UNIVERSIDADE DE CAXIAS DO SUL

GEOVANI MACIEL SCHNEIDER

SISTEMA DE AUTOATENDIMENTO E GERENCIAMENTO DE PEDIDOS NA NUVEM

CAXIAS DO SUL

2012

UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE COMPUTAÇÃO E TECNOLOGIA DA INFORMAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GEOVANI MACIEL SCHNEIDER

SISTEMA DE AUTOATENDIMENTO E GERENCIAMENTO DE PEDIDOS NA NUVEM

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Sistemas de Informação da Universidade de Caxias do Sul. Orientador Prof. Ms**.** André Zampieri

CAXIAS DO SUL

2012

RESUMO

Este trabalho propõe uma solução de autoatendimento para estabelecimentos do setor gastronômico, permitindo uma maior independência dos clientes ao frequentá-los. Terminais de autoatendimento em bancos e aeroportos já fazem parte de um cotidiano onde o tempo, muitas vezes gasto em tarefas rotineiras, é cada vez mais valioso. A tecnologia vem de várias maneiras buscando automatizar diversos processos, na intenção de deixar para as pessoas apenas as tarefas que necessitem de uma interpretação pessoal. Neste trabalho são realizados estudos de alternativas para a elaboração de um sistema que permita ao cliente de um restaurante a utilização do seu *smartphone* para realizar seu próprio atendimento, sem a necessidade de um profissional para atendê-lo. Os estudos resultam em uma solução que utiliza serviços de processamento na nuvem para gerenciamento dos dados, uma interface de controle dos pedidos utilizada pelos estabelecimentos e um aplicativo para o sistema operacional Android, utilizado pelos clientes.

Palavras-chaves: Computação na Nuvem. Android. Autoatendimento.

ABSTRACT

This work proposes a self-service solution for gastronomic establishments, allowing greater independence to clients that frequent them. ATMs in banks and airports are already part of a daily routine where the time, often spent on routine tasks, is increasingly valuable. The technology comes in many ways seeking to automate various processes, with the intention of leave people do only the tasks that require a personal interpretation. This work brings studies of alternatives for the development of a system that allows the client to use his smartphone to perform his own request, without any assist from a professional. The studies resulted in a solution that uses cloud computing for data management, an interface to control the requests, used by the establishments, and an application for the Android operating system, used by customers.

Keywords: Cloud Computing. Android. Self-service.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BYOD	Bring Your Own Device
GAP	Google App Engine
IDC	International Data Corporation
PAAS	Platform As A Service
SAAS	Software As A Service
SDK	Software Development Kit
URL	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, com uma rotina cada vez mais apressada, é comum que o cliente, ao frequentar algum estabelecimento, queira um atendimento rápido e eficiente. A tecnologia vem agindo no setor de *self-service* a fim de fazer com que o cliente tenha independência para obter o que deseja, sem a necessidade de um atendente exclusivo.

Para o estabelecimento comercial, a técnica de autoatendimento também traz vantagens como a agilidade na realização do pedido, evitando uma possível insatisfação do cliente e a redução de custos, uma vez que não é necessário alocar um funcionário para realizar o atendimento.

Um exemplo de sucesso de autoatendimento são os caixas eletrônicos das instituições bancárias, utilizados em grande escala ao redor do mundo. Para Kotler (1995), as denominadas máquinas de venda automática oferecem a vantagem da ausência da manipulação por terceiros, reduzindo o custo com funcionários, e a disponibilidade do produto a qualquer momento, desde que o estabelecimento assim as configure. Outros exemplos desse tipo de autoatendimento são máquinas para a compra de ingressos nos cinemas, *check-in* em aeroportos e até mesmo para a compra de bebidas e lanches que não necessitam de uma preparação manual.

Uma alternativa para essas máquinas de autoatendimento são os dispositivos móveis pessoais presentes no nosso dia a dia, seguindo a tendência do *BYOD (Bring Your Own Device)*, também conhecida como consumerização de TI, que está em ascendência principalmente no ambiente corporativo. Conforme dados apontados em uma pesquisa (WAKEFIELD RESEARCH, 2012), 88% dos profissionais das organizações consultadas utilizam seus dispositivos no ambiente de trabalho e 73% dos gestores apontam como prioridade na empresa os esforços para o crescimento na utilização desta técnica. As principais vantagens citadas são a satisfação do funcionário e a melhoria na produtividade, pois o usuário conhece plenamente as funcionalidades e características do dispositivo e pode mantê-lo configurado da sua maneira. Sem a necessidade de fornecer um dispositivo exclusivo para cada usuário, a empresa tem uma redução no valor do investimento em ferramentas de trabalho e permite que o usuário se sinta mais a vontade e confortável com o próprio equipamento.

No setor gastronômico um sistema de autoatendimento pode ser muito vantajoso, pois ainda há uma grande dependência nos profissionais responsáveis por atender o cliente, registrar

suas solicitações e repassar a demanda para o departamento que irá preparar o pedido. Essa dependência pode causar certa demora no atendimento e consequentemente gerar uma insatisfação do cliente com o serviço oferecido. Considerando esta rotina, o segmento de bares, restaurantes e lancherias passa a ser um candidato ideal a uma reformulação, introduzindo o autoatendimento nos seus processos e trazendo benefícios tanto para o consumidor quanto para quem oferece o produto ou serviço.

Utilizando uma ferramenta de autoatendimento, também é possível disponibilizar ao cliente informações importantes como a ocupação atual do estabelecimento, prévia do cardápio e consulta de avaliações realizadas por usuários que já frequentaram determinado estabelecimento. Estas informações, aliadas a funcionalidades como a reserva de mesas pelo próprio aplicativo, auxiliam o cliente na tomada de decisão e tornam mais cômodo o momento da sua refeição, permitindo uma fidelização com o estabelecimento que oferece tais diferenciais.

Este trabalho propõe uma solução tecnológica que permite, aos clientes de estabelecimentos do setor gastronômico, utilizar seu próprio dispositivo móvel na intenção de obter um atendimento mais ágil e eficiente. Serão abordados diversos temas com o objetivo de desenvolver uma aplicação com reduzido custo de implantação para o estabelecimento, considerando a utilização de computação na nuvem, e possibilitando uma fidelização do estabelecimento com os clientes, através de funcionalidades e informações disponibilizadas na solução.

Para atingir os objetivos, serão estudadas, e apresentadas no decorrer do texto, as soluções existentes atualmente que se assemelham à solução proposta e os estudos técnicos sobre as tecnologias de dispositivos móveis, computação na nuvem e comunicação entre as interfaces. Em seguida será demonstrada a estrutura da solução e as aplicações que a compõe. Após os estudos será realizada a etapa de desenvolvimento e de testes funcionais com usuários e estabelecimentos. Ao final, projeta-se a captação de depoimentos e avaliações dos usuários sobre sua experiência na utilização das aplicações.

1.1 SOLUÇÕES EXISTENTES

No mercado, há atualmente algumas ferramentas com características semelhantes às previstas para este projeto, incluindo informações sobre cardápio, reservas ou até mesmo o autoatendimento no estabelecimento. A seguir serão apresentadas algumas das ferramentas mais utilizadas neste segmento.

1.1.1 Restorando

O Restorando (RESTORANDO, 2012), lançado em 2010, é uma ferramenta que oferece o serviço de reserva em restaurantes, conforme ilustram as telas capturadas do aplicativo e apresentadas na Figura 1. Disponível para diversos estabelecimentos situados nas principais capitais do país, conta também com estabelecimentos conveniados na Argentina e no Chile.

