Universidade Estadual de Goiás
Campus de Itaberaí

FERNANDO FERREIRA ALVES
PAULO RODRIGUES VIANA JÚNIOR

ANÁLISE E DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID DE AUTOATENDIMENTO PARA RESTAURANTE.

ITABERAÍ
2015

FERNANDO FERREIRA ALVES
PAULO RODRIGUES VIANA JÚNIOR

ANÁLISE E DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID DE AUTOATENDIMENTO PARA RESTAURANTE.

Trabalho Final de Curso apresentado à Universidade de Goiás, Unidade Universitária de Itaberaí, como requisito parcial para a conclusão do curso de graduação em Sistema de Informação, sob orientação do Professor (a) Thais Peres Montes.

Itaberaí
2015

Dedicamos nosso TCC para todos aqueles que de alguma forma estiveram próximos, proporcionando-nos forças para que não desistíssemos de buscar a realização de nossos objetivos. Valeu a pena os dias árduos de estudo e dedicação, estamos colhendo juntos, frutos de nosso empenho!

Agradecemos aos nossos professores, pela dedicação, paciência e comprometimento. Aos nossos colegas de curso pela companhia diária e compartilhamento de conhecimentos, dentro ou fora da sala de aula. Aos nossos pais e familiares por entender e nos apoiar em momentos difíceis. E principalmente à Deus, que nos deu forças, sabedoria e disposição para nunca desistir dos nossos sonhos.

“A melhor maneira de prever o futuro é inventá-lo.”

Alan Kay, cientista da computação (1971).

RESUMO

A tecnologia mobile está presente em vários setores do mercado, os dispositivos móveis têm ganhando muito espaço, graças ao crescimento e popularidade dessas plataformas. O Android é o sistema operacional mais utilizados entre os usuários de smartphones. Nosso projeto trata-se de uma aplicação Android voltada para autoatendimento em um restaurante especifico. De modo que auxilie o usuário a realizar suas solicitações sem que ocorra nenhum tipo de falha na comunicação, pois assim será eliminado o uso do tradicional "bloquinho de anotação" evitando erros comuns que resultam em demora no atendimento, produtos entregues diferentes do que foram pedidos, filas, uma longa espera do cliente, entre outros problemas. Na implementação e construção do sistema foram utilizadas metodologias através de pesquisas bibliográficas realizadas a partir da coleta de dados provenientes de livros e internet, uso de materiais complementares, estudo de caso para melhor entendimento do ramo de negócio da empresa o qual o aplicativo é voltado. Análise e levantamento de requisitos para ter-se uma total compreensão do funcionamento da empresa. E também a entrevista com o empresário para conhecimento do problema e estudar soluções.

Palavras-chave: Android, autoatendimento, restaurante, aplicativo.

ABSTRACT

The mobile technology is present in various sectors of the market, mobile devices have gained a lot of space, thanks to the growth and popularity of these platforms. Android is the operating system most widely used among smartphone users. Our project it is an Android application aimed at self-service in a specific restaurant. In order to assist the user to make their requests without the occurrence of any failure in communication, thus it will be eliminated the use the traditional "note block" avoiding common mistakes that result in delays in care, products delivered different than request , queues, a long customer expects, among other problems. Implementation and system construction methodologies were used through literature searches performed from the collection of data from books and internet, use of complementary materials, case study to better line of business understanding of the company which the application is turned. Analysis and requirements gathering to take up a full understanding of the company's operation. And also the interview with the entrepreneur to knowledge of the problem and explore solutions.

Keywords: Android, self-service, restaurant, application.

LISTA DE FIGURA

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

IDE - Integrated Development Environment ou Ambiente de Desenvolvimento Integrado.

ADT - Android Developer Tools

SDK - Software Development Kit ou Kit de Desenvolvimento de Software

AVD - Android Virtual Device

I/O - Input/Output

XML - Extensible Markup Language

USB - Universal Serial Bus

POI - Point of Interest

API - Application Programming Interface

OHA - Open Handset Aliance

OMG - Object Management Group

ISO - International Organization for Standardization

UML - Unified Modeling Language

RAD - Rapid Application Development

DBMS - Database Management System

SGBD - Sistema de Gerenciamento de Banco de Dados

E-R - Entidade/Relacionamento

MER - Modelo Entidade/Relacionamento

MRN - Modelo Relacional Normalizado

HTML - HyperText Markup Language

PHP - Hypertext Preprocessor", originalmente Personal Home Page

JSON - JavaScript Object Notation

SQL - Structured Query Language

ISAM - Indexed Sequential Access Method

EUA - Estados Unidos da América

HTC - High Tech Computer Corporation

APK - Android Application Package

WSDL - Web Services Description Language

SOAP - Simple Object Protocol

CASE - Computer-Aided Software Engineering

SUMÁRIO

INTRODUÇÃO

Pelo fato dos avanços da tecnologia, a utilização dos aparelhos celulares atualmente ganhou múltiplas funções, ultrapassando os conceitos de uso unicamente para chamadas telefônicas. Com a criação dos smartphones “telefones inteligentes”, o qual os usuários podem usufruir de diversas funções que até então somente os computadores eram capazes de oferecer, como: leitura de e-mails, media players, navegadores de internet, esses aparelhos começaram a se destacar por seu poder com relação às suas capacidades de armazenamento, de processamento e de comunicação, e por se tornarem mais acessíveis aos consumidores.

Por ser um tipo de produto novo, os aplicativos para dispositivos moveis, possuem um grande mercado de desenvolvimento ainda pouco explorado. Observando esta oportunidade de crescimento, popularização potencial de utilização dos aplicativos e capacidade de venda dos produtos. Buscamos aprender um pouco e dedicamos a desenvolver um aplicativo mobile, previsto para a plataforma Android, a qual possui a maior quantidade de usuários.

Este trabalho apresenta um Aplicativo Android de Autoatendimento para Restaurantes para a implementação na Empresa Cabanas Restaurante e Chopperia localizada em Itapuranga- Go. O aplicativo está sendo desenvolvido para garantir que o usuário possa aproveitar todas as funcionalidades que ele proporciona, acomodando maior conectividade, mobilidade e portabilidade de seu uso para fazer pedidos sem dependências de garçons, usando seu próprio smartphone para visualizar, escolher e finalizar seu pedido. Este processo é um meio de otimizar o tempo, diminuir erros, proporcionar interação do cliente e assim ser uma solução que atenda totalmente o que o usuário espera obter.

Além de pedidos, o aplicativo conta com um menu para que o usuário possa reservar mesas, visualizar shows e eventos, poder gerenciar sua conta e pedidos tudo de forma simples e automatizada na palma de sua mão através do smartphone, trata-se de uma maneira de atrair mais clientes oferecendo uma inovação tecnológica para marketing promocional e satisfação do cliente de forma mais ágil.

Para a implementação e construção do sistema foram utilizadas metodologias através de pesquisas bibliográficas realizadas a partir da coleta de dados provenientes de livros e internet, uso de materiais complementares, estudo de caso para melhor entendimento do ramo de negócio da empresa o qual o aplicativo é voltado. Análise e levantamento de requisitos

para ter-se uma total compreensão do funcionamento da empresa. Foi utilizado também entrevista com o empresário para conhecimento do problema, estudar soluções para esse problema e levantamento das necessidades que o nosso sistema deverá atender.

O embasamento teórico do projeto foi feito utilizado alguns livros como referências para auxiliar na construção e fundamento do sistema. Os Autores Referenciados são: Sommerville, Pressman, Guedes, Lecheta, Ogliari, Silberschatz, Date e Milani. Foram utilizados tais autores com a finalidade de se conhecer melhores técnicas para modelagem do sistema, respeitando os padrões propostos e assim montar a documentação do sistema com o uso de listas de requisitos, diagramas da UML e relacionamento com banco de dados. Foi indispensável o uso da Prototipação que nos ajudou no entendimento dos requisitos além de tornar visíveis os conceitos e funcionalidades do aplicativo, aumentando nossa percepção para validar a ideia base e poder visualizar o desenvolvimento por meio gráfico.

Findando esse assunto, o trabalho está divido em 7 capítulos: 1º – Android; 2º - Engenharia de Requisitos; 3º- Engenharia De Software; 4º- Uml; 5º- Sistema De Gerenciamento De Banco De Dados; 6º- Estudo De Caso; 7º- Implementação. E, por fim, estão as Considerações Finais, as Referências Bibliográficas, os Apêndices e Anexos.

1 ANDROID

Segundo Ogliari (2014), tudo começou quando uma pequena empresa chamada Android Inc, de Palo Alto (Califórnia – EUA), construiu uma plataforma para desenvolvimento e execução de programas para dispositivos móveis com facilidade de aprendizagem e utilização. Em 2005 passou a ser da Google.

Lecheta (2015), diz que, comparado a outras plataformas, o Android é muito simples, pois utiliza a programação Java, e interface visual, que facilita o processo de desenvolvimento. Android é um projeto de código aberto, então surgirá versões a todo momento.

Os smartphones, como são chamados, resumem-se a celulares com maior desempenho, processadores de alta capacidade, conectividade com a Internet, telas sensíveis ao toque, sensores, câmeras, etc. Com todos esses recursos, se tornaram a grande aposta do Android.

> "O Android é a primeira plataforma para aplicações móveis completamente livre e de código aberto (open source), o que representa uma grande vantagem competitiva para sua evolução, uma vez que diversas empresas e desenvolvedores do mundo podem contribuir para melhorar a plataforma." (LECHETA, 2015)

Android ainda conta com uma máquina virtual chamada Dalvik, que otimiza a execução de aplicações, memória e os recursos de hardware. Desta forma, aplicações Android recebem a extensão. dex, ou seja, Dalvik Executable.

"Hoje, é de responsabilidade da Open Handset Alliance (OHA), uma aliança fundada em 2007, onde além da Google, outras empresas do setor como, Intel, Acer, Motorola, Asus, DoCoMo, HTC, Huawei, Sprint, Kyocera, T-Mobile, LG, Samsung, Vodafone, SonyEricsson, Qualcomm e NVidia se uniram para fomentar o crescimento da plataforma Andoid."

Ogliari e Brito (2014), diz que várias empresas se juntaram e formaram uma aliança para melhorar a plataforma Android. Esse acordo faria com que todas essas empresas pudessem sair ganhando.

Depois de fundar essa aliança, foi lançado o primeiro dispositivo móvel Android Enabled, chamado HTC T-Mobile, em 2008. Em seguida foram lançados vários aparelhos com essa plataforma, até mesmo automóveis com painéis inteligentes.

Várias vantagens tornaram do Android uma grande potência, entre elas destacam-se seu designe sofisticado e até mesmo o fato de ser criada e associada ao Google, isso impulsionou seu avanço no mercado.

Na infraestrutura de software, o sistema operacional é baseado no Linux, tem conjunto de bibliotecas, uma API chamada Android Runtime, e diversas aplicações. Este sistema operacional é responsável pelo acesso à rede, gerenciamento de memória e processos e fazer comunicação entre o dispositivo e o programa que está sendo desenvolvido.

O Android tem uma característica, que os aplicativos nativos não têm prioridade nos que forem instalados por terceiros, como era no caso da Java Micro Edition. Além disso qualquer aplicativo instalado, pode ter acesso a ferramentas e funções do sistema.

O ambiente de desenvolvimento Android é composto por duas IDEs, que são elas: NetBeans e Eclipse. Mas a Google já anunciou sua nova IDE, o IntelliJ IDEA com base para Android Studio.

NetBeans é destacada por ser de grande facilidade e usabilidade, também foi a primeira a implementar ambientes visuais de desenvolvimento mobile.

Já o Eclipse tem como principal vantagem a facilidade para gerenciar plug-ins, ou seja, seu ambiente terá mais funcionalidades, e também possui um ótimo editor de código. Por isso Google e o principal site de desenvolvedores apoiam essa IDE.

A configuração do ambiente de desenvolvimento era difícil e sujeito a falhas, mas daí surge o ADT Bundle, que é um pacote com estrutura de pastas pronta para usar, com Android SDK e Eclipse junto.

Depois de instalar, é só executar o arquivo SDKManager.exe, que irá abrir o aplicativo de gerenciamento. Nesse aplicativo que serão escolhidos os pacotes para o desenvolvimento. Cada pacote tem sua função, como por exemplo o pacote tools, que traz ferramentas que permitem fazer testes com o emulador de dispositivo.

Com o emulador, Android Virtual Device (AVD) é possível simular um dispositivo Android real, onde é definido o tipo do teclado, a memória, aparência, etc.

Após o término do ambiente de desenvolvimento Android, pode-se criar o primeiro aplicativo.

O primeiro passo para iniciar o aplicativo, é a criação de um projeto. Com a IDE aberta, acesse o menu na barra Arquivo-Novo-Projeto de Aplicativo Android. Abrirá uma tela em que serão informados os dados do novo projeto.

Estes dados são Aplicação Name, ou seja, o nome apresentável do projeto, que será o nome que abaixo do ícone do aplicativo. Project Name, que é o nome físico do projeto, para que possa ser armazenado. Package Name, é o pacote onde será armazenada a aplicação. Minimum Required SDK, para informar qual a versão mínima para que o dispositivo Android possa rodar a aplicação.Target SDK, para informar a versão ideal do Android.Compile With, informa qual versão do Android irá compilar o projeto. Theme, onde será escolhido o tema, se

a versão for moderna.

Depois será criado o ícone do projeto (Activity), classe que uma tela de aplicação Android. Esse ícone é feito utilizando uma imagem, ou mesmo um texto para representação visual do aplicativo.

Na próxima tela é defino o template para o Activity, que tem opções como tela inteira, todo branco, e outro com padrão Master/Detail.

Na última tela são informados três campos: o nome da classe Activity, o nome do arquivo XML, e o tipo de navegação entre elas.

Ao apertar o botão finish, será finalizado o wizard, e o projeto estará criado.

O arquivo XML representa a tela de uma aplicação Android, em que seus códigos a tag RelativeLayout define qual será a altura e largura da tela. Também existe a tag TextView que são atribuídas largura e altura do componente de texto. Porém este código XML possui somente a interface gráfica da aplicação, então será necessário um arquivo activity, que tenha a extensão PrincipalActivity.java por exemplo.

Para criar ótimos aplicativos, é preciso entender o ciclo de vida de um Activity. Em primeiro lugar é a representação de uma tela, que é apresentada ao usuário. Quando a tela é chamada, começa então o ciclo, que é onCreate, onStart e onResume. Neste terceiro método o usuário já está com a tela disponível.

A partir daí a tela pode ficar em segundo plano, com a chamada do onPause. Caso o Activity perca seu foco, onStop será chamado depois do onPause. Se a tela encontrar-se em onStop, há uma chance de que o sistema operacional do Android possa destruí-la, caso esteja com pouca memória por exemplo. Porém, existe um método chamado onDestroy que faz este papel, que destrói a tela e sai da memória.

Sem modificar os arquivos .xml e .java, basta executar o aplicativo, na opção Run As – Android Aplication. Esse passo pode levar alguns minutos dependendo da máquina que se está programando, então existe um caminho ais rápido, que é executar via cabo USB diretamente do dispositivo Android. Depois que carregar, aparecerá na tela do AVD, sua aplicação.

Mas não é somente este modo que permite criar interfaces gráficas. O Android também oferece outro modo em que o desenho da interface é feito quando está em fase de codificação, ou seja, dentro do próprio aplicativo.

No Android também existe uma grande diferença entre os componentes de interface gráfica, que são View e ViewGroup.

Na classe View, tem a área na tela em que se desenha e controla os eventos do

componente, ou seja, botões, caixas de texto etc. Já a classe ViewGroup, tem componentes que gerenciam o layout, como se fosse um contêiner invisível que recebem os Views.

O posicionamento dos componentes visuais da tela é feito por um gerenciador de layout, onde podem ser alteradas altura e largura. Os principais gerenciadores são: LinerarLayout, AbsoluteLayout, TableLayout, RelativeLayout, FrameLayout, ScroolLAyout, GridLayout.

No gerenciador LinearLayout os componentes são mostrados na tela em um formato linear, sendo alinhados horizontalmente ou verticalmente.

No AbsoluteLayout as posições dos componentes são definidas com base em coordenadas x e y. A vantagem dele é que se pode escolher a posição exata do componente.

No TableLayout os componentes são organizados em formata de linhas e colunas, bom para apresentar formulários de cadastro.

No RelativeLayout permite-se que adicione componentes na tela, levando em conta a posição do outro componente adicionado ou do gerenciador que ele se encontra.

No FrameLayout dá para reservar um espaço na tela para adicionar outros componentes.

ScroolView ou ScroolLayout serve para fazer barras de rolagens quando os componentes não cabem na tela.

DridLayout é quase igual ao TableLayout, a única diferença é que nele pode definir células vazias ou definir quantas linhas ela ocupará.

Um ponto bem interessante no Android é que estes gerenciadores podem se juntar, ou seja, em uma situação que o programador precise fazer uma tela de cadastro, ele utilizará o TableLayout. E nessa tela haverá vários campos. Então entra o ScroolView para que exista uma barra de rolagem. De fato, ficaria bem mais interessante.

Os principais componentes visuais do Android, chamados de view, estão definidos como tags no arquivo XML. Eles são divididos em grupos: Views básicas que são caixas de textos e botões. Pickers Views, que permitem a seleção de dados. Views de listas, que mostram a lista de informações. Views para imagens, que são os componentes que apresentam as imagens. Menus, que são as opções de menu. E extras que são aqueles componentes especiais com suas devidas funções.

TextView é um componente de texto, que tem características como android:id que define o nome do componente visual, android:text que informa o texto inicial, android:layout_width que define a largura, e android:layout_heigth que define altura.

Button é um componente que representa um botão visual na tela do dispositivo móvel.

ImageButton é um componente igual ao Button, mas este botão não é do dispositivo, mas sim do conteúdo que o usuário está usando. EditText é um componente que apresenta uma caixa de texto, em que são informados os dados pelo usuário. CheckBox é um componente utilizado para ligar ou desligar alguma opção. RadioGroup e RadioButton são componentes que permitem ao usuário marcar duas ou mais opções de seleção. ToogleButton é um componente de básico que pode estar ligado ou desligado, dependendo da opção que o usuário marque, sendo sim para o botão acionado e não para o botão não acionado.

Para fazer um tratamento de evento com usuários, o Android utiliza sete formas diferentes:

Clique, ou seja, quando o usuário aperta uma tecla por exemplo, ou por meio do touch feito com um clique na tela.

Clique Longo, quando o usuário clica e mantém por alguns segundos.

Menu de contexto, pode se criar um menu para cada view.

Evento de toque, quando o usuário toca algum componente ele é chamado, sabendo então qual direção está sendo apontada

Mudança de foco, quando o usuário faz interação, pode haver mudança de foco.

Evento de tecla acontece quando o usuário quer mudar um componente de uma caixa de texto, por exemplo. Chamado evento onKey.

Item selecionado acontece quando o usuário seleciona um item, onde este item pode estar em um Spinner, por exemplo.

Para quem está acostumado com limitações de interface gráfica, com certeza, não conhecem a plataforma Android. Pois não só tem os componentes visuais básicos, como também componentes avançados como o AutoCompleteTextView, ProgressBar Spinner, PickerViews e ListViews.

O componente AutoCompleteTextView é como se fosse o filho do TextView, pois ele traz um campo onde o usuário insere as informações. Mas o grande diferencial dele é que quando o usuário começa a inserir dados há um auto complemento, assim facilitando para o usuário digitar.

O componente ProgressBar é uma barra de progresso, muito utilizada para que o usuário possa acompanhar o progresso de suas tarefas, como um download, por exemplo.

O componente Spinner permite que uma grande quantidade de dados seja apresentada em uma lista.

O componente PickerViews permite que o usuário possa selecionar uma hora ou data, através do calendário.

O componente ListViews permite que uma grande quantidade de dados sejam apresentados em formato de lista, com o diferencial de usar os eventos de cliques e seleção.

