google.com/places/web-service/search> Acesso em: out. 2017

16 GOOGLE DEVELOPER. Place Details. 2017 Disponível em: <https://developers.google.com/places/web-service/details> Acesso em: out. 2017

17 GOOGLE MAPS STYLE. Google Maps style site. 2017 Disponível em: <https://mapstyle.withgoogle.com/> Acesso em: out. 2017

18 HOW TO FIREBASE: What is Firebase? 2017 Disponível em: https://howtofirebase.com/what-is-firebase-fcb8614ba442

19 INFINUM. Android development is 30 percent more expensive than ios. 2017 Disponível em: <https://infinum.co/the-capsized-eight/android-development-is-30-percent-more-expensive-than-ios> Acesso em: out. 2017

20 NONAKA, Ikujiro; TAKEUCHI, Hirotaka. Criação do Conhecimento na Empresa – Como as empresas japonesas geram a dinâmica da inovação. Rio de Janeiro: Campus, 1997.

21 OPEN SOFT. Web service: o que é, como funciona, para que serve? 2017. Disponível em: https://www.opensoft.pt/web-service/

22 PREFEITURA DE CURITIBA. Prefeitura publica guia para proprietários de food truck. 2015 Disponível em: <http://www.curitiba.pr.gov.br/noticias/prefeitura-publica-guia-para-proprietarios-de-food-truck/38247> Acesso em: out. 2017

23 PREFEITURA DE PORTO ALEGRE. Decreto regulamenta funcionamento de food trucks. 2016 Disponível em: <http://www2.portoalegre.rs.gov.br/smic/default.php?p_noticia=190568&DECRETO+REGULAMENTA+FUNCIONAMENTO+DE+FOOD+TRUCKS> Acesso em: out. 2017

24 RAYWENDERLICH. Cocoapods tutorial for swift: Getting Started. 2017. Disponível: https://www.raywenderlich.com/156971/cocoapods-tutorial-swift-getting-started

25 REALM. A database you may not have heard of but app does have - touts cloudy platform. 2016 Disponível em: <https://www.theregister.co.uk/2016/09/29/realm_floats_realtime_database/ > Acesso em: out. 2017

26 SPACEOTECHNOLOGIES. : 6 Reasons Why Startups & Enterprises Choose Swift For Their Custom iOS App Development . 2016 Disponível em: <https://www.spaceotechnologies.com/6-reasons-swift-custom-ios-app-development/> Acesso em: out. 2017

27 STACKOVERFLOW. Developer Survey. 2015 Disponível em: <https://insights.stackoverflow.com/survey/2015> Acesso em: out. 2017

28 TIOBE. The swift programming language. 2017 Disponível em: <https://www.tiobe.com/tiobe-index/swift/> Acesso em: out. 2017

29 THEMINDSTUDIOS. Advantages of using swift over Object-C. 2017 Disponível em: <https://themindstudios.com/blog/advantages-of-using-swift-over-objective-c/ > Acesso em: out. 2017

30 THE VERGE. The swift effect: Apple`s new programming language means way more iPhone developers and Apps. 2014 Disponível em: https://www.theverge.com/apple/2014/6/2/5773928/apple-swift-programming-developers-objective-c

31 TUTSPLUS. How to submit an ios app to the app store. 2017 Disponível em: <https://code.tutsplus.com/tutorials/how-to-submit-an-ios-app-to-the-app-store--mobile-16812> Acesso em: out. 2017