Para realizar a reserva, é necessário criar um usuário e logar-se no site ou no aplicativo disponível apenas para dispositivos com iOS. A escolha do estabelecimento pode ser feita após definir a cidade desejada, permitindo uma seleção por bairros ou pela especialidade do cardápio. Com o local definido, o cliente informa o horário da sua reserva e o número de pessoas. Em seguida o estabelecimento é informado dessa escolha, finalizando o processo de reserva. Na ferramenta também são disponibilizadas fotos do local e um espaço para comentários de clientes.

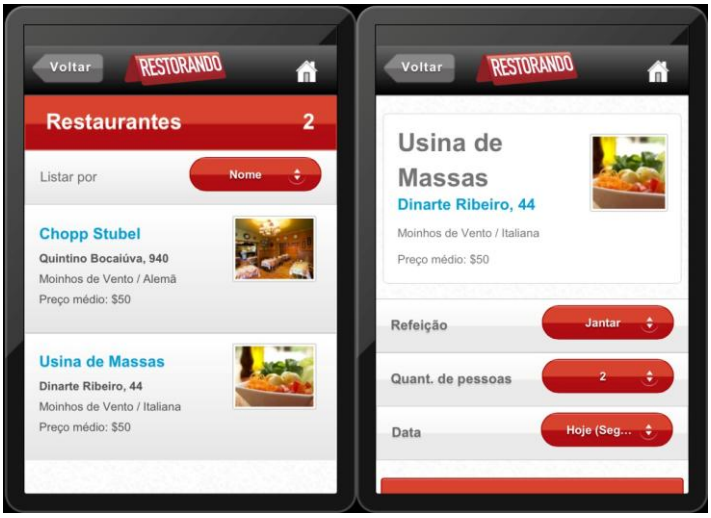

Figura 1 - Telas do aplicativo Restorando
Fonte: Autor

1.1.2 iFood

Especializado em tele-entrega, a ferramenta iFood (IFOOD, 2012) possui uma base de dados com o cardápio de vários estabelecimentos das maiores cidades do país. Utilizando o site, o aplicativo Android ou o aplicativo para iOS, é possível navegar entre os vários pratos oferecidos, visualizando informações de valor, ingredientes, tempo de preparo e avaliação do estabelecimento por outros clientes.

Após a escolha dos pratos é possível realizar um pedido para tele-entrega, seguindo para uma área exclusiva de usuários cadastrados. Em seguida o pedido é enviado ao estabelecimento, junto com o endereço do cliente. Alguns estabelecimentos não possuem os dados completos, estando disponível apenas uma imagem do cardápio para consulta e não sendo possível também a realização de pedidos diretamente pela ferramenta. As informações dos cardápios estão disponíveis apenas se o cliente estiver conectado à internet, não permitindo realizar qualquer consulta em modo *offline*.

Na Figura 2 são apresentadas algumas telas do aplicativo, desde o início com a informação da localização, passando pela lista de estabelecimentos e chegando à tela de escolha do prato desejado.

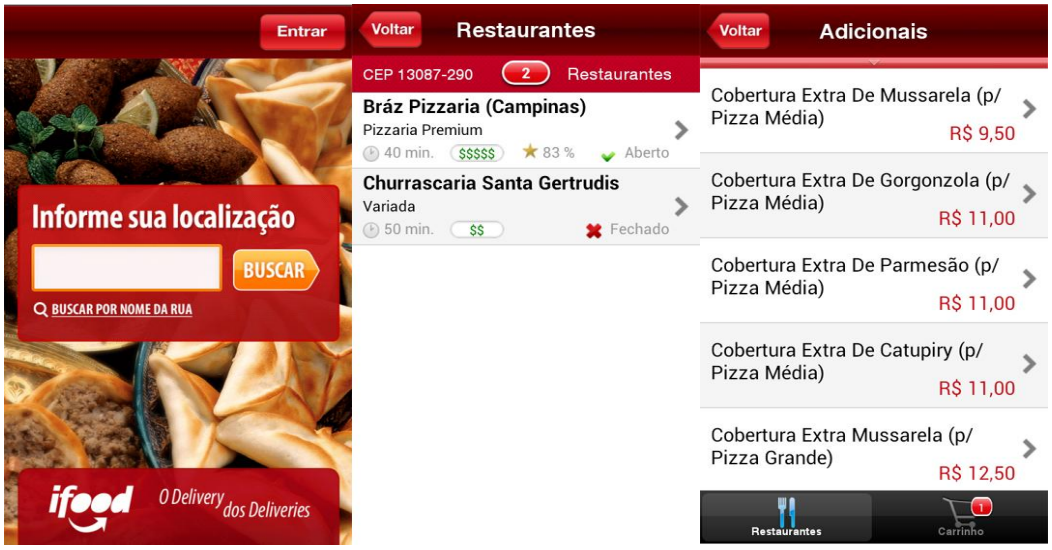

Figura 2 - Telas do aplicativo iFood
Fonte: Autor

1.1.3 Tabber

Na pesquisa realizada, encontrou-se apenas uma ferramenta que possui como objetivo central a possibilidade de realizar pedidos diretamente do *smartphone* do cliente com a proposta de autoatendimento. Recém saído da fase de testes, atualmente o Tabber está sendo utilizado apenas em alguns bares na região da cidade de São Paulo (TABBER, 2012). Para acesso à ferramenta foram disponibilizados um site, um *web app*[1] para Android e um *web app* para o sistema operacional iOS. Por ser totalmente online, a ferramenta não permite que o cliente tenha acesso aos cardápios registrados se o dispositivo não estiver conectado na internet.

A ferramenta é simples porém intuitiva, com a opção de chamar o garçom ou incluir itens no pedido em aberto. Cada item solicitado pode conter observações das preferências do cliente. Enquanto o pedido está em aberto, é possível ver o valor gasto até o momento e solicitar a conta, que neste caso, é providenciada pelo garçom. A ferramenta não substitui o sistema de ERP do restaurante, pois não possui recursos e controles mais avançados. Seu objetivo é apenas agilizar o processo de atendimento.

A solução apresenta uma característica que deixa o estabelecimento bastante vulnerável a usuários mal intencionados. Qualquer pessoa com acesso à internet pode acessar a ferramenta, selecionar um estabelecimento e incluir pedidos sem qualquer tipo de verificação ou controle. Todos os pedidos gerados seriam entregues na mesa informada, mas o cliente pode não estar naquela mesa ou nem ao menos estar no local, o que geraria um custo para o estabelecimento, pois o pedido teria sido produzido em vão.

Na Figura 3 estão representadas duas telas capturadas da ferramenta, uma no momento do *login* que é obrigatório e outra tela no momento da inserção de um item ao pedido.

[1] Web app: Aplicativo atalho para uma determinada página de internet. O conteúdo e o layout da página são interpretados, assim como em um browser, mas dando a impressão para o usuário que o processamento é local. Tem-se a vantagem de uma padronização de layout independente do sistema operacional utilizado.

Figura 3 - Telas do aplicativo Tabber

Fonte: Autor

2 ESTUDOS TÉCNICOS

Para disponibilizar uma solução ao problema proposto, existe a opção de um site onde seria possível realizar o pedido, assim tem-se o desenvolvimento de apenas um produto que irá abranger todos os sistemas operacionais, visto que todos têm suporte a *websites*. Em contrapartida, criando um aplicativo específico, é possível adaptar os botões e atalhos para uma melhor usabilidade em dispositivos móveis que possuem uma tela menor. Neste projeto, estão previstas funcionalidades para utilização *offline*, ou seja, algumas informações disponíveis sem que o cliente precise estar conectado à internet. Isto não seria possível com um site, restando a opção de desenvolver um aplicativo para ser instalado nos dispositivos móveis.