Google Maps e Location API serão utilizados para geolocalização. São ferramentas de grande utilidade para qualquer serviço. Através delas será possível mostrar dados dos pontos da cidade e também um POI (Point of Interest) que identifica onde a pessoa está.

Um POI é inserido no mapa com a classe MarkerOptions, que possui vários métodos para customizar, como título, subtítulo e ícones. Para adicionar um POI ao mapa é necessária uma instância de GoogleMap.

Para acompanhar as mudanças de posicionamento do usuário, o desenvolvedor não precisa mais trabalhar na lógica, pois a nova Location Service API faz tudo, o aplicativo somente precisa solicitar a posição geográfica quando necessitar. Mas a exatidão depende da capacidade do hardware que o smartphone possui e das permissões de usuário para marcação.

Essas permissões podem ser duas: ACCESS_COARSE_LOCATION e ACCESS_FINE_LOCATION.

Existem diversas formas de utilização das notificações no Android, desde as mais simples, como texto, até as mais complexas, como imagens e textos divididos em linhas.

O projeto tem uma tela contendo opções que permitem a criação de notificações, entre elas existem:

Normal View e Big View, onde o normal view tem suas posições restringidas e o big view são diversas possibilidades.

Estilos de Big View, onde o usuário pode escolher entre opções de Big Picture, Big Text ou Inbox.

Progress Bar, onde o usuário escolhe se a notificação aparece na barra de progresso.

Context text e Context Title, textos que são visualizados na notificação.

Conjunto de imagens, onde o usuário escolhe qual imagem será marcada como ícone da notificação.

Depois de escolher a opção, a notificação ficará visível por alguns segundos na parte superior do smartphone do Android, e se quiser, o usuário poderá visualizar a área de serviços e esconder a mesma.

A função da notificação é chamar outra tela quando o usuário clicar nela, e que será redirecionado para a tela ResultActivity. Para remover a notificação deve chamar o método setAutoCancel e passar true como parâmetro. Assim quando o usuário clicar nela, ela se autodestruirá.

“Em um mundo abarrotado de aplicativo nas lojas virtuais, devemos fazer com que

nossas soluções sejam mais atraentes e mais inteligentes que nossos concorrentes. E não precisamos sofrer na programação para conseguir tal feito.” (OGLIARI; BRITO, 2014)

Ogliari e Brito (2014), descreve o que vários programadores tentam conseguir, entretanto a maioria esquece de coisas que o usuário mais quer, que é interação com facilidade. Então quando deixa de lado a interface intuitiva, deixa também de agradar o gosto do usuário, perdendo competitividade no mercado. Não adianta o grau de complexidade de seu projeto, se ele não atende as necessidades dos usuários, busque a inteligência juntamente com a aparência atraente que terá sucesso.

Depois de criar um aplicativo, é hora de publicar. Mas antes de publicar na Google Play, precisa definir a versão do aplicativo no arquivo app/build.gradle, configurando os itens versionCode e versionName. O iten versionCode é usado para instalar ou atualizar um aplicativo pela loja. E o itens versionName mostra para o usuário em qual versão ele está.

1.1 Banco de Dados

1.1.1 SQLite

Segundo Lecheta (2015), o SQLite é um poderoso banco de dados nativo da própria plataforma android, cada aplicação pode ter um ou mais banco de dados, ele pode ser criado de diversas maneiras: Utilizando a própria API do Android, usando um cliente do SQLite como o SQLite Expert Personal, que está disponível para download gratuitamente, ou pode usar o aplicativo sqlite3 pelo console do emulador.

“O Android tem suporte ao SQLite (http://www.sqlite.org), um leve e poderoso banco de dados. Cada aplicação pode criar um ou mais banco de dados que ficam localizados na pasta /data/data/pacote.do.aplicativo/databases/” (LECHETA, 2015)

Utilizando a API é a melhor maneira e deve escrever os scripts SQL para criar as tabelas, ou pode-se criar usando uma ferramenta externa e depois importar o banco para o projeto, é necessário algum código para copiar o arquivo do banco para dentro das pastas do aplicativo.

O banco de dados SQLite possui todas características de um banco de dados normal, é possível gerencia-lo fazendo inserção, atualização e exclusão de registros no banco de dados, além de fazer qualquer tipo de busca de informações no banco de dados

1.1.2 Web Service

Para quem está iniciando no desenvolvimento mobile fica em dúvida como fazer para conectar sua aplicação ao banco de dados do servidor e buscar ou atualizar informações nesse banco de dados. A melhor resposta para isso é que o aplicativo não deve conectar-se ao banco de dados do servidor diretamente, por questões de segurança e por padronização de acesso, para facilitar esses acessos é usado os web services, pois não importa a linguagem de programação usada no servidor ou cliente.

Para aplicações mobile usar informações do servidor é usado o web service que fica hospedado no servidor web, ele pode ser criado em várias linguagens de programação (Java, .NET, PHP, Ruby, Python).

A principal função do web service é acessar o banco de dados, processar as informações e retornar os dados no formato XML ou JSON para o cliente ser possível ler os dados. Há várias maneiras para criar web service as principais serão as seguintes:

1. WSDL (Web Services Description Language) – é um serviço escrito em XML, e as requisições são feitas via HTTP com o protocolo SOAP (Simple Object Protocol). Esta maneira não é adequada para mobile, pois, o protocolo SOAP é um XML que passa pela rede e causa lentidão no uso de internet 3G
2. REST – criado sobre o protocolo HTTP, tem formato leve e pode ser utilizado os métodos GET, DELETE, POST e PUT para padronizar os acessos, o formato mais comum de retorno do REST é o JSON.
3. Páginas básicas com GET ou POST – é uma forma simples de usar web service, pois é criada uma página web que pode receber requisição por GET ou POST e retornar os dados em XML ou JSON.

As requisições HTTP com GET ou POST devem ser usadas na maioria dos web service, no uso de uma requisição do tipo GET no servidor passa os parâmetros na URL e é utilizada para a consulta de dados e retorno de dados geralmente em formato XML ou JSON. Para requisições do tipo POST envia os parâmetros no corpo da requisição HTTP.

A vantagem de usar o método POST é a segurança pois pode mandar mensagens confidenciais, como senhas serão enviadas no corpo da requisição enquanto se fosse pelo GET ficaria visível na URL junto com os parâmetros.

> “Traçando um paralelo mais uma vez, quando você faz login no Gmail ou em qualquer formulário de cadastro na internet, geralmente é feita uma requisição do tipo POST no site. A requisição do tipo GET também tem um limite em relação a quantidade de parâmetros que podem ser enviados, portanto o POST é recomendado para trafegar muitas informações”. (LECHETA, 2015).

1.1.3 JSON

Lecheta (2015), tem a seguinte definição sobre JSON (JavaScript Object Notation), “é uma estrutura de dados que representa um objeto em JavaScript. ”

É um modelo para armazenamento e transmissão de informações no formato texto, estrutura informações de forma mais compacta que o XML, JSON passou a ser mais utilizado do que o XML, por ser mais leve e ter um formato mais simples.

JSON se divide em duas estruturas:

- Coleção de valores/nomes, chamados de object.
- Lista de valores ordenados, também conhecidos como array, vetor, ou lista.

Comparando JSON e XML, percebemos que ambos têm informações no formato de texto, e que são independentes de linguagem, ou seja, quando acessados por API’s específicas, podem ser reconhecidos em qualquer linguagem de programação.

1.2 Google Play

“Ao publicar aplicativos no Google Play, talvez algo que você tenha interesse em aprender é como colocar os famosos anúncios no aplicativo, afim de tirar um lucro com o aplicativo, mesmo se ele for gratuito.” (LECHETA, 2015)

Lecheta (2015), recomenda aprofundar o conhecimento em anúncios, através de vídeos de palestras feitas no Google I/O. Nelas estão diversas formas de rentabilizar a sua aplicação.

No arquivo app/build.gradle tem que ser feita a configuração de compatibilidade da aplicação com a versão do sistema do dispositivo. Então deve configurar o atributo minSdkVersion para a menor versão que o aplicativo irá suportar e o atributo targetSdkVersion para a última versão, assim fazendo com que os dispositivos usem seus recursos. O Google Play será o responsável pela instalação, ou seja, se o dispositivo não for

compatível (API Level mínima), a aplicação não vai ser instalada.

Depois de ter criado o certificado keystore de release, é preciso assinar o aplicativo, através do menu Build, em seguida, Generate Signed APK ou fazer o build pelo próprio Gradle, pois temos dois tipos de builds, o debug e release.

Agora basta entrar no site https://play.google.com/apps/publish/ utilizando um com conta do Gmail. São cobrados US$ 50,00 para abrir sua conta, a partir daí poderão ser feitos uploads de aplicações.

Dentro da Google Play existe a opção Add New Application, onde são preenchidos formulários com uma breve descrição de sua aplicação e enviar imagens dela. Depois basta fazer o upload do arquivo .apk, escolher a categoria, ou seja, jogo, esporte, educação, etc. E definir se é gratuita ou paga.

2 ENGENHARIA DE REQUISITOS

Para Sommerville (2007), “O conceito de da Engenharia de Requisitos é o processo de descobrir, analisar, documentar e verificar as funções e restrições do sistema”.

Os requisitos de um sistema são as descrições dos serviços fornecidos pelo sistema e suas restrições. Eles refletem a necessidade dos clientes de um sistema que ajuda a resolver algum tipo de problema.

Na engenharia de requisitos o papel principal é procurar sistematizar o processo de definição de requisitos, então para isso é necessária uma maior atenção no entendimento do problema antes de buscar pela solução.

O conceito de engenharia de requisitos é algo que um sistema ou componente deve possuir para satisfazer um contrato, padrão ou especificação.

Segundo Pressman (2006), “A Engenharia de Requisitos é uma das primeiras etapas de alta relevância na elaboração de um sistema. Ela tem como principal objetivo elicitar, negociar, especificar, validar e gerenciar os requisitos propostos pelos clientes”.

Entende-se que as citações encontradas definem o mesmo conceito sob diferentes pontos de vista. Podemos compreender requisitos como sendo o conjunto de necessidades evidenciadas pelo cliente que deverão ser atendidas para sanar um problema específico do ramo no qual o cliente faz parte. É importante ficar atento já que embora o requisito seja definido pelo cliente, nem sempre o que o cliente quer é o que o negócio necessita. Fica a cargo da equipe de desenvolvimento a tarefa de identificar as reais necessidades do domínio de negócio.

Domínio de negócio é a área específica que o software será desenvolvido, o contexto para a solução do problema. Alguns livros trazem nomenclaturas como domínio da aplicação ou domínio do problema.

2.1 Requisitos Funcionais

Pressman (2001), diz que “Os requisitos funcionais tratam de funções que o sistema deve fornecer, como o sistema deve se comportar a estradas e a determinadas situações.”

A especificação desses requisitos deve ser integral deixando todas as funções em evidência além de não conter definições que se contradizem.

Os funcionais são as instruções dos serviços que devem ser atendidas pelos sistemas demostrando como devem ser realizadas. São subdivididos em requisitos de domínio que são aqueles originados do domínio de aplicação.

Os requisitos funcionais de um sistema descrevem o que o sistema deve fazer, e eles dependem do tipo de software que está sendo desenvolvido, dos usuários o qual o software é destinado e outros fatores da abordagem geral da organização que descreveu os requisitos.

Os requisitos funcionais de um sistema podem ser expressos de várias formas, depende do software e do usuário que ele abrange, em um requisito funcional de usuário eles definem recursos específicos a serem fornecidos pelo sistema.

2.2 Requisitos Não-funcionais

Pressman (2001), diz que “Os requisitos não funcionais dizem respeito às restrições sobre os serviços ou funções do sistema. Por exemplo, restrição de tempo, restrição do processo de desenvolvimento, padrões, etc.”

Ou seja, não se referem às funções específicas do sistema. Elas estão relacionadas com as propriedades do mesmo como confiabilidade, tempo de resposta e espaço em disco.

Os requisitos funcionais estão diretamente ligados as funções especificas fornecidas pelo sistema, e podem estar relacionados as propriedades emergentes do sistema, como confiabilidade, tempo de resposta, espaço de armazenamento. Isso implica que eles são mais importantes que os requisitos funcionais individuais.

Os requisitos não funcionais surgem devido as necessidades do usuário, as restrições de orçamento, as políticas organizacionais, a necessidade de interoperabilidade com outros sistemas de hardware ou software, ou fatores externos.

Os requisitos não-funcionais limitam o sistema que está sendo desenvolvido e o método de desenvolvimento a ser utilizado. São os requisitos não-funcionais são subdivididos em requisitos de produtos, organizacionais ou externos e estão frequentemente associados às propriedades emergentes do sistema aplicando-se a todo o sistema.

1- Requisitos de Produto: Estes especificam o comportamento do produto;
2- Requisitos Organizacionais: Estes são derivados de políticas e procedimentos da organização do cliente e do desenvolvedor;
3- Requisitos Externos: Estes abrangem os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento.

Para que haja uma concepção inicial do software esses dois tipos de requisitos são levantados procurando sempre entender o problema, os envolvidos, a natureza da solução e iniciar o processo de comunicação entre clientes e colaboradores.

2.3 Levantamento de Requisitos

À primeira vista, determinar requisitos pode aparentar uma tarefa fácil. Afinal, o cliente é a pessoa que mais conhece os objetivos gerais do sistema, o que se necessita atingir, qual será o uso do sistema na rotina da empresa, entre outros. No entanto, Pressman (2006), alerta que não é tão simples obter informações do usuário. Segundo Sommerville (2007), diversos conflitos fomentam tal afirmação: o usuário pode não saber o que deseja do sistema em termos detalhados e específicos, o relato de requisitos não realistas, diferentes tipos de usuários relatam diferentes tipos de requisitos, determinantes políticos podem influenciar os requisitos, e diversos outros fatores.

Chegando na fase de levantamento de requisitos, a equipe de desenvolvimento empenha-se em compreender o negócio que o sistema vai gerenciar. Esse levantamento também explora as necessidades dos usuários. Se já houver um sistema, a recomendação é não se prender ao mesmo e começar um novo projeto. O tempo exigido para entender o antigo sistema pode ser muito oportuno depois no novo sistema.

Um bom levantamento de requisitos começa sempre pela seleção das melhores fontes de informação que serão usadas para montar a matriz de requisitos, que será a base central para montar o escopo do projeto de software. O escopo é muito importante, pois se ele prover de algumas falhas em sua definição, indicará um trabalho mal feito na coleta e seleção das fontes mais adequadas para a coleta dos requisitos.

Não existe uma fórmula pronta para o processo de levantamento de requisitos. Obter um levantamento de requisitos bem definido é necessário o uso de técnicas a serem aplicadas em cada tipo de cenário encontrado.

É crucial o analista saber além de desenhar fluxogramas. Ele tem de entender os as informações vagas do cliente, filtrá-las, reorganizá-las e criar as soluções para cada problema.

A boa comunicação analista/cliente faz com que o software a ser desenvolvido seja bem definido evitando o retrabalho, perda de tempo e principalmente a perda de dinheiro.

3 ENGENHARIA DE SOFTWARE

Segundo Sommerville (2007), “engenharia de software é um ramo da engenharia cujo foco é o desenvolvimento dentro de custos adequados de sistemas de software de alta qualidade.”

Sommerville (2007), fala que a Engenharia de Software é uma ramificação da engenharia com a sua atenção voltada no desenvolvimento de softwares dentro de prazos, custos e qualidade adequados. Sendo o software um produto abstrato o mesmo não possui limitações físicas onde essa falta de limitações pode levá-lo a ser de difícil compreensão.

Ao decorrer dos anos, poucos pensavam que o software seria uma ferramenta tão formidável capaz de gerenciar a informação. A informática tem passado por grandes avanços tecnológicos criando sistemas perfeitos e trazendo dificuldades para as equipes de desenvolvimento de softwares mais complexos.

Surge o processo de engenharia de software como uma disciplina de engenharia relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação até a finalização do sistema e suas manutenções.

A engenharia de software não está relacionada somente com os processos técnicos de desenvolvimento de software, mas em atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software.

O processo de software está dentro da engenharia de software, ele é um conjunto de atividades e resultados associados que produz um produto de software. Para ter seu desenvolvimento tem que observar algumas atividades fundamentais:

1- Especificação de software: Fase em que os clientes e engenheiros definem o software a ser produzido e as restrições para sua operação.
2- Desenvolvimento de Software: Fase em que o software é projetado e programado
3- Validação de Software: Fase em que é feito testes para garantir que é o software está de acordo com o cliente deseja.
4- Evolução de Software: Fase em que é feita atualizações para modificar o software e ele permaneça atendendo as necessidades do mercado e do cliente.

> “As preocupações dos engenheiros de software para desenvolverem os softwares sem defeitos e entregarem estes produtos no tempo marcado, assim leva a aplicação da disciplina de engenharia de software. Com o crescimento desse segmento muitas empresas possuem mais especialistas em TI em que cada um tem sua responsabilidade no desenvolvimento de software e é diferente de antigamente que era um único profissional de software que trabalhava sozinho numa sala” (PRESSMAN, 2006).

A engenharia de software passa por fases até a entrega do software para o cliente. São elas:

- Comunicação: comunicar-se com o cliente e interessados.
- Planejamento: criar um "mapa" que guia a equipe ao fim do projeto.
- Modelagem: criar-se um "esboço" para se ter uma ideia do todo
- Construção: geração de código e teste de erros.
- Emprego: entrega do produto e feedback do cliente.

3.1 Software

Segundo Sommerville (2007), “software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente.”

Sommerville (2007), fala que a definição de software não se limita ao código do mesmo, mas também a outros artigos que o compõem como por exemplo também os seus manuais e suas especificações.

Muitos abrangem software como programas de computador, nesse termo dá-se uma visão muito restritiva. Software não é o programa em si, mas também todos os dados de documentação e configuração necessários para que o programa opere corretamente

De acordo Pressman (2006) os softwares estão categorizados em seguintes tipos, tais como:

Software de sistema: São programas que apoiam outros programas, como o software que realiza a comunicação com o hardware (sistema operacional) e software que ajuda na construção de outro software (compiladores).

Software de aplicação: São programas que são desenvolvidos para executar no negócio de uma empresa determinada.

Software científico e de engenharia: São algoritmos que processam números.

Software embutido: São programas construídos para executarem dentro de um produto especifico como as teclas digitais de um forno micro-ondas.

Software para linhas de produtos: São os softwares conhecidos como software de prateleiras.

Software de web: São aplicativos que são executados via Internet.

Software de inteligência artificial: São softwares que fazem os usos de algoritmos não numéricos.

Software aberto: São software que disponibiliza a visualização do código fonte da aplicação para o engenheiro de software modifica da maneira que deseja.

Software legado: O nome de software legado é dado quando refere se num programa de computador que foi desenvolvido por há muito tempo. A preocupação do engenheiro de software com os softwares legados está na baixa qualidade do software. Muitas vezes não existem documentações e se existem são pobres de detalhes, os casos de teste são pobres quando tem e sem um controle de mudanças. E muitas vezes não mexem no software legado quando eles atentem as necessidades do cliente (PRESSMAN, 2006).

A Engenharia de Software estabelece com precisão um dos conjuntos de atividades mais importantes a serem trabalhadas em projetos de desenvolvimento de software. Mesmo ela não garantindo uma boa qualidade do produto final, é com certeza um pré-requisito essencial para que se consiga êxito no projeto.

3.2 Modelos de Processo de Software

“Modelo de Processo de Software é uma representação abstrata de um processo de software.” (SOMMERVILLE, 2007)

Sommerville (2007), fala que este modelo fornece informações parciais sobre um determinado processo e que podem ser usadas para explicar várias abordagens para o

desenvolvimento do software.

As atividades que envolvem o desenvolvimento do software são auxiliadas por ferramentas CASE (Computer-Aided Software Engineering), entretanto elas são limitadas, pois existem diversos tipos de processos de software, que evoluíram e exploram cada vez mais as capacidades das pessoas.