A desvantagem de desenvolver um aplicativo é a incompatibilidade entre os sistemas operacionais, sendo necessário duplicar o código e criar um aplicativo específico para cada sistema operacional. A interface e as funcionalidades da aplicação devem permanecer idênticas, independente do sistema operacional.

2.1 SISTEMA OPERACIONAL MÓVEL

Atualmente o mercado de *smartphones* é caracterizado por duas grandes plataformas: iOS e Android. Segundo uma pesquisa realizada pela IDC (2012), o Android aparece em primeiro lugar em *market share* de sistemas operacionais para *smartphones* com de 68,1% de participação. Em segundo lugar, o iOS conta com 16,9% de *market share*.

2.1.1 iOS

Em 2007 a Apple revolucionou o mercado de telefones móveis anunciando o iPhone, um telefone com tela *touchscreen*, dispondo de apenas um botão frontal e com a possibilidade de instalação de aplicativos desenvolvidos por terceiros. A Apple é até hoje a responsável pelo desenvolvimento do sistema operacional do iPhone, o iOS, e também responsável pela estrutura física, garantindo que *software* e *hardware* não enfrentem nenhum tipo de incompatibilidade e assegurando uma padronização em todas as versões.

Essa dominância da Apple sobre seus produtos traz algumas desvantagens para os utilizadores, pois cada funcionalidade é extremamente controlada e algumas até restritas, diferente dos seus concorrentes onde há maior liberdade na utilização das funcionalidades do produto.

Os aplicativos desenvolvidos por terceiros são verificados antes de serem liberados na App Store, loja de aplicativos do iOS. Isso garante uma maior segurança nos aplicativos, evitando códigos maliciosos, mas pode se tornar um incômodo para o desenvolvedor, atrasando o lançamento ou até mesmo impedindo o aplicativo de ser publicado e disponibilizado ao restante dos usuários. As ferramentas de desenvolvimento para o iOS estão disponíveis apenas para computadores da linha MAC, proprietária da Apple, sendo assim, não é possível desenvolver aplicativos utilizando o sistema operacional Windows ou Linux.

Em geral, os dispositivos com Android tem um custo menor que o iPhone, isto faz com que o iOS não seja o líder do mercado neste segmento. Esta característica pode ser uma estratégia de mercado, mas deixa o iOS em segundo lugar no número de usuários.

2.1.2 Android

A empresa Android Inc. e seu produto, o sistema operacional Android, foram comprados pelo Google em 2005. A premissa do Android era ser um sistema operacional móvel que pudesse ser utilizado por vários fabricantes de *smartphones* e customizável por cada um destes (HASHIMI; KOMATINENI; MACLEAN, 2010), ao contrário do iOS.

O Google, junto com fabricantes de celulares e operadoras de telefonia, criou o grupo OHA (Open Handset Alliance), com o objetivo de controlar as atualizações realizadas no Android e manter a compatibilidade entre diversos dispositivos e as redes móveis. Em 2008 foi anunciado o primeiro telefone móvel com o sistema operacional Android, junto com o SDK para desenvolvedores que pode ser utilizado em vários sistemas operacionais para PCs, como Windows, Linux e até mesmo Mac OS.

O sistema operacional móvel do Google possui o código fonte *open source* sob a licença Apache, que institui a liberdade de utilização e alteração do código fonte (LEE, 2011), não sendo necessário aos fabricantes o pagamento de direitos autorais ao utilizarem o Android nos seus aparelhos. Por ser um sistema de código aberto, o Android é altamente utilizado em diversos

outros dispositivos e estudos acadêmicos. O Android também incentiva desenvolvedores a criarem aplicativos para sua plataforma, permitindo um desenvolvimento sem custos, ao contrário do iOS, onde é cobrada uma taxa para desenvolvedores.

2.2 COMPUTAÇÃO NA NUVEM

Velte, Velte e Elsenpeter (2010) definem o termo *Cloud Computing* como uma estrutura que permite ao usuário acessar do seu próprio computador ou de qualquer outro dispositivo conectado à internet um mesmo aplicativo que, na maior parte dos casos, está armazenado em um *datacenter* distante do usuário que solicita a conexão. Miller (2009) complementa que a nuvem é constituída de vários servidores separados geograficamente mas interconectados pela internet. Para o usuário, essa estrutura é invisível, pois o método de acesso aos serviços na nuvem é único, independente do servidor em que ele esteja armazenado.

Todas as aplicações armazenadas em determinado serviço de computação na nuvem dispõem de um mesmo servidor ou grupo de servidores, permitindo uma otimização física e redução nos custos. Parte dos custos de um *datacenter* é referente a questões físicas como energia elétrica, substituição de equipamentos danificados e segurança do ambiente. Como na estrutura de computação na nuvem inúmeras aplicações utilizam os mesmos recursos, os custos para quem está mantendo os servidores podem ser distribuídos entre todos os clientes, resultando em um valor menor para cada utilizador.

Segundo Miller (2009), a escalabilidade é outro ganho importante para a aplicação. Ao disponibilizar um serviço, o desenvolvedor não sabe ao certo qual o número de acessos que os servidores devem suportar e é muito comum ocorrerem erros nessa previsão. Utilizando uma plataforma na nuvem, é contratada uma determinada capacidade de processamento e conforme houver um aumento ou diminuição na utilização da aplicação, este processamento disponibilizado pode ser facilmente alterado, considerando que o prestador dos serviços na nuvem possui disponível um hardware sempre acima da capacidade inicial contratada.

Sobre as desvantagens, Miller (2009) cita a necessidade da conexão com a internet sempre que for solicitado o acesso a um serviço na nuvem. Uma pane na comunicação ou velocidade baixa no tráfego dos dados pode tornar inviável a utilização desses serviços. Outro risco é quanto à segurança dos dados, pois as informações estão armazenadas em um lugar

desconhecido pelo usuário e não se pode ter total certeza de quais pessoas terão acesso a esses dados.

Na Figura 4 podemos observar uma situação comum em determinados *datacenters*: aumento da demanda seguido de uma queda e possível retomada futura. Com servidores próprios, caso ocorra um aumento na demanda, seria necessário adquirir mais equipamentos físicos para suportar as requisições. Com uma queda no uso, dificilmente esses servidores são vendidos ou desligados, na grande maioria dos casos eles ficam ociosos e geram um custo desnecessário. Em se tratando de computação na nuvem, havendo um aumento da demanda, é possível contratar mais processamento e no momento de baixa é necessário simplesmente solicitar uma diminuição no poder de processamento, resultando na diminuição do custo do serviço contratado.