Quando se define um modelo de processo garantimos uma certa organização e controle das atividades.

3.2.1 Modelo em espiral

Segundo Sommerville (2007), o modelo em espiral do processo de software foi proposto por Boehm (Boehm, 1988).

"Em vez de representar o processo de software como uma sequência de atividades como algum retorno entre uma atividade e outra, o processo é representado como uma espiral". (SOMMERVILLE, 2007).

O modelo espiral possibilita acrescentar informações aos requisitos do projeto, pois a cada volta da espiral que começa pelo centro, uma nova visão é adquirida durante o projeto. Por isso nos ajuda a encontrar detalhes nos requisitos e também alterá-los.

Quando se utiliza uma nova linguagem de programação, significa que o sistema está sujeito a alguns problemas, pois existem riscos que podem ser a causa destes problemas.

O primeiro ciclo da espiral é onde são elaborados os objetivos, que são numerados de acordo com suas restrições, para depois serem avaliados em relação as fontes de riscos. Depois resolver os riscos por meio de uma análise detalhada, como a prototipação.

> "O modelo espiral para a engenharia de software foi desenvolvido para abranger as melhores características tanto do ciclo de vida clássico como da prototipação, acrescentando ao mesmo tempo, um novo elemento – a análise dos riscos – que falta a esses paradigmas." (PRESSMAN, 1995).

Neste modelo de processo, existem fases de processo de software, que são representadas por um loop, onde o mais interno é relacionado a viabilidade do sistema, em seguida à definição de requisitos, assim por diante.

> Sommerville (2007), cita quatro setores que são divididos em um loop. São eles:
>
> 1. Definição de objetivos. Determinação dos objetivos, alternativas e restrições.
> 2. Avaliação e redução de riscos. Análise de alternativas e identificação ou resolução dos riscos.

3. Desenvolvimento e validação. Desenvolvimento do produto no próximo nível.

4. Planejamento. Revisão do projeto e tomada de decisão para prosseguir para o próximo loop.

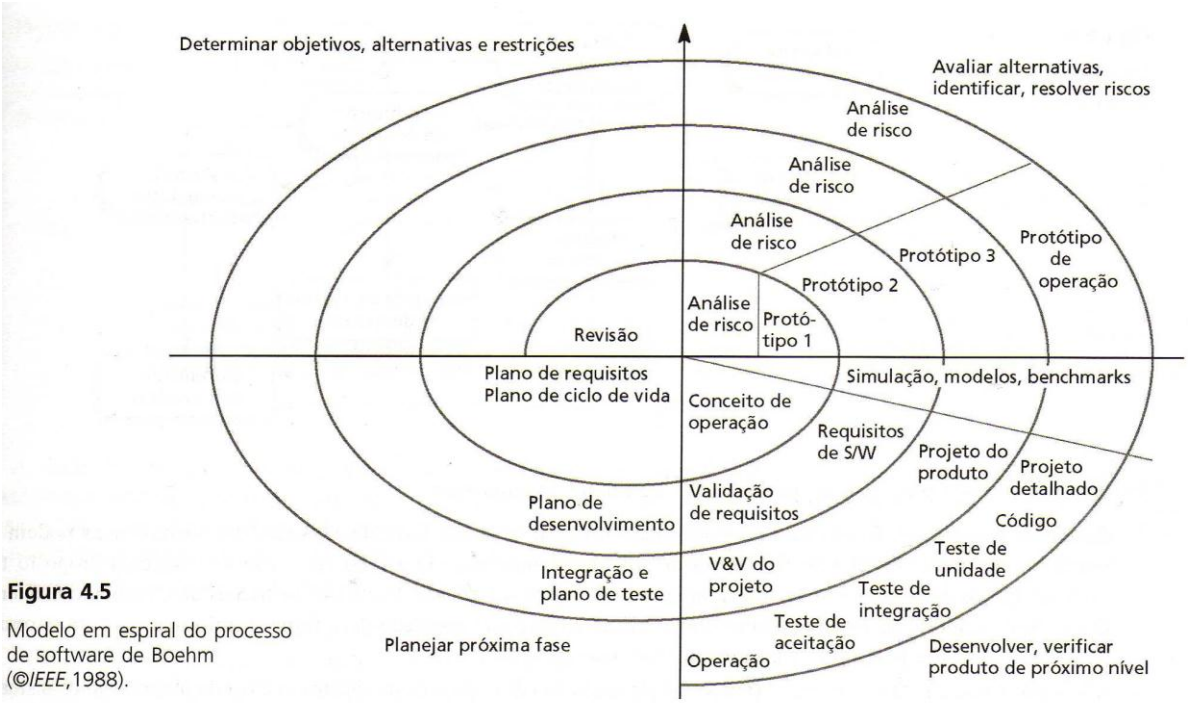

Figura 4.5

Modelo em espiral do processo de software de Boehm (©*IEEE*,1988).

Figura 1 - Modelo em espiral
Fonte: Engenharia de Software – (SOMMERVILLE, 2007)

4 UML

A UML (Unified Modeling Language), ou linguagem de modelagem unificada é “uma linguagem padrão para descrever/documentar projeto de software. A UML pode ser usada para visualizar, especificar, construir e documentar os artefatos de um sistema de software-intensivo”. (PRESSMAN, 2006)

Pressman (2006), compara a UML com plantas e projetos usados em empresas de construções por arquitetos, pois ela age da mesma forma, ajudando os arquitetos de software a construírem os diagramas UML que ajudam no desenvolvimento de software.

“UML é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos.” (GUEDES, 2011)

Guedes (2011), diz que UML não é uma linguagem de programação, e sim uma linguagem de modelagem, usada para que os engenheiros de software possam definir os requisitos e a estrutura lógica, antes mesmo de desenvolver o software.

A modelagem de software é importante, para que possa obter êxito no desenvolvimento do software, pois mesmo com profissionais que dizem saber de todas as necessidades que o software terá, provavelmente ele nunca será um projeto com uma base bem planejada. Então não importante o quanto seu projeto será complexo ou simples, ele terá que ser modelado antes de ser iniciado. Existe a probabilidade de crescimento do software, isto é, o aumento de tamanho e complexidade. Na verdade, são mudanças que decorrem de vários fatores, entre eles são pedidos de clientes para melhoria do software ou modificações, as adaptações de mercado e novas leis.

Dessa forma se vê a importância da modelagem para documentação do software, sendo uma das diversas vantagens que ele fornece.

O modelo de software nos dá uma visão de um sistema físico para que se possa ter as funcionalidades e comportamentos do sistema.

Na década de 1990 surgia a UML, desenvolvida por Grady Booch, Jim Rumbaugh e Ivar Jacobson. Foi uma mistura de notações de modelagens usadas pela indústria de software da época. A UML 1.0 foi apresentada em 1997 para a OMG (Object Management Group),

que é uma associação sem fins lucrativos e passava as especificações para as indústrias de computadores. Depois de revisada passou a ser UML 1.1. E agora tornou se o padrão ISO a UML 2.0.

4.1 Prototipação

Segundo Pressman (2006) “A prototipação é um processo que capacita o desenvolvedor a criar um modelo do software que será implementado. ”

“Essa técnica consiste em desenvolver rapidamente um rascunho do que seria um sistema de informação quando ele estivesse finalizado.” (GUEDES, 2011)

Segundo Guedes (2011), ao utilizar um protótipo, os riscos de encontrar erros após a implementação do sistema, são evitados.

O protótipo é um sistema ou modelo, que não possui funcionalidades especificas, mas serve para fins de ilustração e capacitar o melhor entendimento em forma gráfica do software. A prototipação é muito importante, pois assim mostra graficamente como vai ser “o primeiro sistema”, ajudando com isso a identificar os requisitos de software.

Um protótipo ilustra como seria a interface do software a ser desenvolvido, além de mostrar como as informações serão utilizados no sistema.

Guedes (2011), afirma que “é possível desenvolver protótipos com extrema rapidez e facilidade, por meio da utilização de ferramentas conhecidas como RAD (Rapid Application Development ou Desenvolvimento Rápido de Aplicações). ”

Existem várias ferramentas para facilitar no desenvolvimento, como NetBeans, Delphi, Visual Basic ou C++ Builder, entre outras.

4.2 Diagramas da UML

Segundo Guedes (2011), a UML possui diversos elementos para representar partes de um sistema, analisando o sistema todo ou parte dele é possível obter uma determinada visão sob diferente ótica, podendo observar uma visão externa como o Diagrama de Casos de Uso ou uma abrangência mais profunda do sistema ou software, para verificar uma característica especifica ou enfoque mais técnico.

> "O objetivo disso é fornecer múltiplas visões do sistema a ser modelado, analisando-o e modelando-o sob diversos aspectos, procurando-se assim, atingir a completitude da modelagem, permitindo que cada diagrama complemente os outros." (GUEDES, 2011, P.30)

Essa UML 2.0 dispõe de 13 diagramas diferentes usados na modelagem de software, entre eles, diagrama de caso de uso, de classes, sequência, atividade e estado.

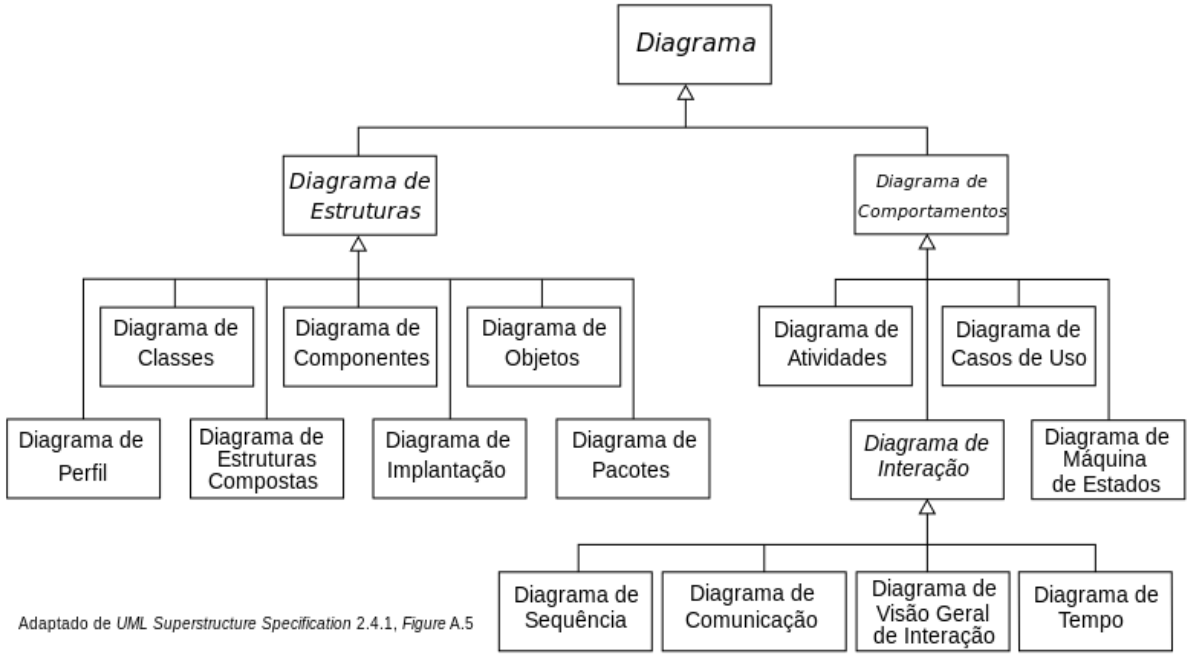

Figura 2- Diagramas da UML 2.0
Fonte:Wikipédia – UML (Visão Geral da UML)

A utilização de diversos diagramas ajuda na descoberta de falhas, para diminuir a possibilidade de erros futuros, e com isso minimizando os custos de desenvolvimento.

O objetivo desses diagramas é dar vários tipos de visões do sistema que está sendo modelado, assim analisando como um todo e permitindo que um diagrama complemente o outro, como se fosse em camadas, desde uma visão externa até as características mais complexas do sistema.

Então diversos fatores influenciam na utilização dos diagramas, como descobrir falhas no sistema, diminuindo o risco de erros futuramente e análise em diferentes níveis.

Os diagramas da UML 2.0 dividem-se em dois grupos: Diagramas Estruturais e os Diagramas Comportamentais, estes possuem uma subdivisão representada pelos Diagramas de Interação.

4.3 Diagrama de Caso de Uso

Segundo Guedes (2011, p.30), ”O diagrama de casos de uso é o diagrama mais geral e informal da UML [...]”, diz- se assim por que sua utilidade vem normalmente nas fases iniciais de levantamento e análise de requisitos do sistema e é necessário ser consultado a medida que o processo de modelagem é fundamentado.

O diagrama de caso de uso possibilita qualquer pessoa de compreender o comportamento do externo sistema, ou seja, suas funcionalidades, através de uma linguagem simples. É utilizado geralmente, no início da modelagem, quando está na fase de levantamento e análise de requisitos, apesar de sofrer mudanças ao decorrer do projeto, e também é usado como base para outros diagramas.

O objetivo é mostrar ao usuário uma visão geral de como o sistema irá se comportar. Serve para identificar os requisitos do sistema, ajudando a especificar, visualizar e documentar as funções que o usuário deseja. Todos os usuários que interagem diretamente com o sistema são identificados, e quais papeis serão desempenhados por eles.

O diagrama de caso de uso contém dois itens principais que são atores e caso de uso. Os atores representam o papel que o usuário desempenha, podendo ser qualquer elemento externo que interaja com o sistema. São representados no diagrama por um boneco contendo a descrição do ator.

Os casos de uso são utilizados para capturar requisitos do sistema, ou seja, referem-se aos serviços, tarefas ou funcionalidades identificadas como necessários ao software e que podem ser utilizados de alguma maneira pelos atores.

4.4 Documentação de Casos de Uso

Descreve por meio de uma linguagem simples, as informações contidas nos casos de uso, as funções gerais que os atores desempenham e quais restrições e validações o caso de uso deve ter.

> Segundo Guedes (2011), “não existe um formato específico de documentação para casos de uso definidos pela UML, o que está de acordo com a característica do próprio diagrama, ou seja, o formato de documentação de um caso de uso é bastante flexível, permitindo que se documente o caso de uso de forma que se considerar melhor, até mesmo por meio do uso de pseudocódigo ou da utilização do código de

> uma linguagem de programação propriamente dita, embora isso fuja bastante do objetivo principal do diagrama, que é utilizar uma linguagem simples que até mesmo usuários leigos possam entender".

Primeiramente deve se fornecer uma descrição para o caso de uso que será documentado, ou seja a mesma descrição que já foi feita na criação do caso de uso. Depois são fornecidas informações sobre o caso de uso geral para saber de qual caso de uso ele é derivado.

Seguindo temos, Ator Principal, que é o que mais interage com o caso de uso.

Ator Secundário, que são aqueles que interagem menos com o caso de uso ou que não tem muito interesse nos resultados.

Resumo, onde uma breve explicação do objetivo.

Pré-condições e Pós-Condições para saber os requisitos antes de começar e depois que terminar o caso de uso.

Fluxo principal, que representa quem realizam determinadas ações.

Fluxos Alternativos, que mostram outros passos para determinadas ações

Fluxos de Exceção, que basicamente representam uma regra para que certas ações possam ser realizadas.

4.5 Diagrama de Classes

Segundo Guedes (2011), o diagrama de classes é um dos mais importantes da UML, mostra de forma ampla a estrutura de um sistema, seus atributos, operações, relações e associações com outras classes. Isso é feito através de ícones que representam as classes e suas respectivas ligações. Dentro destas caixas estão divididos os atributos e métodos. Os atributos podem ter um nome, um tipo e um nível de visibilidade, que são opcionais. Os métodos estão na parte de baixo das caixas e se referem ao que os objetos da classe podem fazer.

> "O diagrama de classes é provavelmente o mais utilizado[...]". "[...] define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de esclarecer como as classes se relacionam e trocam informações entre si." (GUEDES, 2011, P.31)

As classes também podem estar relacionadas, ou seja, uma classe tem algum tipo de vínculo à outra, podendo ser generalização, associação, agregação, composição e dependência.

Generalização é quando uma subclasse é ligada a outra classe por uma linha sólida e a ponta triangular vazia. A seta aponta da subclasse para a superclasse. Em outras palavras é como se tivesse uma superclasse Carro, por exemplo, com as subclasses Terrestre e Aéreo.

Associação indica que as classes estão ligadas por meio de uma relação estrutural e são representadas por linhas sólidas. Podem ser rotuladas, para saber qual é o papel de cada classe, setas para se ter navegabilidade, e um valor de multiplicidade nas extremidades das linhas.

Agregação indica a relação da parte com o todo, ou seja, uma linha com um losango vazio aponta para a parte da classe que representa o todo. Essa parte não existe sem o todo por isso está agregada a ele.

Composição indica que as partes vivem e morrem com o proprietário, por que não têm um papel no sistema independente do proprietário. Supondo que uma classe Escola tivesse uma agregação ao objeto Edifício, mesmo que a Escola fechasse, os edifícios ainda continuariam a existir, mas com outras finalidades.

Dependência representa outra conexão entre classes representada por uma linha tracejada, onde uma classe depende da outra e se alterar algo na segunda classe terá que alterar na primeira também. Quando uma classe é associada a outra, automaticamente ela indica dependência, mas não é preciso usar a linha pontilhada se houver a associação.

4.6 Diagrama de Sequência

Segundo Guedes (2011), “Um diagrama de sequência costuma identificar o evento, e determina como o processo deve se desenrolar e ser concluído por meio da chamada de métodos disparados por mensagens enviadas entre os objetos.”

Diagrama de sequência é um diagrama que mostra a ordem temporal que as mensagens são enviadas entre os objetos durante a execução de alguma tarefa. São mostrados todos os passos detalhadamente, desde o evento gerador do processo, como o ator responsável por esse evento. E como o processo está sendo realizado, por meio da chamada de métodos e mensagens enviadas entre objetos.

O Diagrama de Sequencia trata-se de uma ferramenta UML para formalizar a representação de interação entre o cenário e o objeto feitas através dos métodos, a sequência em que a troca de mensagem é feita deve ser retratada por ordem todos os passos seguidos até o fim do processo. Este diagrama é construído a partir da construção de outro diagrama: Diagrama de Casos de Usos.

5 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

"Sistema de Gerenciamento de banco de dados (DBMS) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados." (SILBERSCHATZ, 2006).

O objetivo do SGBD é auxiliar na recuperação de informações de banco de dados, e fornecer mecanismos para a manipulação de informações.

Silberschatz (2006), define DBMS, ou SGBD em português, como um conjunto de dados organizados que são acessados por um programa específico.

Date (2003), diz que SGBD "é o software que trata todo o acesso ao banco de dados".

Esses dados organizados são chamados de banco de dados, onde estão localizadas as informações.

Com uma recuperação de dados eficiente, o sistema passa a ser funcional. Mas para que possa ser eficiente, antes de tudo são usadas estruturas de dados complexas em vários níveis de abstração, que são o nível físico, nível lógico e nível de view.

Na estrutura de um banco de dados está o modelo de dados, que são ferramentas conceituais utilizadas para descrever dados, relações, semântica e restrições. Existem vários modelos de dados diferentes, e entre eles usamos o Modelo de Entidade/Relacionamento e o Modelo Relacional Normalizado.

5.1 MER

"O Modelo de Entidade/Relacionamento (E-R) é baseado em uma percepção de um mundo real que consiste em uma coleção de objetos básicos, chamados entidades, e as relações entre esses objetos".

Silberschatz (2006), diz que o Modelo de Entidade/Relacionamento é um modelo abstrato que tem por finalidade descrever, de forma conceitual, os dados, através de entidades que contém atributos e que são relacionadas.

Entidades e relacionamentos tem propriedades, ou seja, em um exemplo de funcionário, pode ter nome, salário, etc.

Date (2003), diz que "Cada espécie de propriedade tira seus valores de um conjunto de valores correspondentes (isto é, domínio, em outras palavras)."

As propriedades podem ser, simples ou compostas, chave, multivalorada.

Existem também os relacionamentos, que são as ligações entre as entidades, podendo

ser um para um, de um para muitos, ou de muitos para muitos.

Foi desenvolvido para facilitar o projeto de banco de dados, permitindo criar a especificação de um esquema que represente a estrutura lógica de um banco de dados.

5.2 MRN

"O Modelo Relacional usa uma coleção de tabelas para representar os dados e as relações entre eles. Cada tabela possui diversas colunas, e cada coluna possui um nome único." (SILBERSCHATZ, 2006).

Este modelo é baseado em tabelas relacionadas entre si, onde estão guardados todos os dados. Onde a atribuição de valores inicia um registro na tabela, e a relação faz com que cada registro seja relacionado a outras tabelas.

Silberschatz (2006), diz que o Modelo Relacional é composto por tabelas que representam os dados, baseados em registros, onde cada tabela contém registros que definem um número fixo de campos, ou atributos.

O Modelo Relacional Normalizado nos permite uma melhor manipulação dos dados dentro do banco de dados, ou seja, implementa estruturas de dados organizados em relações, mas para manipular essas tabelas, temos que fazer restrições para não repetir informações, por isso este modelo é a base para a criação de um banco de dados.

> Segundo Date (2003), "Os objetivos gerais do processo de normalização são os seguintes:
> - Eliminar certas espécies de redundâncias.
> - Evitar certas anomalias de atualização.
> - Produzir um projeto que seja uma boa representação do mundo real – isto é, que seja intuitivamente fácil de entender e uma boa base para crescimento futuro.
> - Simplificar a imposição de certas restrições de integridade. "

5.3 MySQL

Segundo Milani (2012), Mysql é um servidor e gerenciador de banco de dados também representado pela sigla (SGBSD), tem sua versão de código livre muito utilizado para trabalhar com aplicações de pequeno e médio porte, seu potencial é comparado e reconhecido por programas de código fechado como o SQL Server. O Mysql é o banco de dados open source mais popular e com maior capacidade de tratamento de dados do mercado.

A origem do Mysql veio depois que alguns desenvolvedores da década de 90 precisaram de uma interface SQL que seja compatível com as rotinas ISAM. Com base numa API do mSQL tais desenvolvedores escreveram uma nova API usando C e C++, assim originou o MySQL, desse tempo até hoje, ele foi difundido e suas características de rápido acesso o tornou muito popular.

Sua utilização recomendada é para trabalhar com aplicações medianas, mas se adequa a grande capacidade de armazenamento, em torno de 100 milhões de registros por tabela. O MySQL é altamente confiável e rápido por armazenar código de baixo nível. Seu uso atualmente atende uma demanda maior para aplicações para internet, como lojas virtuais que precisam de acesso rápido para a geração de páginas HTML. Ele tem suporte a maioria dos Sistemas Operacionais existentes.

O MySQL contém todas características de um SGBD, como armazenamento de dados, características de multiacesso aos dados e várias outras funcionalidades de um SGBD.

Segundo Milani (2012), “O MySQL possui gerenciamento de acesso, integridade dos dados e relacional, concorrência, transações, entre outros”

A segurança do MySQL trabalha com Sistema gerenciador de conexões com criptografias no trafego de senhas e uso de autenticação, até mesmo com a opção de habilitar o firewall para autenticação fornecendo conexões habilitadas para cada estação e domínios especificados em sua lista de acesso.

O MySQL é open source, ou seja, de código aberto e para conseguir esse software, basta ter acesso a internet necessariamente ao site oficial do MySQL, onde será possível encontrar os pacotes de instalação dessa ferramenta. Após o download basta seguir as sequencias para conseguir instalar o software no Sistema Operacional.

Para auxílio ao MySQL é utilizado uma ferramenta gráfica e simples conhecida como PhpMyAdmin, usada no gerenciamento e administração do servidor e das bases de dados Mysql. O PhpMyAdmin usa uma interface gráfica Web, para facilitar o usuário na movimentação das bases de dados, criação e manutenção de tabelas, gerenciamento de processos, tudo isso com uma forma mais simplificada e de forma visual intuitiva para o usuário. E o melhor é que tudo que pode ser feita através do terminal e códigos pode ser usado no PhpMyAdmin.

Para Milani (2012) “O grande objetivo do PhpMyAdmin é tornar mais simples e pratica a interação com o MySQL, utilizando os conceitos de programação gráfica”

Para obtenção do PhpMyAdmin basta acessar o site oficial da ferramenta e procurar pela área de Download, a instalação é simples e fácil basta seguir os passos que o aplicativo de instalação fornece.

O PhpMyAdmin fornece um total conjunto para gerenciamento de bases de dados, desde a criação, manutenção, população e exclusão de dados, tudo de forma automatizada e intuitiva com o usuário. Além de ser possível também ter uma completa interação com SQL por meio de queries SQL, caso de backup, perda e restauração é possível exportar ou importar tabelas via SQL, o que se torna muito importante.

> “Tratando um pouco a questão de migração de banco de dados, ou até mesmo arquivos-texto de backups, o PhpMyadmin possui um modulo completo para a realização dessas operações. Tanto para a exportação ou importação de apenas uma tabela quanto de todas as tabelas de uma determinada base de dados, e até mesmo de vários bancos de dados ao mesmo tempo”. (MILANI, 2012)

6 ESTUDO DE CASO

6.1 Descrição da Necessidade

A grande dificuldade que se encontra no restaurante, é a disponibilidade de garçons e a demora na preparação para entregas, entretanto este problema não está relacionado à cozinha, pois esta demora vem das anotações de pedidos e dados dos clientes. Então o tempo que o garçom gasta fazendo as anotações, o cozinheiro poderia estar preparando os pedidos. Este problema se dá por conta de atrasos ou mudanças dos pedidos, que ocorrem quando os clientes fazem suas escolhas. Vários fatores influenciam nessa parte, como por exemplo, uma má comunicação entre cliente e garçom, seja ela causada por idioma ou mesmo por mal entendimento das palavras. A escrita feita por garçons nem sempre é legível, mas para que essas informações possam ser repassadas corretamente à cozinha, é preciso uma identificação dos pedidos feita pelo próprio garçom que a fez, pelo contrário irá causar confusão no preparo do mesmo.

6.2 Objetivo

Essa aplicação Android para autoatendimento em restaurante, tem como objetivo a realização de pedidos através de smartphones, uma vez que o cliente cadastrado no sistema, possa realizar seu pedido com maior rapidez, sem necessidade de repetir dados na hora da compra, além de facilitar os serviços dos garçons e todo pessoal da cozinha. A partir do momento em que o cliente dispõe seu tempo para preencher dados cadastrais, já poderia ter realizado seu pedido. Então essa aplicação pode agilizar ainda mais no seu processo, sem ter que informar seu nome e endereço a cada vez que utilizar o sistema. Com uma interface totalmente interativa, o cliente irá se sentir mais à vontade e poderá realizar seus pedidos com mais clareza, pois estará vendo o que irá receber através da tela de seu celular, sem risco de fraude.

6.3 Escopo

O intuito é poder estabelecer um vínculo entre o aplicativo e cliente, de maneira que trabalhem como uma ponte de comunicação. Transmitindo dados de forma ágil, segura e com precisão ao restaurante, sem risco de erros. Isto fica evidente em comparação à finais de semana que são dias de maior movimento e consequentemente os erros ocorrem com maior frequência. Levando em consideração essas necessidades presentes no cotidiano do restaurante, podemos afirmar que o aplicativo será uma importante ferramenta no auxílio ao processo de pedido, promovendo ao usuário mais independência, mobilidade, precisão e agilidade em suas atividades. Assim os problemas de lentidão, transtorno e erros podem ser minimizados, otimizando a satisfação do cliente que implica na obtenção de lucro e crescimento da empresa, garantindo preferência nos serviços prestados.

6.4 Descrição Geral do Cliente

O cliente já utiliza um sistema para controle de estoque. Esse entendimento básico já serviu de grande ajuda no levantamento de requisitos, o que nos trouxe facilidade para coletar dados. Utilizando a nova aplicação, o cliente poderá ampliar ainda mais seu conhecimento nessa área que se evolui a cada instante, além de ter funcionalidades que proporcionam atendimento com agilidade e qualidade. Um sistema inovador acrescentado a experiência de quem conhece o trabalho no ramo de alimentação, resultará em vantagens de utilização para o usuário.

6.5 Lista de Requisitos Funcionais

Nº	Interesses	Motivo	Solicitante	Data	Ator
R1	Gerenciar Conta	Armazenar dados pessoais de clientes, assim como a alteração e exclusão desses dados.	Gerente	04/07/2015	Usuário
R2	Manter Usuário	Controle das informações de cadastro do cliente, especialmente para separar os clientes ativos e inativos.	Gerente	04/07/2015	Gerente

R2	Gerenciar Pedido	Armazenar dados do produto que o cliente estará consumindo, bem como o acompanhamento do mesmo, desde a aprovação até a entrega.	Gerente	04/07/2015	Funcionário
R3	Gerenciar cardápio	Descrever os pratos e bebidas com seus respectivos preços.	Gerente	04/07/2015	Gerente
R4	Gerenciar eventos	Adicionar imagens e informações adicionais sobre a data de shows e eventos no local.	Gerente	04/07/2015	Gerente
R5	Gerenciar mesa	Controlar o status das mesas, assim como o horário de entrada e duração, e reservas.	Gerente	04/07/2015	Gerente

Tabela 1- Lista de Requisitos

6.6 Requisitos Não Funcionais

- Segurança

  Autenticação do Usuário com Login e Senha.

- Compatibilidade

O smartphone deve possuir compatibilidade com Android e API mínima suportada 9 referente a versão 2.3 do Android conhecida como (GingerBread) e a API Alvo é 21 correspondentes a versão 5.0 do Android conhecido como (Lolipop)

- Conexão

Disponibilidade de Conexão a Internet, o dispositivo móvel deve ter acesso a internet para ser possível carregar as imagens e dados do web service.

- Desempenho

Referente ao nível que o sistema utiliza recursos dos smartphones escassos, como memória, CPU, processador e espaço em disco. Pois não adianta ter um bom processador e não ter memória suficiente no dispositivo.

Figura 3 - Pouco espaço de armazenamento
Fonte: Elaborada pelo autor

6.7 Diagramas de Casos de Uso

6.7.1 Diagrama de Caso de Uso de Software

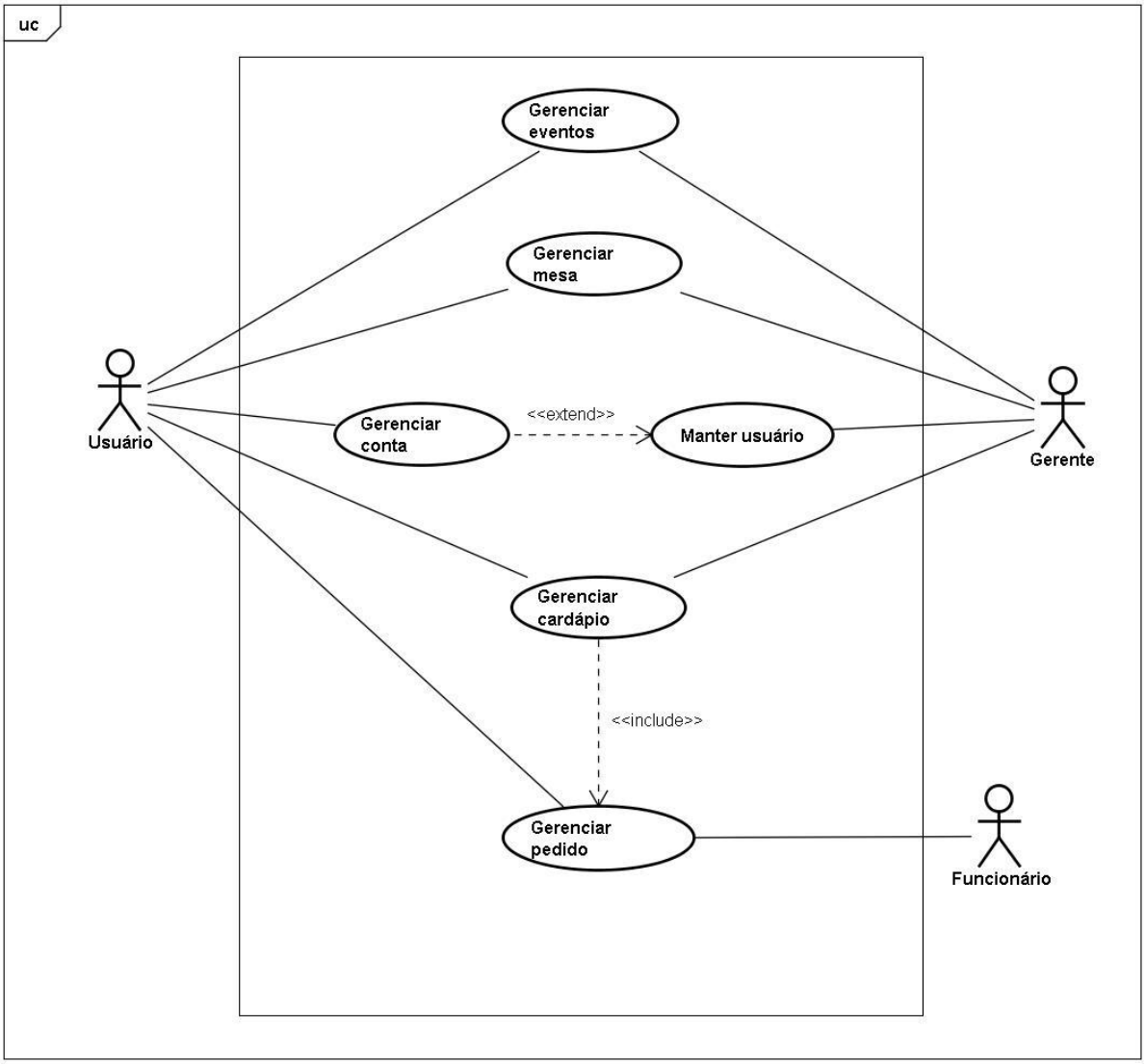

Figura 4 - Diagrama de Caso de Uso de Software

6.8 Documentação do Caso de Uso

6.8.1 Documentação de Caso de Uso: UC 01 - Gerenciar Conta

Nome do caso de uso	Gerenciar Conta
Caso de Uso Geral	
Ator Principal	Usuário
Atores Secundários	Gerente
Resumo	Este caso de uso permite que o usuário possa se cadastrar e assim ter acesso às áreas privilegiadas do sistema, como visualização de eventos, reserva de mesa, além de incluir, alterar ou excluir qualquer informação relacionada à sua conta.
Pré-Condições	O ator deverá preencher um formulário com suas informações pessoais para se cadastrar no sistema
Pós-Condições	Após a confirmação dos dados será necessário logar no sistema
Fluxo Principal	O fluxo principal inicia quando o usuário solicita ao sistema a funcionalidade cadastro de usuário.
Ações do Ator	Ações do Sistema
1. O ator solicita novo cadastro	
	2. O sistema disponibiliza um formulário com campos de preenchimento
3. O ator preenche os campos e confirma seus dados.	
	4. Se for solicitada função salvar, o sistema guarda as informações no banco de dados [MSG. 0012]
	5. O sistema retorna uma tela de login
6. O ator informa login e senha	
	7. O sistema retorna uma tela de carregamento [MSG. 0013]

	8. O Sistema retorna uma tela informando que o usuário está logado [MSG. 0016].
Fluxo de Exceção	
Ações do Ator	Ações do Sistema
	1. Caso o usuário não preencha os campos obrigatórios, o sistema retorna uma mensagem [MSG. 001] até a [MSG. 007], dizendo qual o campo deverá ser preenchido.
	2. Caso algum campo preenchido não corresponder, o sistema retorna uma mensagem [MSG. 0010] e retorna ao item 3.
	3. Caso o CPF do usuário seja inválido, retornar mensagem de erro [MSG. 009] e retorna ao item 3.
	4. Caso o Email do usuário seja inválido, retornar mensagem de erro [MSG. 008] e retorna ao item 3.
	5. Caso o usuário preencha os campos obrigatórios com dados já existentes o sistema retorna uma mensagem [MSG. 0011] e retorna ao item 3.
	6. Após três tentativas de login falharem, o sistema retorna uma mensagem de recuperação de senha [MSG. 0015].
Fluxo Alternativo - Alteração de dados	
Ações do Ator	Ações do Sistema
1. O usuário solicita alteração de cadastro	
	2. O sistema retorna um formulário com seus dados já preenchidos
3. O usuário realiza sua alteração	
	4. O sistema retorna uma mensagem que sua alteração foi realizada com sucesso [MSG. 0017]
Restrições/Validações	Não possui
	Não possui

Tabela 2 - Documentação de Caso de Uso: UC 01 - Gerenciar Conta

6.8.2. Documentação de Caso de Uso: UC 02 - Manter Usuário

Nome do caso de uso	Manter Usuário
Caso de Uso Geral	
Ator Principal	Gerente
Atores Secundários	Usuário
Resumo	Este caso de uso tem como objetivo manter os dados cadastrais do usuário, ou seja, permite incluir, alterar, ou consultar dados. Um cliente pode ser excluído ou se tornar inativo
Pré-Condições	O ator deverá estar autenticado no sistema e possuir permissão para executar esta funcionalidade
Pós-Condições	Após a conclusão da tarefa o sistema registra os dados do usuário no banco de dados
Fluxo Principal	O fluxo principal inicia quando qualquer usuário entra no sistema e clica na opção Usuários – Gerenciar Usuários.
Ações do Ator	Ações do Sistema
1. O ator seleciona o usuário	
	2. O sistema disponibiliza tela com os dados do usuário selecionado
	3. O sistema fornece opções de inserir, alterar e excluir, além de tornar o usuário inativo.
4. O ator realiza sua tarefa	
	5. O sistema guarda as informações no banco de dados [MSG. 0011]
Restrições/Validações	Não possui
	Não possui

Tabela 3 - Documentação de Caso de Uso: UC 02 - Manter Usuário

6.8.3. Documentação de Caso de Uso: UC 03 - Gerenciar Pedido

Nome do caso de uso	Gerenciar Pedido
Caso de Uso Geral	
Ator Principal	Usuário
Atores Secundários	Funcionário
Resumo	Este caso de uso permite que o usuário possa gerenciar suas informações relacionadas ao pedido. Isso inclui adicionar, alterar e remover item do carrinho de compra, consultar status do pedido e cancelar compra. E também a opção de escolher o local da entrega, ou seja, se ele está em sua residência ou se está em determinada mesa no local.
Pré-Condições	O ator deve estar logado no sistema
Pós-Condições	Após a confirmação do pedido será gerado um relatório
Fluxo Principal	O fluxo principal inicia quando o usuário adicionar algum item no carrinho de compra e finaliza sua compra
Ações do Ator	Ações do Sistema
1. O ator solicita o cardápio	
	2. O sistema retorna uma tela de cardápio contendo todos itens e seus respectivos preços
3. O ator adiciona os produtos ao carrinho.	
	4. O sistema guarda no banco de dados
	5. O sistema retorna uma tela para finalizar a compra
6. O ator seleciona o local e a forma de pagamento	
7. O ator informa a quantia para o troco	
	8. O sistema calcula o troco do usuário