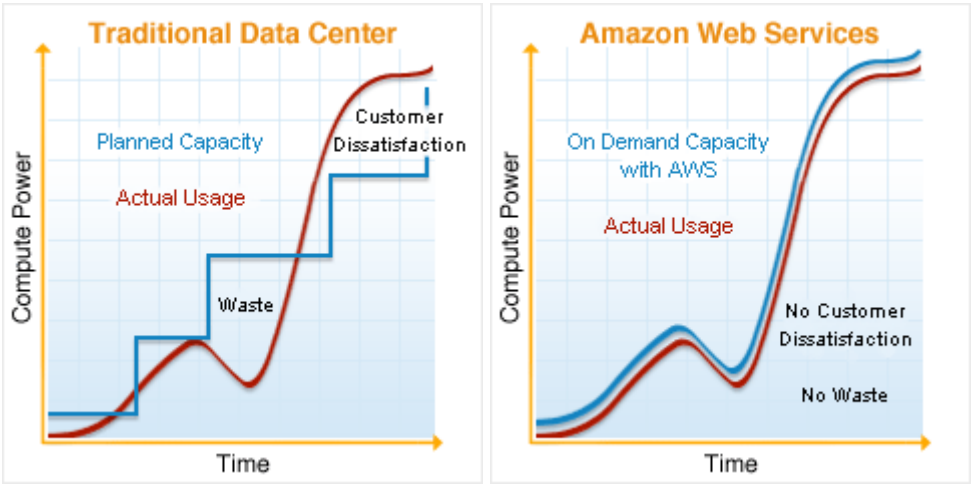

Figura 4 - Comparativo entre os custos de um datacenter tradicional e um serviço na nuvem

Fonte: Amazon (2012)

2.3 SAAS – SOFTWARE AS A SERVICE

Miller (2009) cita que o modelo de SAAS, ou *Software As A Service*, é provavelmente o maior beneficiário e o maior utilizador da computação na nuvem. Conforme ilustra a Figura 5, este modelo define que um serviço seja disponibilizado na internet, permitindo o acesso por qualquer pessoa conectada na rede mundial de computadores. Dessa maneira é possível atingir

milhares de usuários sem grandes problemas de logística, sendo necessário apenas disponibilizar o acesso do serviço via internet.

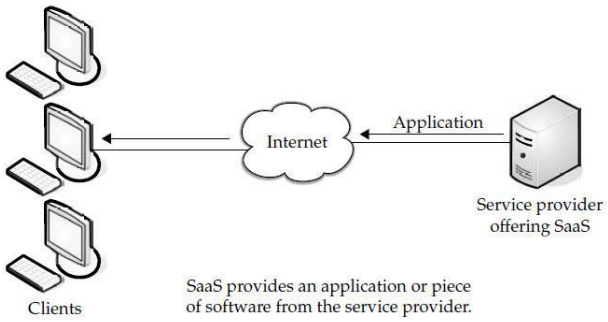

Figura 5 - Estrutura do SAAS

Fonte: Velte, Velte e Elsenpeter (2010)

O modelo de SAAS traz muitas vantagens aos utilizadores que necessitam de determinada aplicação apenas em situações específicas. Segundo Velte, Velte e Elsenpeter (2010) o melhor benefício em utilizar *Software as a Service* para o usuário é o fato de não ter a obrigação de comprar uma licença completa, mas sim pagar apenas pelo seu uso. Em muitos casos, o cliente necessita apenas de uma funcionalidade do *software* e é obrigado a comprar todo o pacote, com uma solução SAAS a utilização do software pode ser modularizada, afetando no valor que será pago pelo cliente.

Para quem está oferecendo o serviço, há uma oportunidade de conquistar mais clientes, pois com um custo mais baixo, o *software* se torna mais atraente e acessível a novos usuários. Todas as informações sobre a utilização dos serviços podem ser armazenadas e permitem uma análise com o objetivo de criar um perfil de uso para cada usuário e oferecer determinadas vantagens para grupos de usuários, como descontos em determinadas operações. Velte, Velte e Elsenpeter (2010) citam que as atualizações de versões também se tornam mais simples, como o cliente não terá o *software* instalado em seu dispositivo, é necessário apenas atualizar a versão disponibilizada na *web*.

A desvantagem deste modelo fica por conta de uma possível falha na conexão com a internet, que impossibilitaria a utilização do serviço, assim como qualquer operação da computação na nuvem.

Este método de comercializar *software* também incentiva o criador da solução a realizar a divulgação dos produtos de seus clientes, pois como a cobrança de utilização de *software* é baseada na movimentação, quanto mais dados o cliente movimentar, mais receita terá o mantenedor do serviço na nuvem.

2.4 PAAS - PLATFORM AS A SERVICE

Ambientes na nuvem com a finalidade de disponibilizar aplicações são também conhecidas por PAAS (*Plataform As A Service*). Essas plataformas disponibilizam uma determinada configuração de servidores onde as aplicações podem ser acessadas pelos usuários sem a necessidade de realizar o *download* ou de instalar a aplicação na máquina local. A estrutura de funcionamento é semelhante ao SAAS e a cobrança pelo serviço é baseada na utilização dos recursos. Segundo Velte, Velte e Elsenpeter (2010) atualmente existem três serviços expressivos de PAAS, mantidos pelas empresas Amazon, Google e Microsoft.

2.4.1 Amazon

A Amazon foi uma das primeiras empresas a oferecer serviços na nuvem incluindo processamento de dados em grande escala, simulação de uma rede de computadores, armazenamento de dados e ambiente para disponibilização de serviços. Um dos serviços é o Amazon EC2 (Amazon, 2012), um ambiente que permite aos desenvolvedores criarem suas ferramentas e disponibilizá-las na web. São disponibilizadas para contratação várias versões dos sistemas operacionais Linux e Windows, com isso o desenvolvedor pode criar seu aplicativo ou serviço no próprio computador e depois enviá-lo aos servidores da Amazon.

O método de cobrança da Amazon considera o tempo que o servidor virtual está ativo, podendo ser ligado e desligado a qualquer momento pelo desenvolvedor responsável do serviço. As tarifas de cobrança variam de acordo com o desempenho do servidor contratado e o número de máquinas virtuais contratadas, denominadas instâncias.

Um recurso extra é o auto escalonamento de instâncias. Nesta etapa, um processo do sistema operacional verifica a situação dos servidores e caso haja uma demanda muito grande de processamento, novas instâncias são ativadas permitindo que o serviço continue atendendo às

solicitações sem afetar no tempo de resposta ao usuário que está requisitando o serviço. O mesmo processo ocorre no caso contrário, quando um servidor fica ocioso, há uma diminuição das instâncias para este serviço.

Para o desenvolvimento são suportadas várias linguagens como Java, PHP, Ruby, Phyton e .Net. A Amazon disponibiliza vários documentos de auxílio a desenvolvedores, bem como kits de desenvolvimento com bibliotecas e ambientes de desenvolvimento para uma total compatibilidade com a configuração dos servidores da Amazon.

Em conjunto com o serviço EC2, responsável pelo processamento, a Amazon dispobiliza o Amazon Simple Storage Service (Amazon S3) para o armazenamento de dados. As informações são armazenadas como entidades, recebendo uma chave única para cada inserção e a quantidade desses dados armazenados define o valor que será cobrado pelo serviço.

A Amazon possui fazendas de servidores em várias regiões do mundo e permite que o usuário escolha em qual região queira que seus dados ou serviços sejam armazenados. Esta liberdade permite que determinados dados permaneçam localizados geograficamente mais perto dos usuários reais, o que diminui o tempo para acesso aos dados ou a escolha de regiões completamente diferentes para armazenar os mesmos dados, prevendo uma redundância geográfica.

2.4.2 Google

Segundo Miller (2009), a Google é líder em aplicações disponíveis na WEB e isso fortalece a opção da empresa em oferecer um ambiente para computação na nuvem para seus usuários. Este produto possui o nome de Google App Engine (GAP) e permite que o usuário, ou cliente, do Google desenvolva suas próprias aplicações e as disponibilize na mesma estrutura em que rodam os serviços oferecidos pelo próprio Google, garantindo aos clientes a mesma estabilidade e confiabilidade dos serviços do Google.

As ferramentas de desenvolvimento disponibilizadas permitem que o desenvolvedor crie e teste a sua aplicação no próprio computador, graças a um ambiente de execução que pode ser instalado e que simula exatamente a mesma configuração dos servidores do Google. Atualmente o ambiente de execução suporta as linguagens Java e Phyton, fornecendo uma completa biblioteca para desenvolvimento. As aplicações devem ser primeiramente desenvolvidas e

testadas localmente para depois ser realizado o *deploy*[2], não é possível modificar o código ou criar aplicações diretamente nos servidores.

Por questões de segurança não é possível criar arquivos diretamente nos diretórios do servidor (GOOGLE, 2012), todos os dados ou arquivos devem ser obrigatoriamente persistidos na base de dados do Google. A base de dados não possui a estrutura comum de entidade-relacionamento, os dados são gravados seguindo uma tendência para computação na nuvem, agrupando um determinado conjunto de informações e armazenando-os como entidades em um mesmo local para agilizar uma posterior consulta, sem a necessidade de buscar as informações em várias tabelas diferentes.

O valor cobrado pelo Google App Engine é calculado com base no volume de dados armazenado e na largura de banda para acesso aos aplicativos. Qualquer uma das métricas de cobrança pode ser controlada pelo desenvolvedor, sendo possível definir um limite a fim de não aumentar o custo do serviço com um possível aumento na demanda do aplicativo. Além disso, o GAP oferece um nível gratuito de utilização e a cobrança só começa após passar este limite.

O Google permite realizar uma integração entre o aplicativo disponibilizado no Google App Engine e outros serviços nativos como Gmail, Agenda, Mapas, etc. Assim, pode-se usar a própria conta do usuário no Google para realizar a autenticação no aplicativo e ainda utilizar informações dessa conta para diversos fins dentro do aplicativo desenvolvido. É possível também realizar comunicação com outros aplicativos em qualquer serviço na nuvem utilizando *web services*.

2.4.3 Microsoft

O Windows Azure é a plataforma de serviços na nuvem da Microsoft (Windows Azure, 2012). Nesta plataforma são oferecidos serviços de processamento de dados, utilização de máquinas virtuais e armazenamento de dados. Para os desenvolvedores estão disponíveis ferramentas e bibliotecas com suporte a várias linguagens de programação, incluindo .NET, node.js, PHP, Java e Phyton, permitindo uma grande liberdade na escolha da linguagem para o desenvolvimento de aplicações.

[2] Deploy: Disponibilização da aplicação para os usuários. Neste caso, envio da aplicação desenvolvida ao ambiente de execução dos servidores do Google.

O armazenamento das informações deve ser realizado em um banco de dados semelhante ao Microsoft SQL Server, com a possibilidade de utilização de recursos NoSQL. O NoSQL extingue o relacionamento dos dados, que é padrão nos bancos de dados atuais e grava todos os dados agrupados, como se fossem um objeto. A cada objeto gravado é atribuída uma chave única que deve ser utilizada para uma posterior consulta na base de dados. Este método é utilizado em armazenamento de grandes volumes de dados, buscando alta escalabilidade e agilidade na consulta (Velte, Velte e Elsenpeter, 2010).

A definição dos valores de cobrança do Azure depende basicamente de cinco fatores: poder de processamento solicitado; quantidade de máquinas virtuais; tempo de atividade de cada máquina virtual; quantidade de dados que será armazenada e largura de banda para tráfego de dados considerando entrada e saída. No site da ferramenta[3], há uma calculadora para os usuários preverem seus gastos informando projeções dos dados que serão utilizados e podendo escolher a configuração que mais se adapte às suas necessidades.

A Microsoft foi uma das empresas mais recentes a disponibilizar serviços baseados na nuvem e, segundo Velte, Velte e Elsenpeter (2010), com isso acaba ficando atrás do Google e da Amazon neste mercado.

Tabela 1 - Comparativo dos serviços de computação na nuvem na versão de avaliação (sem custos)

Serviço	Tempo máximo de uso	Armazenamento	Limite de Transações
Amazon	1 ano	5 GB	20.000 consultas e 2.000 inserções
Google	Ilimitado	500 MB	5 milhões de acessos/mês
Microsoft	90 dias	35 GB	50 milhões

Fonte: Autor

Para este projeto será utilizada a plataforma do Google no armazenamento e gerenciamento dos dados na nuvem. Considerando os dados apresentados na Tabela 1, a versão de avaliação do Google App Engine se mostra mais interessante, pois possui um período ilimitado para utilização e o espaço de armazenamento comporta a realização de todos os testes necessários e até mesmo uma real utilização após os testes.

[3] URL para acesso à calculadora: https://www.windowsazure.com/pt-br/pricing/calculator/?scenario=full

2.5 WEB SERVICES

Web services são interfaces de comunicação para aplicações disponibilizadas na internet, fornecendo um serviço de processamento de dados remoto (SNELL; TIDWELL; KULCHENKO, 2002). Nesta interface são aceitas conexões originadas de clientes de qualquer plataforma, sendo esta uma das principais vantagens na utilização destes serviços.

Na utilização de um *web service*, a aplicação cliente acessa uma URL pré-definida e tem a possibilidade de passar parâmetros que serão interpretados pela aplicação servidor. O servidor por sua vez usa os parâmetros, quando estes existem, e realiza o processamento, devolvendo ao cliente o resultado obtido.

Estes serviços são utilizados, em sua maioria, quando se deseja realizar uma comunicação entre dois sistemas distintos e a aplicação servidor possui dados ou alguma rotina de processamento que a aplicação cliente não conheça. Na Figura 6 podemos observar a representação de um usuário acessando um serviço através da rede utilizando um *web service*.

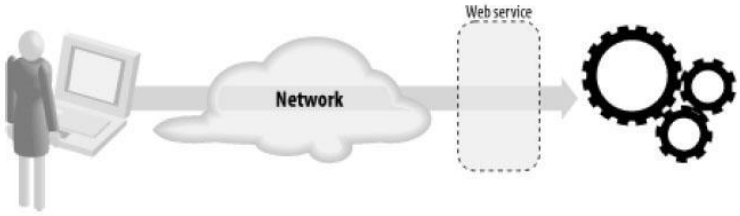

Figura 6 - Representação de uma conexão via *web service*

Fonte: Snell, Tidwell e Kulchenko (2002)

Segundo Richardson e Ruby (2007), todos os *web services* utilizam o protocolo HTTP, mas há vários padrões de encapsulamento e transporte dos dados. Desses padrões, dois ganharam notoriedade e a preferência dos desenvolvedores: SOAP e REST.

2.5.1 SOAP

O padrão SOAP (*Simple Object Acess Protocol*) compreende uma estrutura XML para organização dos dados que serão trafegados. Conforme Snell, Tidwell e Kulchenko (2002), toda e qualquer mensagem SOAP deve conter um bloco de cabeçalho e um bloco de dados

encapsulando qualquer informação que deseja ser transmitida, conforme ilustrado na Figura 7. Por conta deste encapsulamento, acaba-se criando um *overhead*[4] na mensagem, com vários dados que não serão utilizados nem pelo emissor nem pelo receptor da mensagem.

Richardson e Ruby (2007) afirmam que o padrão SOAP possui a vantagem de ser facilmente configurável com ferramentas de desenvolvimento atuais, que já contam com bibliotecas prontas para seu uso. Apesar dessa facilidade na implementação, uma eventual manutenção pode ser mais custosa, tendo em vista a complexidade que envolve o padrão SOAP.