9. O ator confirma a compra	
	10. O sistema retorna uma mensagem de sucesso [MSG. 0020 - 0021] e gera um comprovante
Fluxo de Exceção	
Ações do Ator	Ações do Sistema
	1. Caso o pedido já esteja aprovado, não há possibilidade de cancelamento.
Fluxo Alternativo	
Ações do Ator	Ações do Sistema
	1. Caso o usuário não marque uma opção de pagamento, o sistema retorna uma mensagem [MSG. 0018] e retorna ao item 6.
	2. Caso o usuário não marque uma opção de entrega, o sistema retorna uma mensagem [MSG. 0019] e retorna ao item 6.
Fluxo Alternativo – Consultar status do pedido	
Ações do Ator	Ações do Sistema
1. O usuário consulta a aba “Meus pedidos”	
	2. O sistema retorna uma tela com todos seus pedidos
3. O usuário seleciona o pedido	
	O sistema retorna uma tela com a data, hora, preço e status (em andamento/em espera) do pedido
Restrições/Validações	Não possui
	Não possui

Tabela 4 - Documentação de Caso de Uso: UC 03 - Gerenciar Pedido

6.8.4. Documentação de Caso de Uso: UC 04 - Gerenciar Mesa

Nome do caso de uso	Gerenciar Mesa
Caso de Uso Geral	
Ator Principal	Usuário
Atores Secundários	Gerente
Resumo	Este caso de uso permite que o usuário possa visualizar as mesas disponíveis na data prevista e reserva-la. Assim como poderá acrescentar o número de assentos, caso tenha vagas.
Pré-Condições	O ator deve estar logado no sistema
Pós-Condições	Depois de reservar a mesa, caso o usuário queira cancelar, esse procedimento só poderá ser feito dois dias antes da data da reserva.
Fluxo Principal	O fluxo principal inicia quando o usuário seleciona uma mesa disponível e a reserva.
Ações do Ator	Ações do Sistema
1. O ator solicita as mesas	
	2. O sistema retorna uma tela de contendo todas mesas disponíveis
3. O ator seleciona a mesa	
	4. O sistema solicita o número de assentos.
5. O ator informa a quantidade	
	6. O sistema grava no banco de dados
	7. O sistema retorna uma mensagem de sucesso [MSG – 0022]
Restrições/Validações	Não possui
	Não possui

Tabela 5 - Documentação de Caso de Uso: UC 04 - Gerenciar Mesa

6.8.5. Documentação de Caso de Uso: UC 05 - Gerenciar Eventos

Nome do caso de uso	Gerenciar Eventos
Caso de Uso Geral	
Ator Principal	Gerente
Atores Secundários	Usuário
Resumo	Este caso de uso permite que o gerente possa adicionar informações sobre determinado evento que ocorrerá na empresa, assim como atribuir quem serão os cantores, a data do evento, e o preço da entrada. O usuário pode acessar o sistema e visualizar informações dos eventos que serão realizados e suas respectivas datas, assim como informações dos artistas e o preço.
Pré-Condições	O ator deve estar logado no sistema
Pós-Condições	O ator somente visualiza as informações sobre o evento, para marcar presença, deve voltar a área de reservas.
Fluxo Principal	O fluxo principal inicia quando o Gerente solicita ao sistema, a função de adicionar evento, contendo opções de incluir informações sobre determinado evento.
Ações do Ator	Ações do Sistema
1. O ator solicita os eventos	
	2. O sistema retorna uma tela contendo campos de texto para serem preenchidos com as informações do evento, como data, preço, nome dos artistas, nome do evento e vagas.
3. O ator preenche os campos e confirma	
	4. O sistema grava no banco
	5. O sistema retorna uma mensagem de sucesso [MSG. 0023]

Fluxo Alternativo I– Visualizar Evento	O fluxo alternativo visualizar evento inicia quando o usuário acessa o menu de eventos e visualiza as informações de um determinado evento.
Ações do Ator	Ações do Sistema
1. O ator solicita os eventos	
	2. O sistema retorna uma tela de contendo todas datas onde serão realizados os eventos.
Fluxo Alternativo II – Participar do Evento	
Ações do Ator	Ações do Sistema
1. O ator solicita participar do evento	
	2. O sistema retorna uma tela contendo as vagas disponíveis para o evento selecionado [UC - 04] item 1.
Restrições/Validações	Não possui
	Não possui

Tabela 6 - Documentação de Caso de Uso: UC 05 - Gerenciar Eventos

6.8.6. Documentação de Caso de Uso: UC 06 – Gerenciar Cardápio

Nome do caso de uso	Gerenciar Cardápio
Caso de Uso Geral	
Ator Principal	Gerente
Atores Secundários	Usuário
Resumo	Este caso de uso permite que o gerente possa acrescentar, alterar e remover algum tipo de comida, bebida ou sobremesa e seus respectivos preços. Assim como o usuário pode visualizá-lo e selecionar os itens do seu interesse para compra.
Pré-Condições	O ator deve estar logado no sistema
Pós-Condições	Depois que um item é acrescentado, o gerente deve usar o botão salvar, para que o sistema grave as informações no banco

Fluxo Principal	O fluxo principal inicia quando o gerente solicita ao sistema, alguma função de acrescentar, alterar ou remover um item.
Ações do Ator	Ações do Sistema
1. O ator solicita o cardápio	
	2. O sistema retorna uma tela de contendo o menu de comida, bebida e sobremesa, contendo em cada um deles, os produtos e seus respectivos preços.
3. O ator seleciona adicionar item	
	4. O sistema retorna uma tela com os campos de texto para serem preenchidos com as informações do produto, e também selecionar uma foto para demonstração.
5. O ator preenche e confirma	
	6. O sistema grava no banco
	7. O sistema retorna uma mensagem de sucesso [MSG. 0024]
Fluxo Alternativo I – Visualizar Cardápio	Ações do Sistema
1. O ator seleciona cardápio	
	2. O sistema retorna um tela contendo um menu com todos os produtos e seus respectivos preços.
3. O ator seleciona o produto	
	4. O sistema retorna um tela com a descrição, preço e foto do produto, podendo ser adicionado ao carrinho [UC – 03] item 3.
Restrições/Validações	Não possui
	Não possui

Tabela 7 - Documentação de Caso de Uso: UC 06 – Gerenciar Cardápio

6.9 Requisitos de Dados

6.9.1 MER

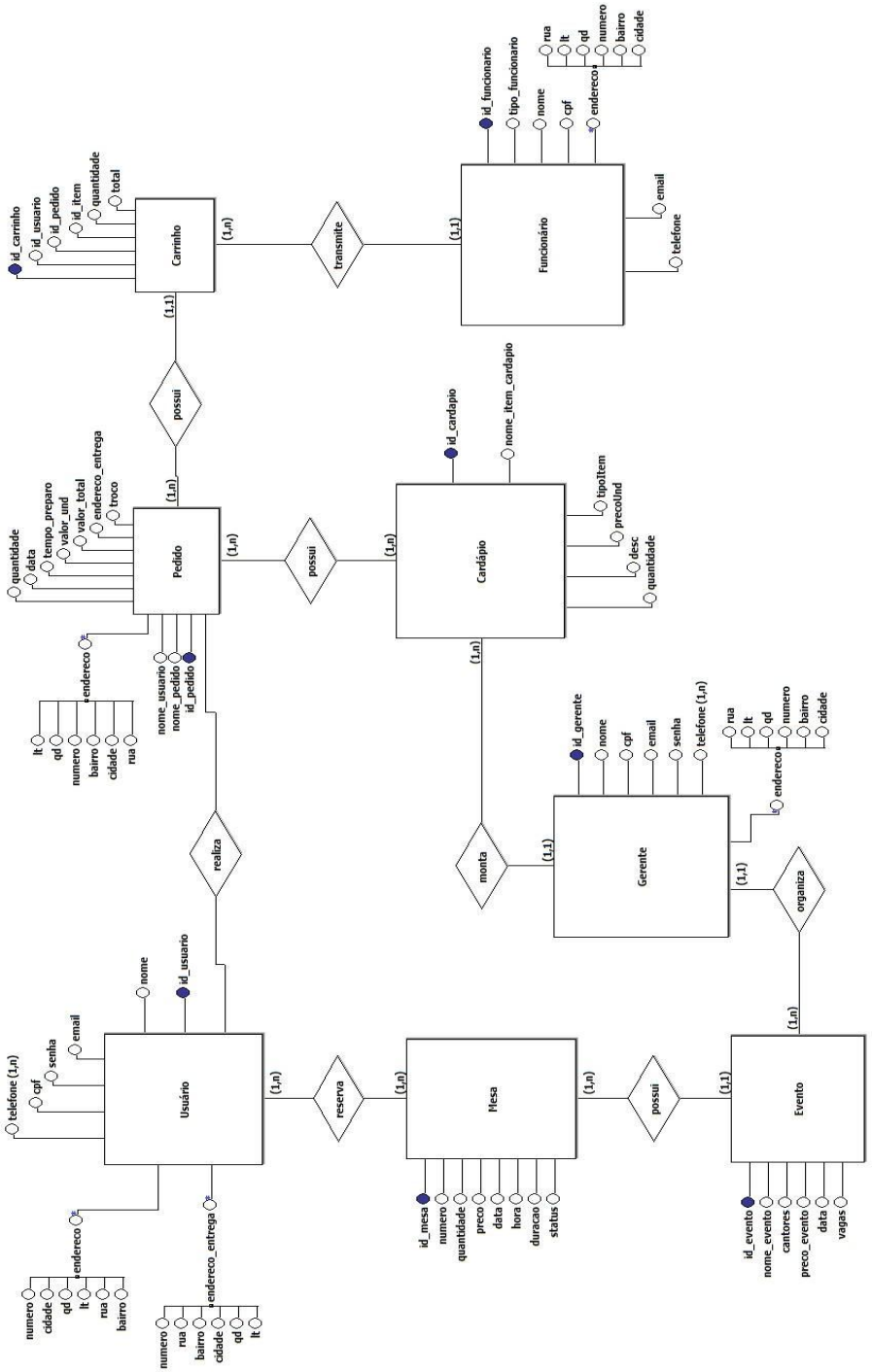

Figura 5 – MER

6.9.2 MRN

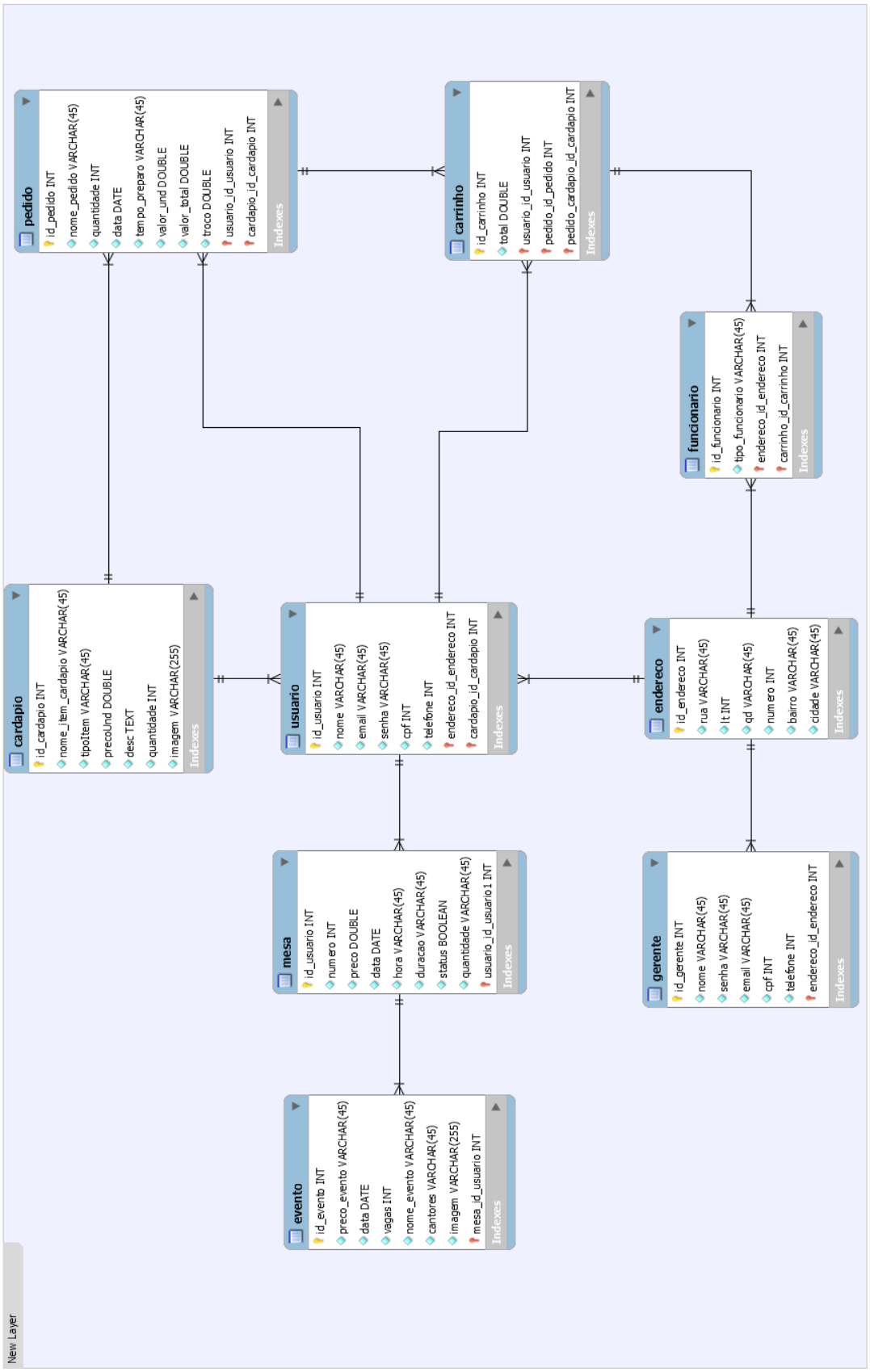

Figura 6- MRN

6.10 **Diagrama de Classes**

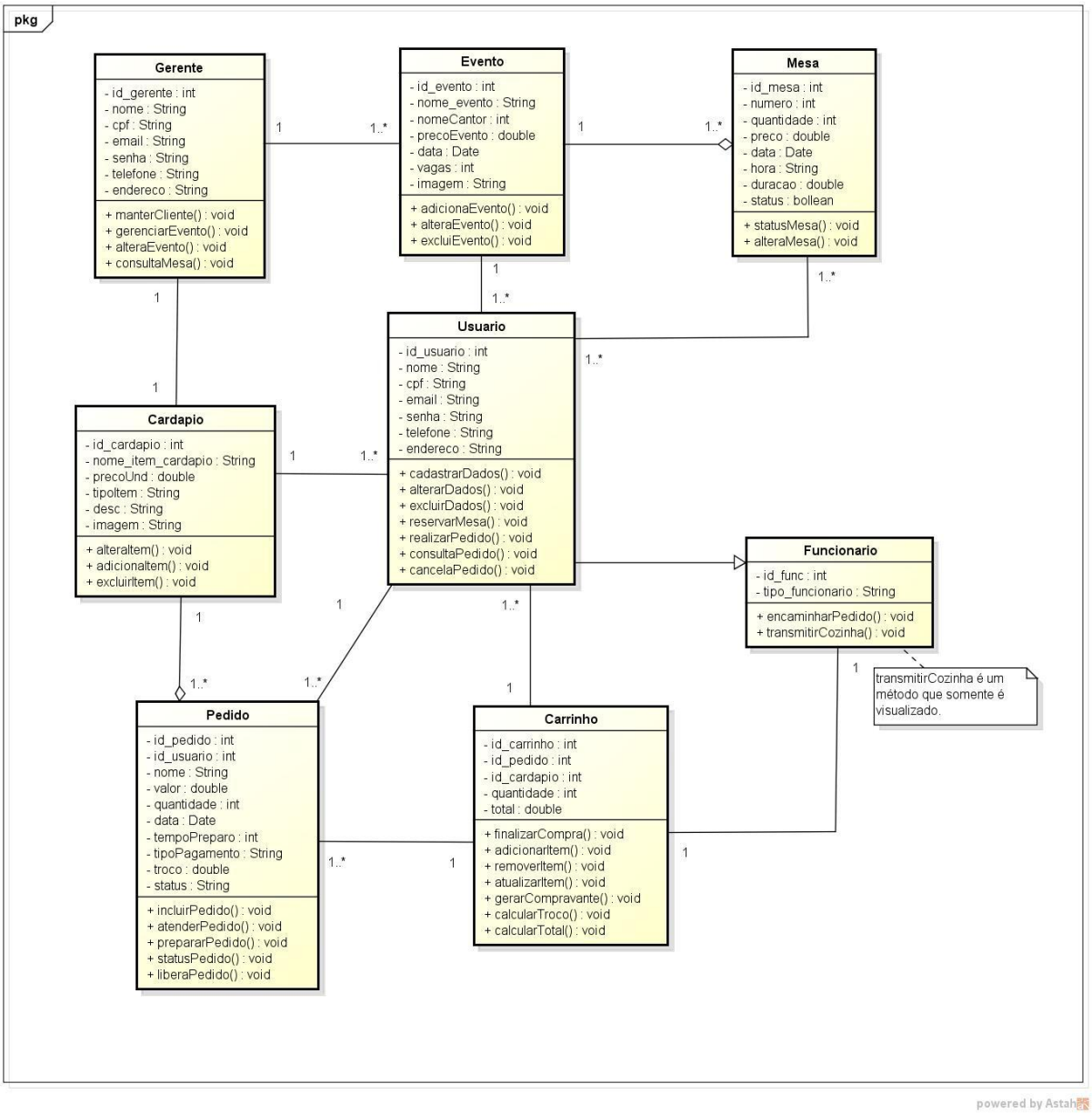

Figura 7 - Diagrama de Classe

6.11 Diagrama de Sequência

6.11.1 Diagrama de Sequência – Cadastrar Login

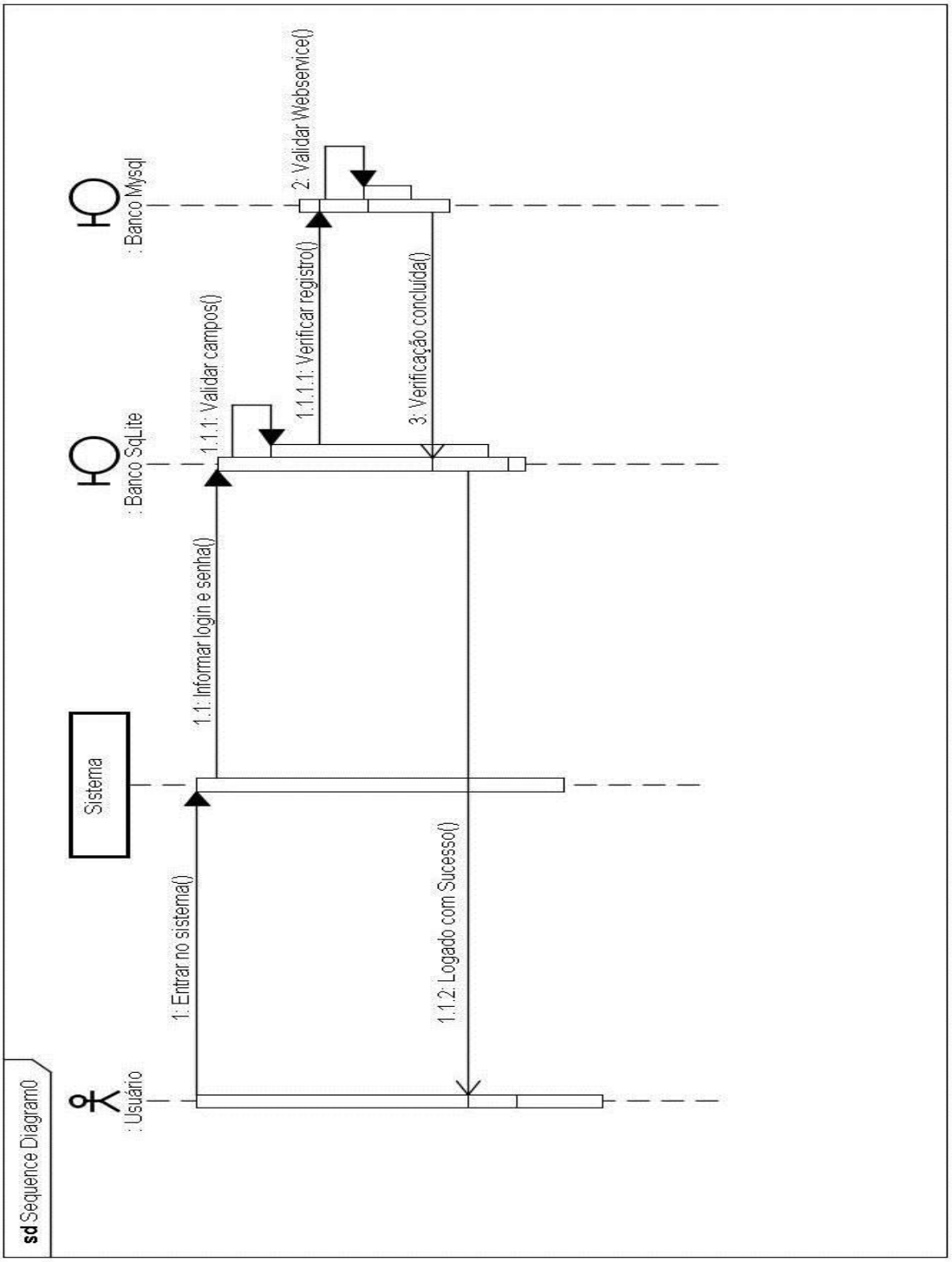

Figura 8- Diagrama de Sequência - Login

6.11.2 Diagrama de Sequência – Cadastrar Usuário

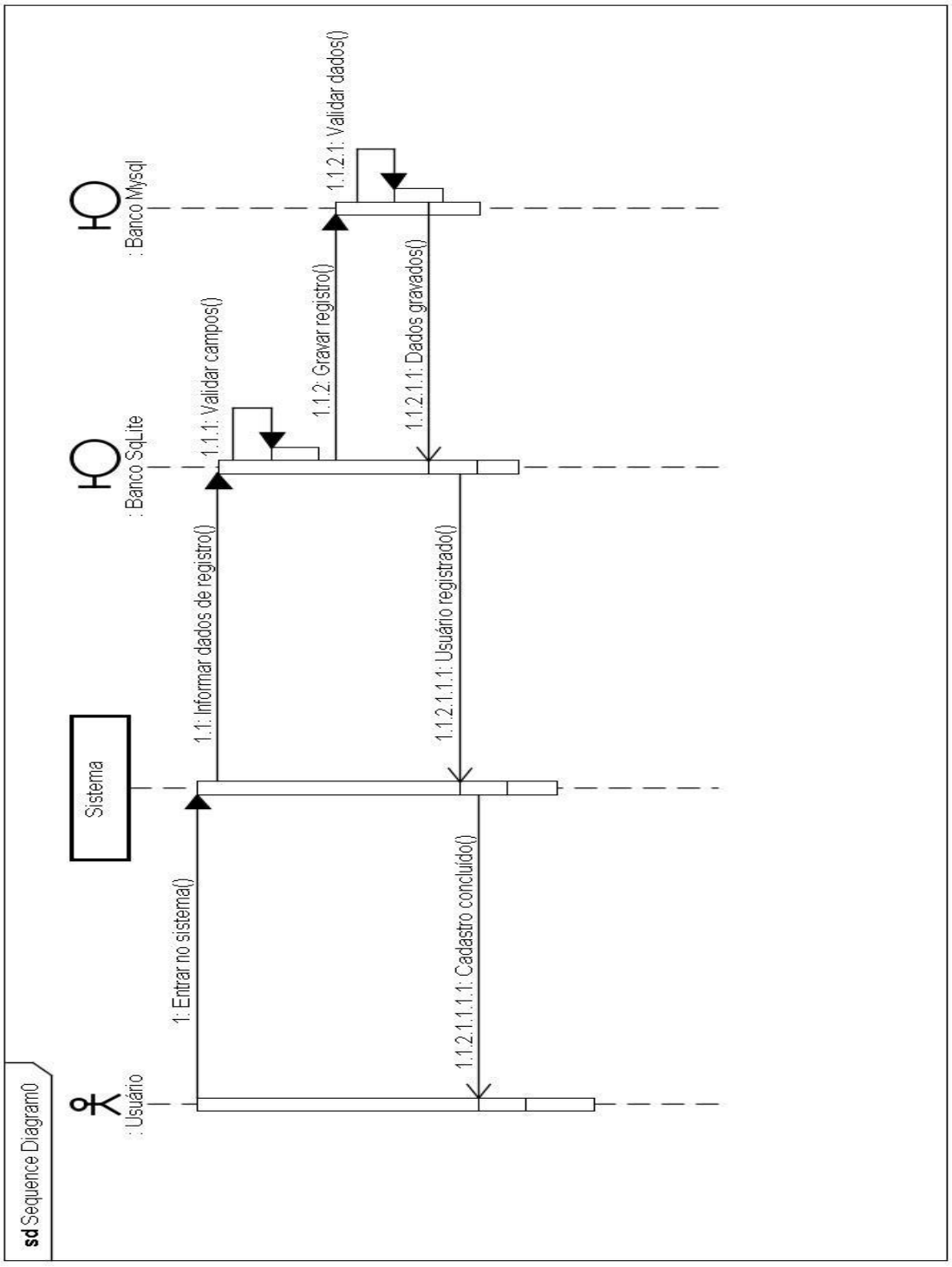

Figura 9- Diagrama de Sequência - Cadastrar Usuário

6.11.3 Diagrama de Sequência – Realizar Pedido

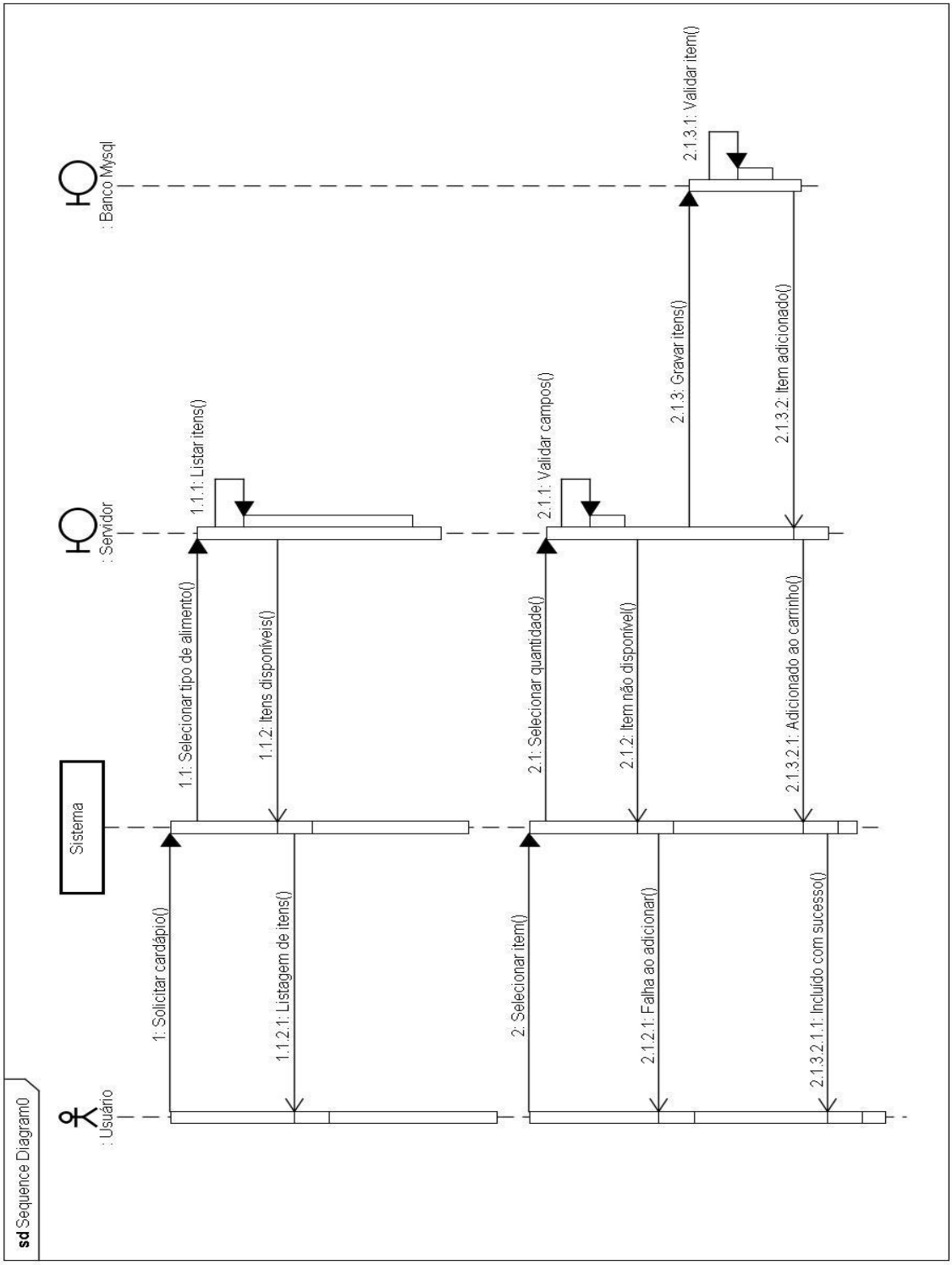

Figura 10 - Diagrama de Sequência - Realizar Pedido

6.11.4 Diagrama de Sequência – Reservar Mesa

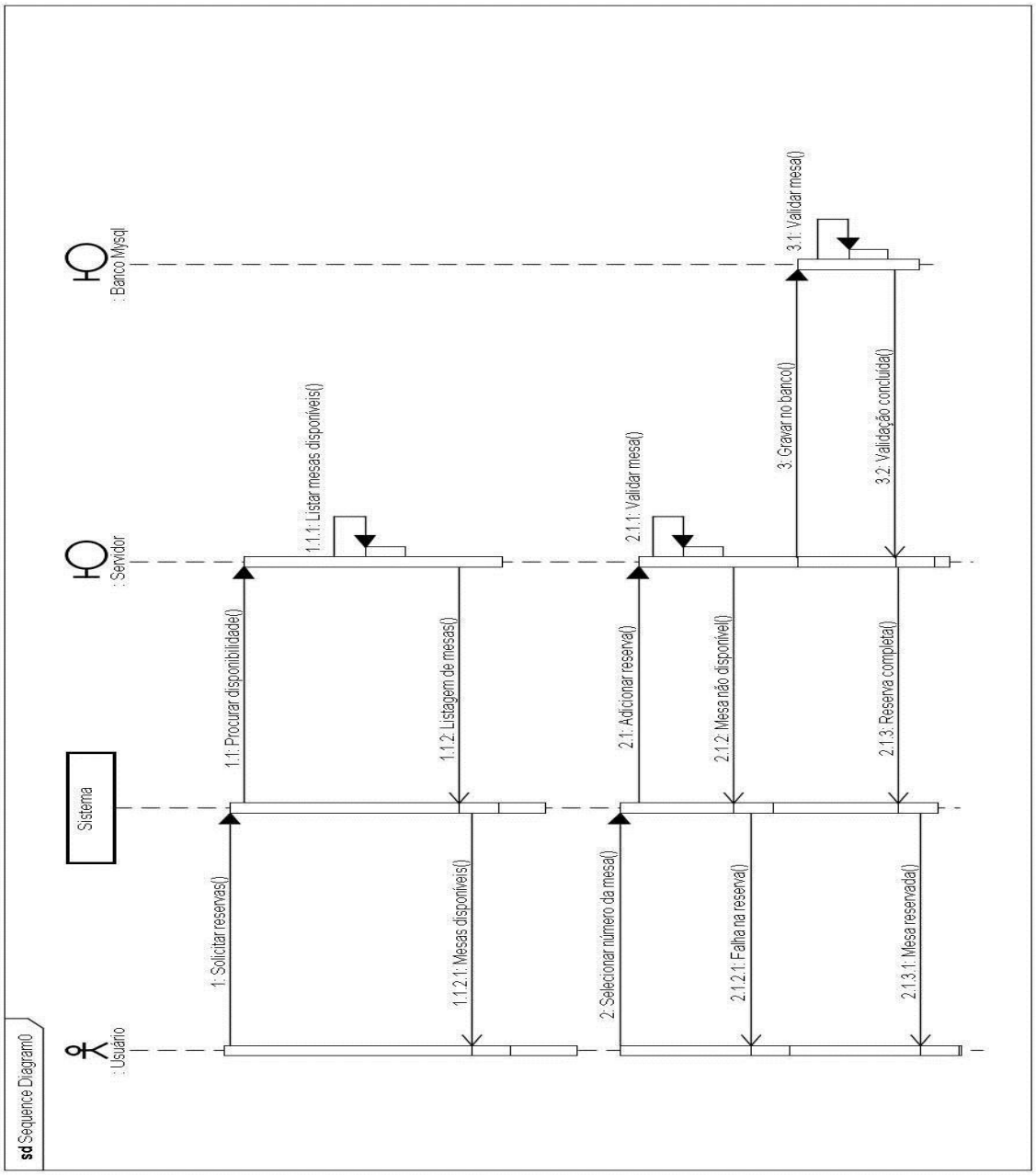

Figura 11 - Diagrama de Sequência – Reservar Mesa

6.11.5 Diagrama de Sequência – Gerenciar Cardápio

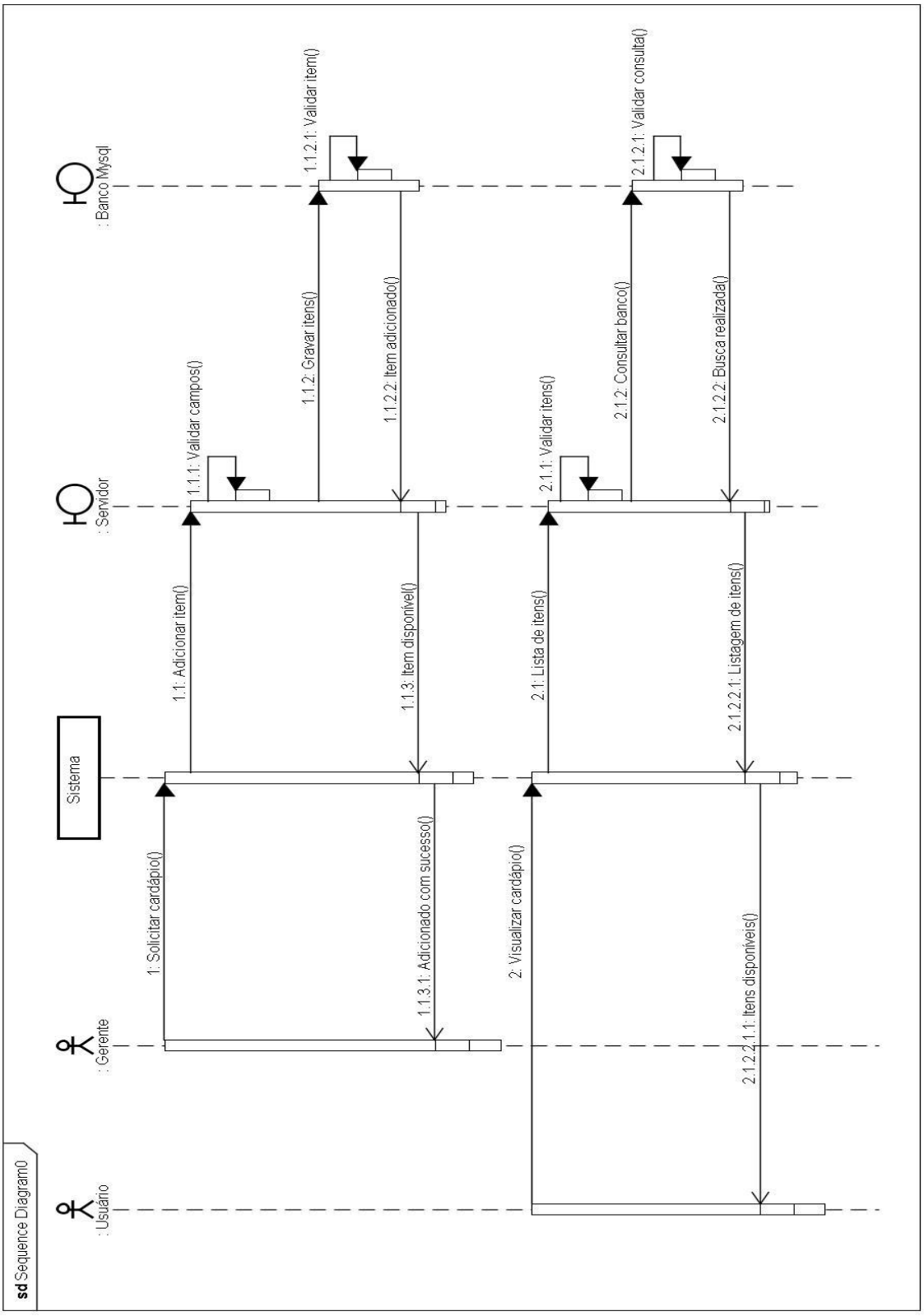

Figura 12 - Diagrama de Sequência – Gerenciar Cardápio

6.11.6 Diagrama de Sequência – Gerenciar eventos

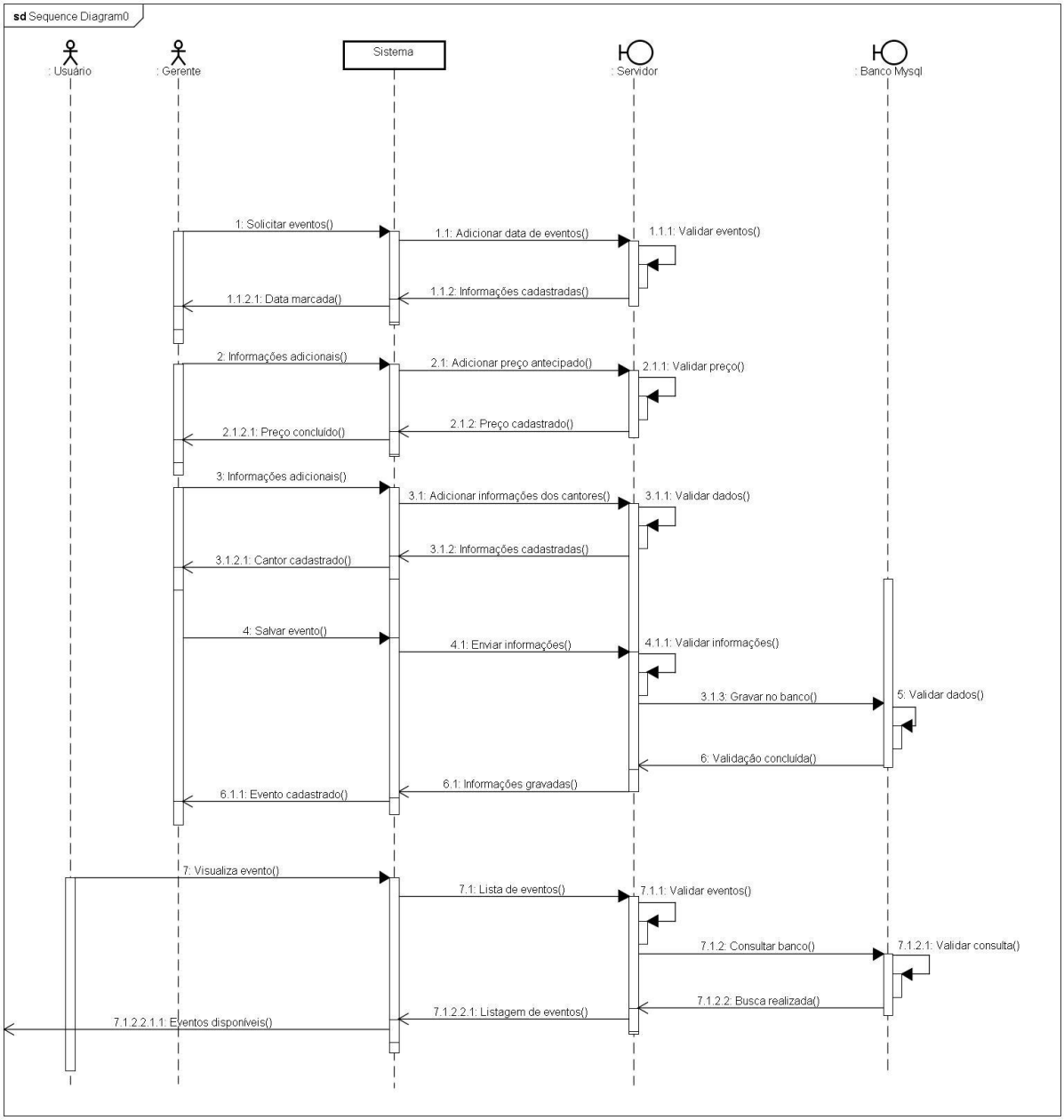

Figura 13 - Diagrama de Sequência – Gerenciar Eventos

7 IMPLEMENTAÇÃO

7.1 Detalhes para implementação (validações).

- Na tela de Cadastro, quando o botão “REGISTRE” é acionado, todos os campos obrigatórios são verificados, caso algum campo não estiver preenchido, ou está preenchido de forma incorreta, o sistema emite uma mensagem sugerindo quais os campos precisam ser preenchidos novamente, depois do preenchimento de todos campos, os dados serão gravados no banco.
- Na tela de Login, quando o botão “LOGIN” é acionado, os campos (Login e Senha) são verificados, caso algum campo não seja preenchido, ou os dados não estejam cadastrados, o sistema emite uma mensagem de erro, somente depois de validar os dados, o sistema disponibilizará acesso ao aplicativo.