Como neste trabalho aborda-se um cenário com utilização de dispositivos móveis para a comunicação, e estes por sua vez podem estar utilizando uma comunicação de baixa velocidade ou instável, o padrão SOAP não é o mais recomendável por conta do excesso de informação trafegada e pelo processamento necessário ao interpretar as informações que estão encapsuladas no arquivo recebido. Em comunicações cliente-servidor, busca-se transmitir a menor quantidade possível de dados na intenção de que a atualização dos dados seja rápida, eficiente e, se possível, imperceptível ao usuário.

```
<s:Envelope
 xmlns:s="http://www.w3.org/2001/06/soap-envelope">
    <s:Header>
       <m:transaction xmlns:m="soap-transaction"
                      s:mustUnderstand="true">
           <transactionID>1234</transactionID>
       </m:transaction>
    </s:Header>
    <s:Body>
       <n:purchaseOrder xmlns:n="urn:OrderService">
          <from><person>Christopher Robin</person>
                <dept>Accounting</dept></from>
          <to><person>Pooh Bear</person>
               <dept>Honey</dept></to>
          <order><quantity>1</quantity>
                 <item>Pooh Stick</item></order>
       </n:purchaseOrder>
    </s:Body>
</s:Envelope>
```

Figura 7 - Exemplo de arquivo XML com encapsulamento SOAP

Fonte: Snell, Tidwell e Kulchenko(2002)

[4] Overhead: Excesso de caracteres que são transmitidos junto com a mensagem, exigidos por conta da estrutura do encapsulamento, mas não sendo utilizados pelas aplicações.

2.5.2 REST

Em alternativa ao padrão SOAP, o REST (Representation State Transfer) mostra-se muito mais simples quanto à implementação e transporte dos dados. Segundo Richardson e Ruby (2007), o padrão REST determina que os dados resultantes do processamento do servidor possam ser apresentados em uma simples página *web*, para posterior interpretação da aplicação cliente. Os dados podem ser apresentados em diversas estruturas como XML, JSON, texto puro ou arquivos binários, mas sem nenhum tipo de encapsulamento específico como no padrão SOAP. O sistema que apresenta estas características pode ser denominado RESTful.

A requisição feita por um cliente, para uma interface REST, é um acesso por uma URL definida pelo servidor. Para a manipulação dos dados, são utilizados comandos conhecidos do protocolo HTTP: GET, PUT, POST e DELETE. Segundo Richardson e Ruby (2007), uma aplicação pode prever todas as situações utilizando somente o método GET ou POST e tratar cada situação de maneira independente com métodos internos, identificando a operação por meio de parâmetros passados na URL.

Conforme o exemplo apresentado por Wilde e Pautasso (2011) na Figura 8, o endereço "example.org/book?" identifica o serviço que será invocado no servidor. Em seguida é apresentada a chave para consulta, neste exemplo definida por "title" e logo após o valor da chave para consulta, identificada por "zen". Com essas informações o servidor processa a requisição e devolve ao cliente o resultado da consulta, semelhante a uma página *web*, caso tivesse sido acessada por algum navegador.

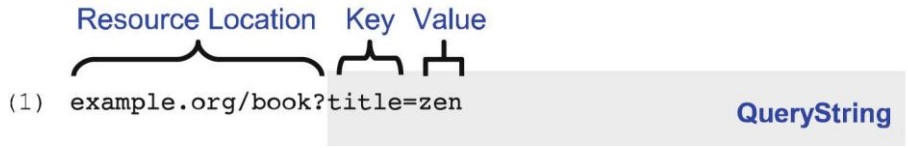

Figura 8 - URLs para consulta de dados com o método REST

Fonte: Adaptado de Wilde e Pautasso (2011)

A plataforma de disponibilização de aplicativos na nuvem Google App Engine, escolhida para este projeto, tem como padrão e sugestão de uso, segundo o Google Developers (2012), a busca de URL para realizar a comunicação entre aplicativos. Esta busca de URL atende as

características do REST citadas acima e mostra-se perfeitamente adaptável ao projeto em questão, não sendo necessário utilizar *frameworks* ou ferramentas de terceiros para viabilizar a comunicação entre cliente e servidor.

O padrão REST apresenta as melhores características e é o padrão mais aconselhável para utilização na implementação deste trabalho. Sua estrutura é simples, não possui um *overhead* com encapsulamento como no padrão SOAP e possui suporte nativo na ferramenta Google App Engine.

3 DESCRIÇÃO DA SOLUÇÃO

Considerando o estudo realizado com as ferramentas existentes no mercado, percebe-se uma necessidade de ferramentas que permitam uma consulta *offline* ao cardápio e sem a necessidade do cadastro de um usuário. Também fica a desejar uma maior cobertura a cidades de menor porte, mas com um perfil de público ideal para esta solução, como cidades turísticas.

A solução proposta neste trabalho conta com uma aplicação para *smartphones* utilizada pelos clientes, uma aplicação WEB utilizada pelos estabelecimentos e o servidor que irá gerenciar os dados, conforme representação na Figura 9. Utilizando *web services* para a comunicação, serão realizadas consultas e inclusões de dados no servidor originadas dos clientes.

Neste projeto, optou-se por desenvolver o aplicativo para apenas um sistema operacional móvel, tendo em vista o tempo necessário para desenvolver dois aplicativos com características de desenvolvimento diferentes e considerando que com um aplicativo já é possível realizar todos os testes verificando a viabilidade do projeto. A plataforma escolhida foi o Android, considerando o maior número de usuários, a possibilidade de utilização do sistema operacional Windows para o desenvolvimento do aplicativo e o valor mais baixo dos dispositivos para realização dos testes. Uma futura continuidade do projeto pode incluir o desenvolvimento desta mesma aplicação para dispositivos com iOS, abrangendo assim quase a totalidades dos dispositivos móveis.

O aplicativo WEB deverá ser gerenciado por um atendente do estabelecimento que tenha contato com a equipe de garçons e se possível também com a equipe que irá preparar o prato. A organização da informação dentro do estabelecimento fica a critério de cada um, confirme sua logística ou particularidades. É possível disponibilizar uma comunicação com impressoras térmicas, onde as impressoras poderiam ficar dentro da cozinha e os pedidos confirmados pelo atendente seriam impressos diretamente nestas impressoras. Neste projeto não será criada a interface para impressão, tendo em vista que esta etapa consome um alto tempo desenvolvimento e não é essencial para o funcionamento da solução.

A receita prevista para este sistema será proveniente de uma mensalidade paga pelos estabelecimentos e um percentual do valor de cada pedido realizado. Este método permite ao estabelecimento escolher entre apenas manter as informações de seu cardápio disponível para consulta ou disponibilizar a solução completa aos seus clientes, com a opção de autoatendimento.

Caso o estabelecimento opte pela solução completa, é possível definir um limite mensal de faturamento livre de cobrança, seguindo a mesma estratégia utilizada pelas empresas fornecedoras de serviços na nuvem. Esta é uma técnica interessante principalmente para soluções novas, até que estejam totalmente maduras e ganhem a confiança de seus usuários.

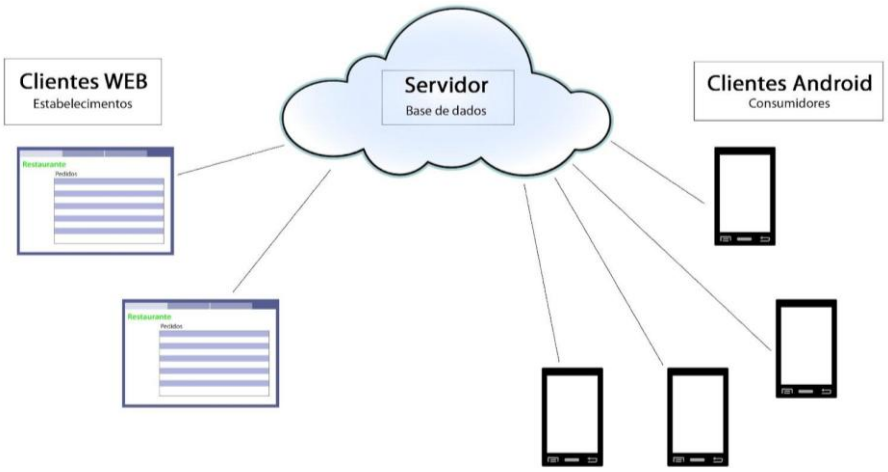

Figura 9 - Estrutura da solução

Fonte: Autor

3.1 APLICATIVO MÓVEL

O aplicativo para *smartphones* será utilizado pelos clientes do estabelecimento com o objetivo de inserir os pedidos no sistema e acompanhar a situação dos mesmos. Ao entrar no estabelecimento, o usuário deverá conectar-se à internet, iniciar o aplicativo e realizar o *login* com usuário e senha assim que solicitado. Em seguida será necessário realizar o *check in*, que consiste em informar ao aplicativo um código de verificação e o número da mesa, ambos disponibilizados pelo estabelecimento. Confirmada a validade dos dados, é habilitada a opção de iniciar um novo pedido. O código de verificação é único por estabelecimento e atualizado periodicamente. Esta é uma proteção ao estabelecimento para evitar que usuários mal intencionados insiram pedidos sem nem ao menos estar no estabelecimento.

O aplicativo irá dispor de funções de “avaliação do estabelecimento” e “chamar garçom” que permanecerão habilitadas enquanto o cliente estiver com o status de *check in* ativo. Serão disponibilizadas também interações caso o cliente ainda não se encontre no estabelecimento, como a informação de ocupação das mesas e a opção de reserva de mesa diretamente pelo

aplicativo. Uma futura funcionalidade a ser implementada será a opção de realizar o pedido e solicitar a tele-entrega em um endereço que será informado pelo cliente.

O cardápio de cada estabelecimento será armazenado no próprio dispositivo, permitindo assim um acesso *offline*. Esta característica é importante para os casos em que o cliente apenas deseja analisar o cardápio, sem a possibilidade de conectar-se com a internet no momento. Sempre que o aplicativo é iniciado será realizada uma tentativa de atualização dos cardápios armazenados, caso uma conexão com a internet esteja disponível. A utilização de imagens dos pratos oferecidos seria um facilitador para o cliente e um grande atrativo do aplicativo, mas optou-se por não utilizar imagens levando em conta o espaço necessário no dispositivo móvel e a alta carga na busca das informações via *web service*, no momento da atualização do cardápio.

Segundo Neil (2012), listas e abas são técnicas altamente aconselháveis para facilitar a navegação em aplicativos para dispositivos móveis. Nesta aplicação as listas serão utilizadas para visualizar os estabelecimentos disponíveis e também para exibir os produtos no momento da consulta ou da inclusão de um novo pedido. A técnica de abas será utilizada nas telas relacionadas às informações detalhadas de cada estabelecimento, como o cardápio e as reservas. Outras técnicas indicadas são o formulário, que será utilizado no momento do cadastro do usuário, e a pesquisa, necessária na busca por algum prato.

Em resumo, podemos definir três status do usuário no aplicativo:

Offline: Não necessita qualquer tipo de identificação. Permite apenas visualizar cardápios e informações como telefone, endereço, localização, horário e avaliações de clientes;

Logado: Necessita conexão com internet e criação de um usuário. Permite visualizar a ocupação dos estabelecimentos e realizar reserva de mesa, além das informações disponibilizadas no status offline;

Check in ativo: Exige status logado e código de verificação do estabelecimento. Permite realizar pedidos, chamar garçom e avaliar estabelecimento quanto ao serviço oferecido.

Este aplicativo será disponibilizado para download gratuito no repositório de aplicativos do Android, o Google Play.

3.2 APLICATIVO WEB

O aplicativo WEB disponibilizado será a ferramenta de uso dos estabelecimentos. Cada pedido inserido pelos clientes aparecerá em uma lista de pedidos a serem confirmados. Um funcionário do estabelecimento deverá ficar responsável pela confirmação do pedido e garantir que o setor responsável irá dar continuidade ao processo.

Na ferramenta também é possível fazer a manutenção do cardápio e das informações que serão disponibilizadas aos clientes. O código de verificação será gerado por esta aplicação, devendo ser utilizado pelos clientes no momento do *check in*. Cada estabelecimento terá também acesso ao seu histórico de pedidos realizados pelos clientes móveis permitindo um acompanhamento da utilização da solução, assim como a consulta das informações referentes às reservas.

Alguns estabelecimentos possuem um sistema de gerenciamento com as informações que serão necessárias na aplicação sugerida por este trabalho. Visando uma maior produtividade é importante uma integração entre os sistemas para a troca dos dados, eliminando o retrabalho no cadastramento destas informações. Não será criada, nesta etapa, uma interface para importação de dados, considerando que o objetivo principal da aplicação não depende desta funcionalidade. Como continuidade do projeto, está previsto um estudo dos principais sistemas utilizados pelos estabelecimentos, seguido de análise e desenvolvimento de uma interface para realizar a importação dos dados.

Esta ferramenta será desenvolvida na linguagem de programação Java para WEB e fará uso do Google App Engine para ser disponibilizada e acessada pelos estabelecimentos. Seguindo técnicas para design de interfaces WEB (SCOTT; NEIL, 2009), o *layout* da página deve ser leve, com as informações claras e as operações de fácil acesso ao usuário. Apesar de prever um treinamento para os usuários dos estabelecimentos, o objetivo é criar uma interface o mais intuitiva possível, reduzindo a necessidade de um eventual suporte para situações rotineiras.

3.3 SERVIDOR

A aplicação servidor será responsável por gerenciar os dados e responder as solicitações dos aplicativos em Android e WEB. Para cada interação de consulta ou inserção será disponibilizada uma URL para conexão via *web service*.

As informações dos estabelecimentos, os pedidos realizados pelos clientes e a aplicação que irá gerenciar esses dados serão armazenados na nuvem, utilizando a plataforma do Google App Engine. Todas as informações serão armazenadas na mesma base de dados, ficando a cargo do aplicativo cliente solicitar ao servidor os dados necessários. O servidor estará preparado para interpretar essas consultas e retornar as informações solicitadas. Essas solicitações de consulta, assim como as inserções dos dados, serão efetuadas através de *web services* disponibilizados pelo servidor e conhecidos pelo aplicativo móvel.

O servidor deverá estar ativo e preparado para receber solicitações a qualquer momento, tanto para consultas dos cardápios, quanto para alterações dos dados por parte dos estabelecimentos. Segundo Roche e Douglas (2009), os aplicativos armazenados na nuvem do Google dispõem de uma biblioteca específica para envio de emails, que neste caso poderão ser usados para confirmação de cadastro ou confirmação de reserva solicitada pelo usuário.

A segurança dos dados é sempre uma questão muito importante, principalmente quando se trata de armazenamento na nuvem. Neste servidor, estarão armazenadas informações sobre o faturamento dos estabelecimentos incluindo número de atendimentos, valores faturados e clientes atendidos. O Google garante a segurança, privacidade e proteção dos dados armazenados, assim como os dados de todos os seus usuários ativos atualmente.

4 ANÁLISE DO SISTEMA

Neste capítulo serão apresentadas informações referentes à análise do sistema. Para a elaboração da análise foram utilizadas algumas fases da metodologia ICONIX, sendo elas: Análise de Requisitos, Modelo de Domínio e Modelo de Casos de Uso.

4.1 REQUISITOS

Segundo Blaha e Rumbaugh (2006), os requisitos descrevem o que o sistema deve contemplar a partir do ponto de vista dos usuários. A Tabela 2 traz uma lista com os requisitos funcionais para o sistema, que estão ligados diretamente na interação dos usuários com o sistema. Os requisitos foram elaborados para atender um processo de autoatendimento contando também com funcionalidades que auxiliem o cliente no momento da escolha da sua refeição. A lista de requisitos contempla as duas aplicações cliente: aplicativo WEB e aplicativo móvel.