7.2 Dificuldades/Facilidades Encontradas

Um grande desafio que tivemos, foi a dificuldade em seguir o cronograma, mantendo os prazos em ordem. Serviu como aprendizado para que não atropelássemos as etapas que são de extrema importância no desenvolvimento. O material de referência sobre a nova plataforma (Android), também foi um ponto crítico, pois o acervo da biblioteca do campus é muito pobre e ainda não possui esse tipo de conteúdo.

Mesmo encontrando dificuldades, não desistimos, pelo contrário, fizemos de nossa dificuldade a motivação para continuar. Com um cliente que já é ligado a tecnologia, foi fácil encontrar os requisitos necessários para conclusão do projeto.

7.3 Testes

Nossa aplicação foi desenvolvida a partir de uma IDE (Android Studio), a maioria dos testes foram feitas na própria IDE utilizando LogCat do Android para realizar depuração no código e mostrar as linhas que continham erros e assim ser possível corrigi-los. A outra parte

dos testes foi feita através de um smartphone Galaxy Gran Duos, para verificar o comportamento no modo físico do aplicativo.

7.4 Descrição de Resultados

Os resultados foram satisfatórios, tivemos um bom desempenho no trabalho em equipe, obtivemos êxito em parte do desenvolvimento, referente a conexão com o banco de dados externo, assim possibilitando o sistema de realizar operações de cadastro de usuários. Além de login e validação de dados no web service. As referências tiveram grande influência nos resultados, pois através delas conseguimos ampliar nosso conhecimento e colocá-lo em prática.

7.5 Ferramentas (softwares, hardware e equipamento utilizado).

- **Astah Professional** – É uma ferramenta CASE de criação de diagramas UML, além de outros diagramas, tais como Diagrama de Caso de Uso, Diagrama de Classe e Diagrama de Sequência, que foram utilizados
- **BrModelo** – É uma ferramenta voltada para modelagem em banco de dados relacional, de fácil utilização, e que foi usada na elaboração do MER (Modelo Entidade e Relacionamento).
- **MySQL Workbench -** É uma ferramenta gráfica para modelagem de dados, integrando criação e designer. A ferramenta possibilita trabalhar diretamente com objetos schema, além de fazer a separação do modelo lógico do catálogo de banco de dados. Foi utilizada na elaboração do MRN (Modelo Relacional Normalizado).
- **PHP - "**Hypertext Preprocessor" é uma linguagem interpretada livre, usada originalmente para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo na Web. Foi utilizado para construção de arquivos WebService para manter conexão entre o aplicativo Android e o Servidor.

- **Mysql** - É um sistema gerenciador de banco de dados relacional de código aberto. O serviço utiliza a linguagem SQL (Structure Query Language – Linguagem de Consulta Estruturada), que é a linguagem mais popular para inserir, acessar e gerenciar o conteúdo armazenado num banco de dados. Foi usado como um banco de dados externo para que houvesse comunicação com o sistema Android.

- **Android Studio** - É um Ambiente de Desenvolvimento Integrado (IDE) criado para facilitar a vida de quem quer desenvolver aplicativos para a plataforma móvel Android. As funções do software incluem a edição inteligente de códigos, recursos para design de interface de usuário e análise de performance, entre outras coisas. Foi utilizado em todo o desenvolvimento e como ferramenta de auxílio para a prototipação das telas.

- **Java JDK** - Java Development Kit (JDK), ou Kit de Desenvolvimento Java, é um conjunto de utilitários que permitem criar sistemas de software para a plataforma Java. É composto por compilador e bibliotecas. A instalação é necessária para que o Android Studio funcione. Foi utilizado como extensão de suporte para o funcionamento do próprio Android

- **CorelDRAW** – É um software para desenho vetorial usado por design gráfico para criação de desenhos, banners, imagens, gráficos, etc. traz diversas funções de edição de imagens, desde a criação digital, gerenciamento de fontes, ilustração, edição, até o desenvolvimento de sites e captura de telas. Foi utilizado para implementar a prototipação de algumas telas que ainda não foram desenvolvidas.

- **O FileZilla** – É um aplicativo de código aberto, recomendado para quem precisa enviar arquivos para algum servidor através do protocolo FTP (File Transfer Protocol). Com interface completamente amigável, torna-se muito mais fácil realizar o upload de arquivos para o servidor. Foi utilizado para fazer upload dos arquivos utilizados no web service juntamente com as imagens dos produtos que serão exibidas dentro do aplicativo.

- **Hostinger** – é uma empresa que oferece serviços de hospedagem web gratuita, com suporte completo a PHP e Mysql, oferece amplo espaço de armazenamento e trafico de dados. Foi utilizado para hospedar arquivos do webservice e imagens necessários para dar suporte as requisições do aplicativo android.

- **Hardware** – Foi utilizado para desenvolvimento um notebook com sistema operacional Windows 10, 6 GB de memória RAM, 1TB de HD e placa de vídeo GeForce 540M.

8 CONSIDERAÇÕES FINAIS

O projeto trata-se de uma aplicação que possibilite ao usuário uma total interação, onde ele possa solicitar o pedido e acompanha-lo. Á medida que o processo é executado, ele é notificado até a finalização do pedido, diminuindo o tempo de espera que o pedido é recebido e transferido para cozinha. O usuário entrará com seus dados cadastrais e terá acesso a todo conteúdo do sistema, entre eles área de perfil, cardápio, reserva de mesas e área de eventos. Visto que a empresa não conseguia ter um controle de anotações de pedidos e endereços de clientes, buscamos desenvolver esta aplicação.

As pesquisas foram a base do projeto, atingimos nossos objetivos através da entrevista, feita diretamente com o cliente, e que nos deu uma visão mais ampla sobre a empresa, o diálogo permitiu que rompêssemos uma barreira que muitas das vezes impedem os desenvolvedores de atingir objetivos na coleta de dados.

A metodologia qualitativa nos ajudou a dar ênfase na subjetividade, ao invés da objetividade. A interação com o cliente foi uma das razões para aperfeiçoarmos nossas técnicas comunicacionais. Pois a partir dela que obtivemos êxito na coleta das informações, assim esclarecendo a situação atual dos problemas.

O projeto foi o resultado de um trabalho árduo somado a todo processo de desenvolvimento, desde o levantamento de requisitos até a implementação.

REFERÊNCIAS BIBLIOGRÁFICAS

DATE, C. J. **Introdução a Sistemas de Banco de Dados.** 8ª ed. Rio de Janeiro: Editora Elsevier, 2003.

GUEDES, Gilleanes T. A. **UML 2: Uma Abordagem Prática.** 2ª ed. São Paulo: Novatec Editora, 2011.

LECHETA, Ricardo R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** 4ª ed. São Paulo: Novatec Editora, 2015.

MILANI, André. **MySQL – Guia do Programador.** 3ª ed. São Paulo: Novatec Editora, 2012.

OGLIARI, Ricardo da Silva; BRITO, Robison Cris. **Android - Do Basico ao Avançado.** 1ª ed. Rio de Janeiro: Editora Ciência Moderna, 2014.

PRESSMAN, Roger S. **Engenharia de Software**. 3ª ed. São Paulo: Pearson Makron Books, 1995.

SILBERSCHATZ, Abraham; KORTH, Henri F.; SUDARSHAN, S. – **Sistema de Banco de Dados.** 4ª ed. Rio de Janeiro: Editora Elsevier, 2006.

SOMMERVILLE, Ian**. Engenharia de Software**. 8ª ed. São Paulo: Pearson Education, 2007.

APÊNDICE

Apêndice A - Entrevista

Empresa: Cabanas Restaurante e Chopperia

Entrevistado: Leomar Ferreira Coelho “Sôla” (Proprietário)

Entrevistadores: Fernando Ferreira Alves e Paulo Rodrigues Viana Júnior

Duração: 40 minutos

Data da entrevista: 20 de Março de 2015

1) A empresa já possui uma aplicação para sua Gestão?

R: Sim, um programa chamado InoveSistemas

2) Quem tem acesso a esse Sistema?

R: Todos funcionários da empresa

3) Este sistema abrange todas as necessidades da empresa?

R: Faz a maioria delas, mas precisamos de um que nos ajude com entregas.

4) Como é feito a anotação dos pedidos?

R: O garçom retira o pedido do cliente, vai ao terminal do sistema e lança o pedido, que é direcionado a cozinha e o bar.

5) Quais são as formas de pagamento aceitas pela empresa?

R: A vista, cartão (débito e crédito), pendura e cheque.

6) Como é feita a reserva de mesa?

R: Pessoalmente ou por telefone.

7) Como é feita a divulgação de shows e eventos?

R: Propagandas de rua, facebook, cartazes e rádio.

8) Já houve tumultos com relação a isso? Falta de comunicação entre cliente garçom ou insatisfação do cliente?

R: Sim, as vezes o cliente reserva uma mesa, mas não cumpre o horário.

9) A empresa já teve a oportunidade de usar algum sistema mobile?

R: Não

10) Com a implantação do aplicativo Android, este manteria algum vínculo com o aplicativo de Gestão?

R: Depende da aceitação da empresa gestora do sistema Inove.

11) Você vê a implantação do aplicativo, como uma maneira de atrair clientes e adquirir mais lucro?

R: Sim, com certeza

12) Como funciona o sistema de pedidos? A empresa realiza a entrega desses Pedidos?

R: Por telefone. Sim

13) Se o cliente optar por receber o pedido em casa, há possibilidade de entrega?

R: Sim.

14) Neste ramo de negócio, você já viu ou ouviu falar de alguma empresa da região que possui seu próprio aplicativo Android, para realizar pedidos online?

R: Não, mas conheço um parecido, chamado entrega10

15) Com a implementação de um sistema onde o cliente utilizará seu smartphone para realizar o pedido, pode ser visto como um diferencial da empresa?

R: Com certeza

16) Você gostaria de ter controle sob os dados de seus cientes, através de cadastro?

R: Sim

17) Com o uso de um aplicativo, que o cliente tem o cardápio na palma da mão, trará algum benefício em relação a disponibilidade dos garçons?

R: Sim, por que as pessoas terão mais facilidade para fazer seus pedidos.

18) Com o uso do aplicativo Android será mais fácil manter o atendimento com mais agilidade e menos risco de erro?

R: Sim, por que as vezes acontecem erros de garçons

19) Na questão de comodidade, o que você acha que seu cliente escolheria. Receber seu pedido em casa ou pegar filas e transito?

R: Receber em casa.

20) Na empresa utiliza rede Wi-Fi? Você vê isso como uma maneira de atrair clientes?

R: Sim, por que todos locais que as pessoas estão, elas querem estar conectadas.

21) Que tal fazer dessa rede uma forma gratuita de distribuir o aplicativo para os clientes?

R: Eu acho uma boa ideia, pois atrairá clientes até a empresa.

Apêndice B - Glossários de Mensagens

Código de Rastreabilidade	**Descrição da Mensagem**	**Caso de Uso**
MSG. 001	Informe o Nome!	1. Gerenciar Conta
MSG. 002	Preencha o Login!	1. Gerenciar Conta
MSG. 003	Preencha a Senha!	1. Gerenciar Conta
MSG. 004	Preencha o CPF!	1. Gerenciar Conta
MSG. 005	Preencha o Email!	1. Gerenciar Conta
MSG. 006	Preencha o Telefone!	1. Gerenciar Conta
MSG. 007	Preencha o Endereço!	1. Gerenciar Conta
MSG. 008	Email Inválido!	1. Gerenciar Conta
MSG. 009	CPF Inválido!	1. Gerenciar Conta
MSG. 0010	Dados Inválidos!	1. Gerenciar Conta
MSG. 0011	Falha ao realizar cadastro! Conta já existente.	1. Gerenciar Conta

MSG. 0012	Usuário Cadastrado com Sucesso!	1. Gerenciar Conta 2. Manter Usuário
MSG. 0013	Logando...	1. Gerenciar Conta
MSG. 0014	Login ou Senha Incorretos!	1. Gerenciar Conta
MSG. 0015	Esqueceu sua Senha?	1. Gerenciar Conta
MSG. 0016	Logado com Sucesso!	1. Gerenciar Conta
MSG. 0017	Alterado com Sucesso!	1. Gerenciar Conta 2. Manter Usuário
MSG. 0018	Selecione uma opção de Pagamento!	3. Gerenciar Pedido
MSG. 0019	Selecione uma opção de Entrega!	3. Gerenciar Pedido
MSG. 0020	Pedido Concluído!	3. Gerenciar Pedido
MSG. 0021	Pedido encaminhado para Preparo!	3. Gerenciar Pedido
MSG. 0022	Mesas Reservadas!	4. Gerenciar Mesa
MSG. 0023	Evento adicionado!	5. Gerenciar Evento

MSG. 0024	Item adicionado!	6. Gerenciar Cardápio

Tabela 8 - Glossário de Mensagens

Apêndice C – Prototipação

Nossos Protótipos foram criados, utilizando a própria IDE de desenvolvimento oficial do Google (Android Studio) para as partes já implementadas e funcionais. Para as demais partes foram utilizadas um programa de ilustração vetorial (Corel DRAW X7).

Figura 14 – Tela disparada quando o aplicativo é iniciado, possui animação simples de rotação, tem por objetivo prender a atenção do usuário enquanto o aplicativo é carregado.

Figura 14 - Splash Screen

Figura 15 – Tela inicial do aplicativo logo após carregamento da Splash Screen, nessa tela o usuário deve entrar com seu login e senha cadastrados para ter acesso à outras funções do aplicativo após validação de seus dados preenchidos.

Figura 15 - Tela de Login

Figura 16 – Tela para cadastro onde o usuário deve informar alguns dados pessoais, preenchendo todos os campos para validação e em seguida gravação no Banco de Dados do Servidor Online.

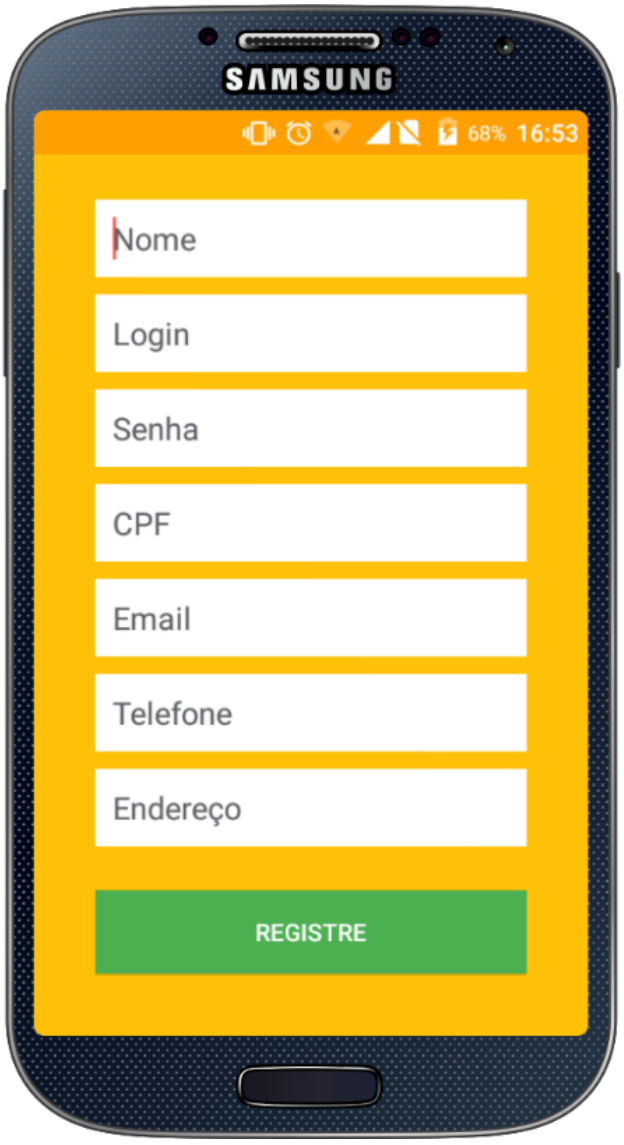

Figura 16 - Tela Cadastro de Usuário

Figura 17 – Tela para apresentação das principais funcionalidades do aplicativo, tela capaz de fornecer acesso a outras telas especificas e tornar mais prático e acessível ao usuário.

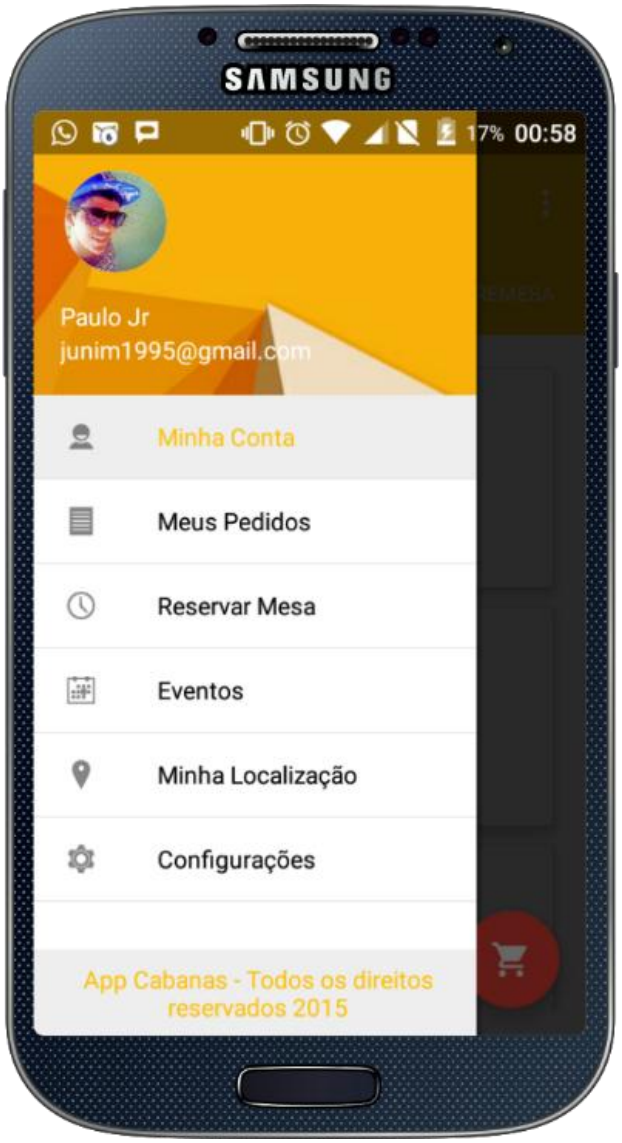

Figura 17 - Tela de Menu

Figura 18 – Tela que será apresentado um ListView com imagem e nome de todos os produtos contidos na categoria Comidas.

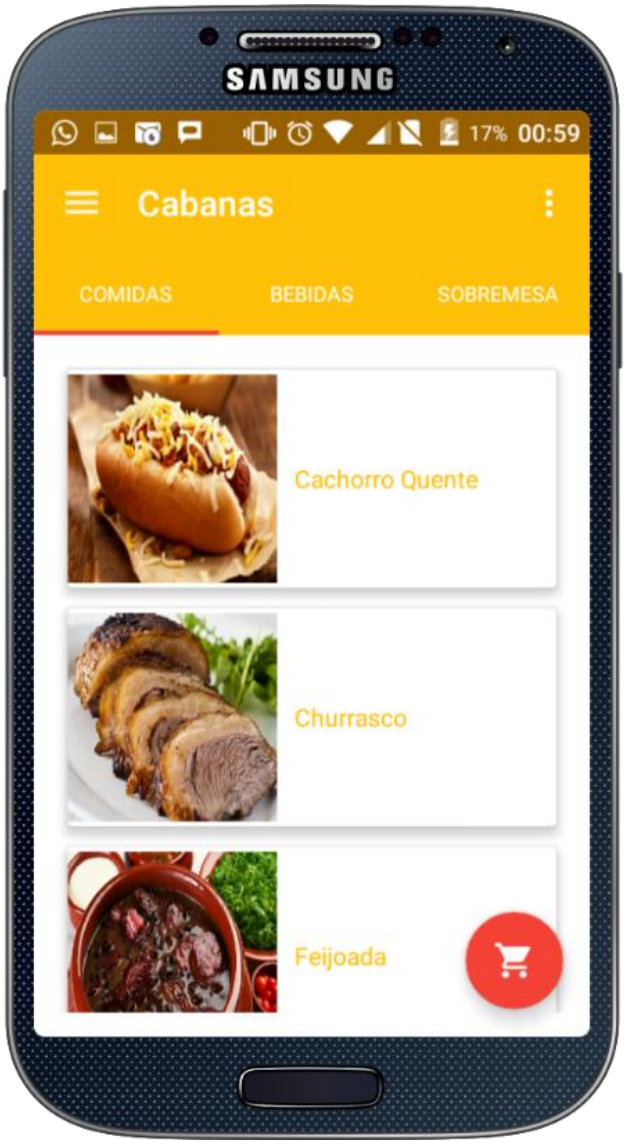

Figura 18 - Tela Cardápio Comidas

Figura 19 – Tela que será apresentado um ListView com imagem e nome de todos os produtos contidos na categoria Bebidas.

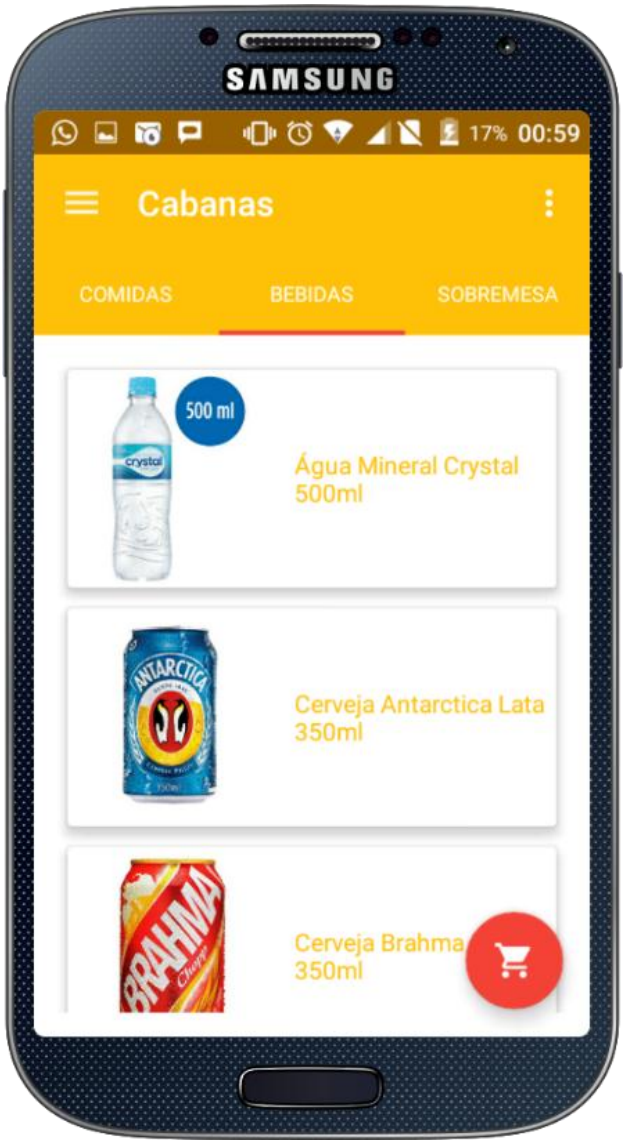

Figura 19 - Tela Cardápio Bebidas

Figura 20 – Tela que será apresentado um ListView com imagem e nome de todos os produtos contidos na categoria Sobremesas.

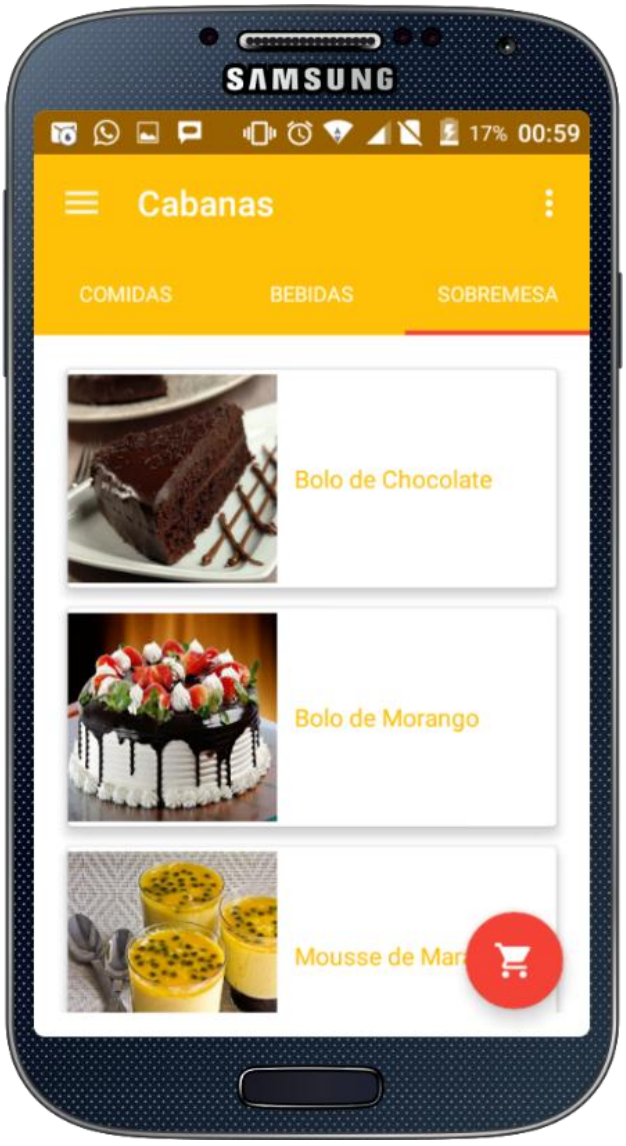

Figura 20 - Tela Cardápio Sobremesas

Figura 21 – Após o usuário clicar no produto escolhido será redirecionado para essa tela, para apresentação dos detalhes do produto (imagem ampliada, descrição, preço). Nesta tela há a opção de incluir no pedido.

Figura 21 - Tela Detalhes do Produto

Figura 22 – Tela de Detalhes dos Produtos, mostra a seção abaixo da tela anterior com as informações sobre código do produto, preço unitário, tempo de preparo e possui a opção de incluir no pedido com a quantidade informada pelo usuário.

Figura 22 - Tela Detalhes do Produto - Seção Incluir no Pedido

Figura 23 – Tela para visualização de todos os produtos que foram incluídos no pedido, nesta tela o usuário pode gerenciar seus pedidos: atualizar quantidade, excluir produto, visualizar total e prosseguir para finalização do pedido.

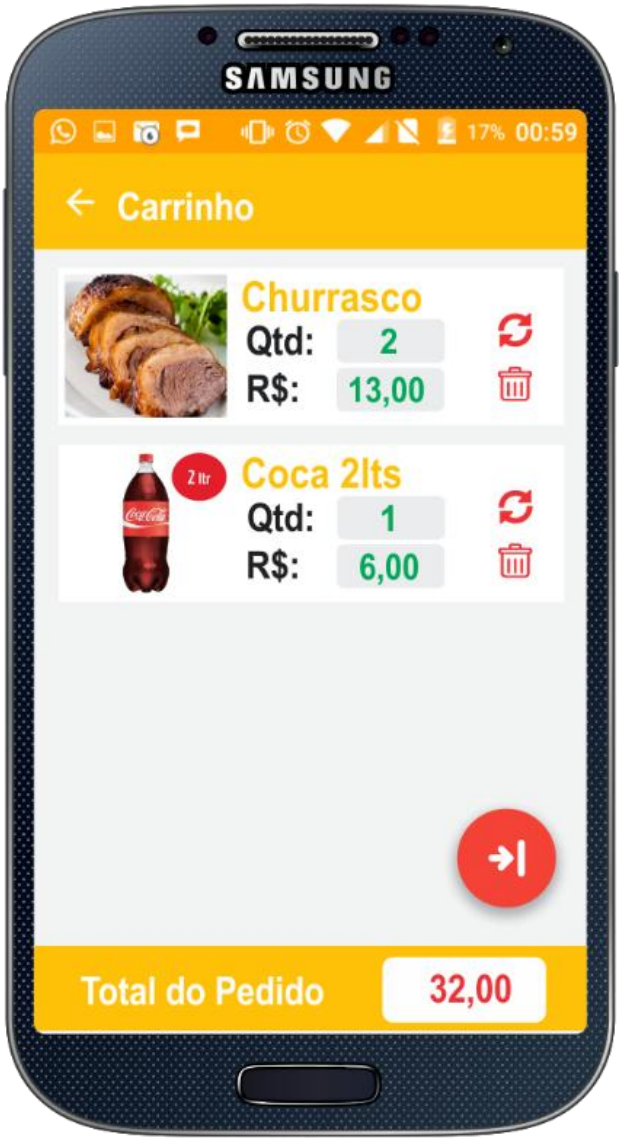

Figura 23 - Tela Carrinho de Pedidos

Figura 24 – Tela para finalização de pedido, o usuário deve escolher forma de pagamento e entrega. Na seção entrega o usuário deve escolher entre entregar no endereço cadastrado no banco de dados, ou caso esteja no estabelecimento deve informar a mesa.

Figura 24 - Tela Finalizar Pedido

Figura 25 – Tela para visualização da localização do estabelecimento em relação a localização do usuário, traçando a rota para chegar ao ponto fixo do estabelecimento.

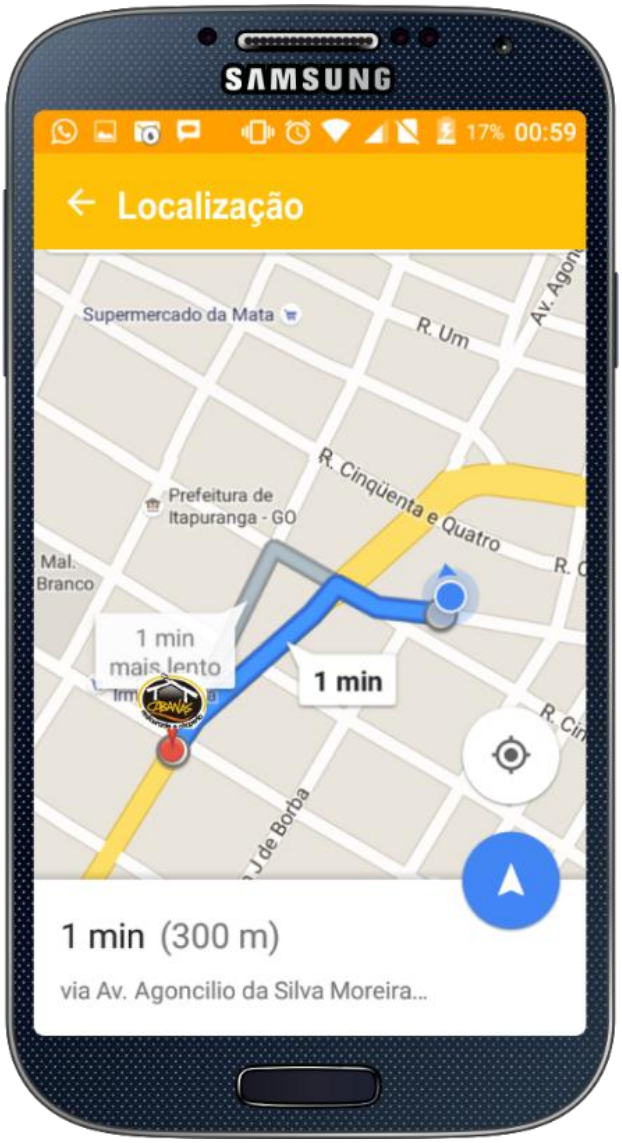

Figura 25 - Tela de Localização

Figura 26 – Tela para informar ao usuário algumas notificações importantes a respeito do preparo ou entrega de seus pedidos.

Figura 26 - Tela de Notificação

Figura 27 – Tela de apresentação do aplicativo, informações sobre versão, desenvolvedores, breve descrição, etc.

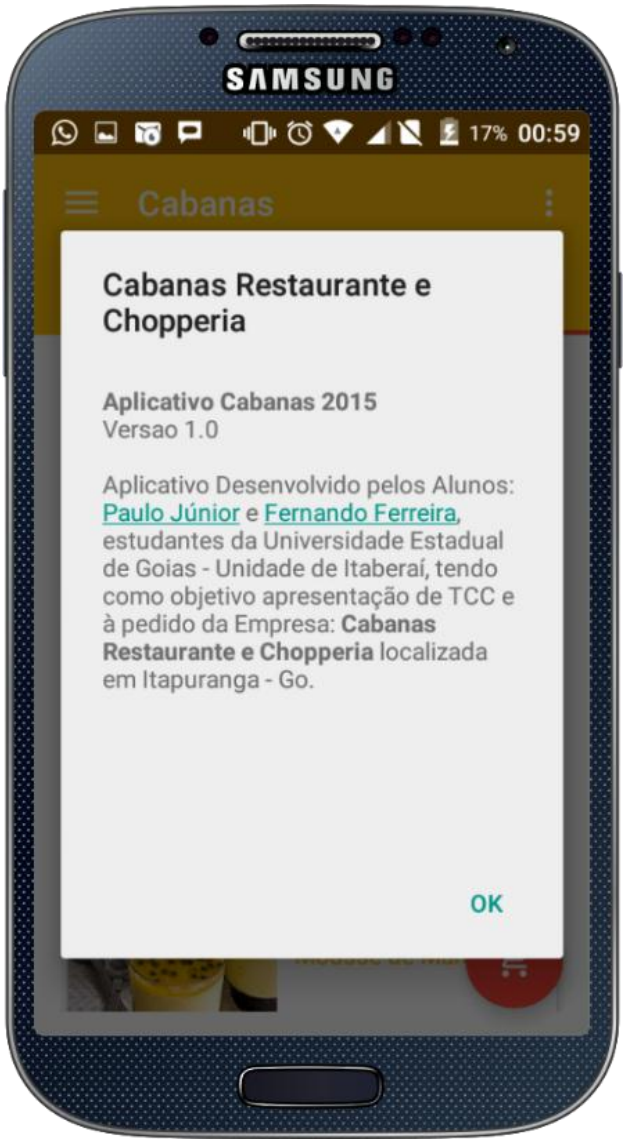

Figura 27 - Tela Sobre o Aplicativo

ANEXO

Anexo A -Bloco De Anotações De Pedidos

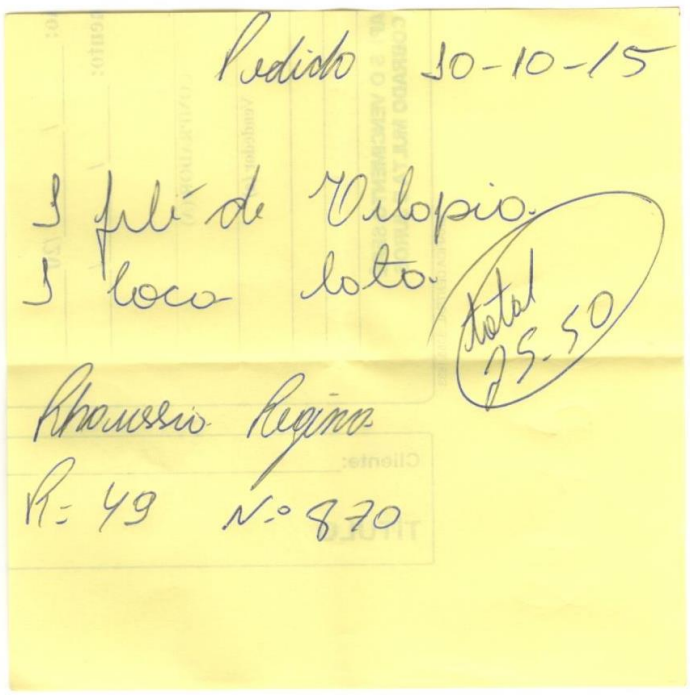

Figura 28 - Bloco de Anotação de Pedidos

Anexo B - Comprovante de Consumo Cliente

Conferencia Consumo
Mesa. : 00009
Operador...: Leomar Coelho
Data.....: 10/11/2015
Abertura...: 14:01:11 Termino: 14:38:28
Permanencia: 00:37:16

PRODUTO	V. UNIT	QTD	TOTAL
REFEICAO SEG A SEX	25,90	0,6	15,64

Total S/ Desconto 15,64

Total R$ 15,64

i9 Sistemas & Automacao
Fone:(62) 3093-0180
SERVIDOR

Figura 29- Comprovante de Consumo Cliente

Anexo C - Cupom Fiscal de Compra

CABANAS RESTAURANTE & CHOPERIA
CABANAS RESTAURANTE & CHOPERIA LTDA. = RUA 48
S/N QD.L LT.16 ST.CENTRO = ITAPURANGA-GO
CNPJ:10.837.439/0001-27
IE:10.447.346-0
10/11/2015 15:47:36V CCF:025869 COO:027518

CUPOM FISCAL

ITEM	CÓDIGO	DESCRIÇÃO	QTD.UN.VL UNIT(R$)	ST	VL ITEM(R$)
1	800	REFEICAO SEG A SEX	0 7UNx25,90	TT	18,13G
2	240	SUCO DE LARANJA	2UNx3,50	T1	7,00G

TOTAL R$ 25,13
CARTAO DEBITO 25,13
T1=01T17,00%

* www.i9sistemas.com *
1GUFDRLF GQUF6ACW 3IXV7O+E 1ENACJJ8 GKRB148U2HON
BEMATECH MP-4000 TH FI ECF-IF
VERSÃO:01.00.02 ECF:001 LJ:0001
QQQQQQQQQUIRTOEQT 10/11/2015 15:47:38V
FAB:BE09111010001127O634 BR

Figura 30 - Cupom Fiscal de Compra