Tabela 2 - Requisitos funcionais do sistema

Item	Requisito	Aplicativo	Descrição
RF1	Aprovar ou cancelar as solicitações dos clientes.	WEB	Possibilitar um controle das solicitações efetuadas pelos clientes como pedidos, garçom e reservas.
RF2	Relatório com a movimentação.	WEB	Relatório com o histórico de todas as solicitações realizadas.
RF3	Manutenção de cardápio.	WEB	Possibilitar incluir, excluir ou modificar o cardápio cadastrado no sistema.
RF4	Manter a informação de ocupação do estabelecimento.	WEB	Permitir ao atendente informar no sistema a ocupação do estabelecimento.
RF5	Consultar as reservas do estabelecimento.	WEB	Permitir ao atendente listar todas as reservas para determinada data, com o objetivo de organizar as mesas do estabelecimento.
RF6	Exibir extrato do pedido.	WEB	Totalizar o pedido do cliente e exibir os produtos consumidos com os respectivos valores.
RF7	Consultar informações do estabelecimento localmente.	Móvel	Permitir a consulta das informações dos estabelecimentos sem a necessidade de conexão com a internet.
RF8	Realizar reservas.	Móvel	Solicitar reserva de mesa nos estabelecimentos.

RF9	Realizar pedido.	Móvel	Permitir o autoatendimento quando o cliente estiver no estabelecimento, com acesso ao cardápio para escolha dos produtos.
RF10	Solicitar garçom na mesa.	Móvel	Permitir ao cliente solicitar a presença de um garçom na mesa.
RF11	Fechar conta.	Móvel	Enviar ao estabelecimento uma solicitação de encerramento do pedido.
RF12	Consultar histórico de pedidos.	Móvel	Consultar histórico dos pedidos realizados pelo cliente na ferramenta.
RF13	Pesquisar pratos.	Móvel	Pesquisa de determinado prato entre os vários estabelecimentos.
RF14	Acompanhar pedido	Móvel	Acompanhar a situação do pedido.

Na Tabela 3 são listados os requisitos não funcionais para o sistema. Os requisitos não funcionais se voltam para a questão de "como o sistema deve fazer" e interferem de forma indireta o processo de análise. O foco destes requisitos é em questões como interface da aplicação, arquitetura do sistema e o hardware que irá suportar e executar a aplicação (DENNIS, WIXOM, 2005).

Tabela 3 - Requisitos não funcionais do sistema

Item	**Requisito**	**Descrição**
RNF1	Compatibilidade entre diversos navegadores.	O aplicativo WEB deverá ser compatível e apresentar o mesmo comportamento nos mais diversos navegadores existentes.
RNF2	Armazenamento de dados na nuvem.	Os dados, bem como o aplicativo que irá gerenciá-los, deverão estar armazenados no serviço de processamento na nuvem do Google, denominado Google App Engine.
RNF3	Desenvolvimento do aplicativo móvel na linguagem Java.	O aplicativo móvel deverá ser desenvolvido na linguagem Java e utilizar as bibliotecas suportadas pelo sistema operacional Android.
RNF4	Desenvolvimento do aplicativo WEB na linguagem Java.	O aplicativo WEB deverá ser desenvolvido na linguagem Java para posterior disponibilização no ambiente do Google App Engine.
RNF5	Utilização de *web services* para a comunicação entre cliente e servidor.	Como citado nos estudos do capítulo anterior, para este projeto foi escolhida a solução de *web services* com encapsulamento REST para realizar a comunicação e a troca de dados entre o servidor e os clientes.
RNF6	Interface simples e intuitiva.	Considerando que o aplicativo móvel pode ser utilizado por qualquer pessoa que possua um dispositivo com Android, a interface deve ser simples e intuitiva fazendo com que o cliente consiga realizar todas as tarefas sem a necessidade de auxílio ou suporte.

4.2 MODELO DE DOMÍNIO

Segundo Rosenberg, Stephens e Cope (2005), o modelo de domínio é uma das etapas mais importantes da análise e serve para identificar os objetos de um problema do mundo real e auxiliar para transformá-los em objetos de um sistema. O modelo de domínio representa uma visão inicial das possíveis classes que irão compor o sistema e a relação entre elas em alto nível. Ao longo do projeto essas classes ganham atributos, métodos e relacionamentos.

Na Figura 10 está representado o modelo de domínio para o sistema em questão.

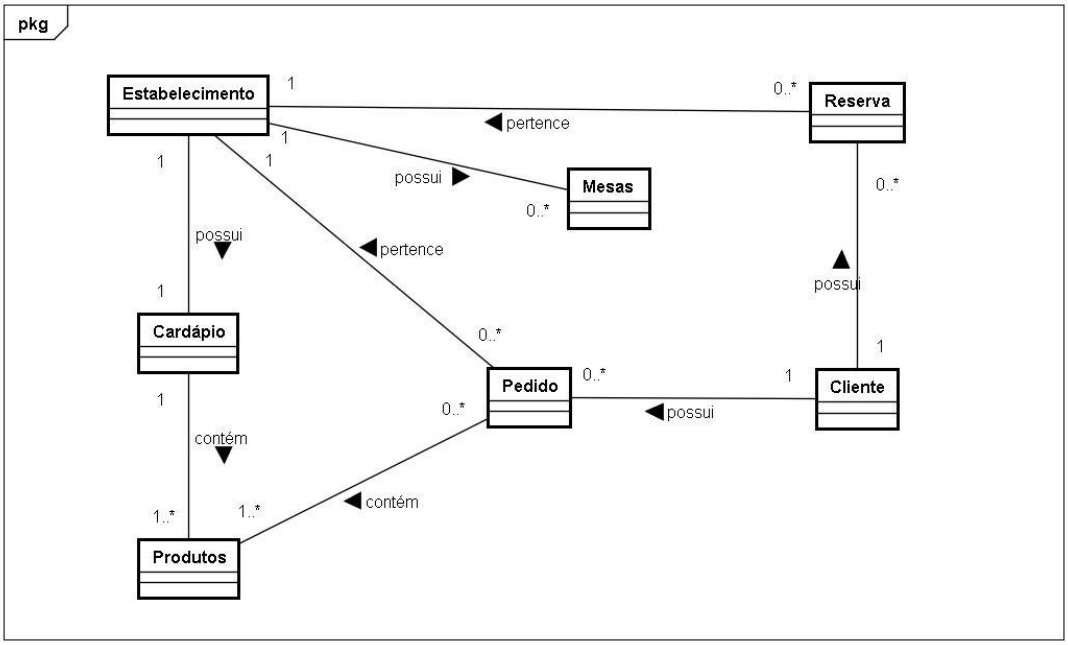

Figura 10 - Modelo de Domínio

Fonte: Autor

4.3 CASOS DE USO

De acordo com os requisitos apresentados, foram definidos casos de uso para facilitar o entendimento das funcionalidades do sistema. A Figura 11 e a Figura 12 apresentam uma visão geral dos casos de uso do sistema, onde constam os três atores que irão interagir com a aplicação:

- Administrador do sistema: Desenvolvedor ou a pessoa responsável pela manutenção do sistema. Ficará a cargo deste ator a liberação do acesso dos atendentes ao sistema.
- Cliente: Usuário da aplicação móvel que busca informações sobre os estabelecimentos e possui a necessidade de autoatendimento.
- Atendente: Pessoa designada pelo estabelecimento para gerenciar as solicitações do cliente.

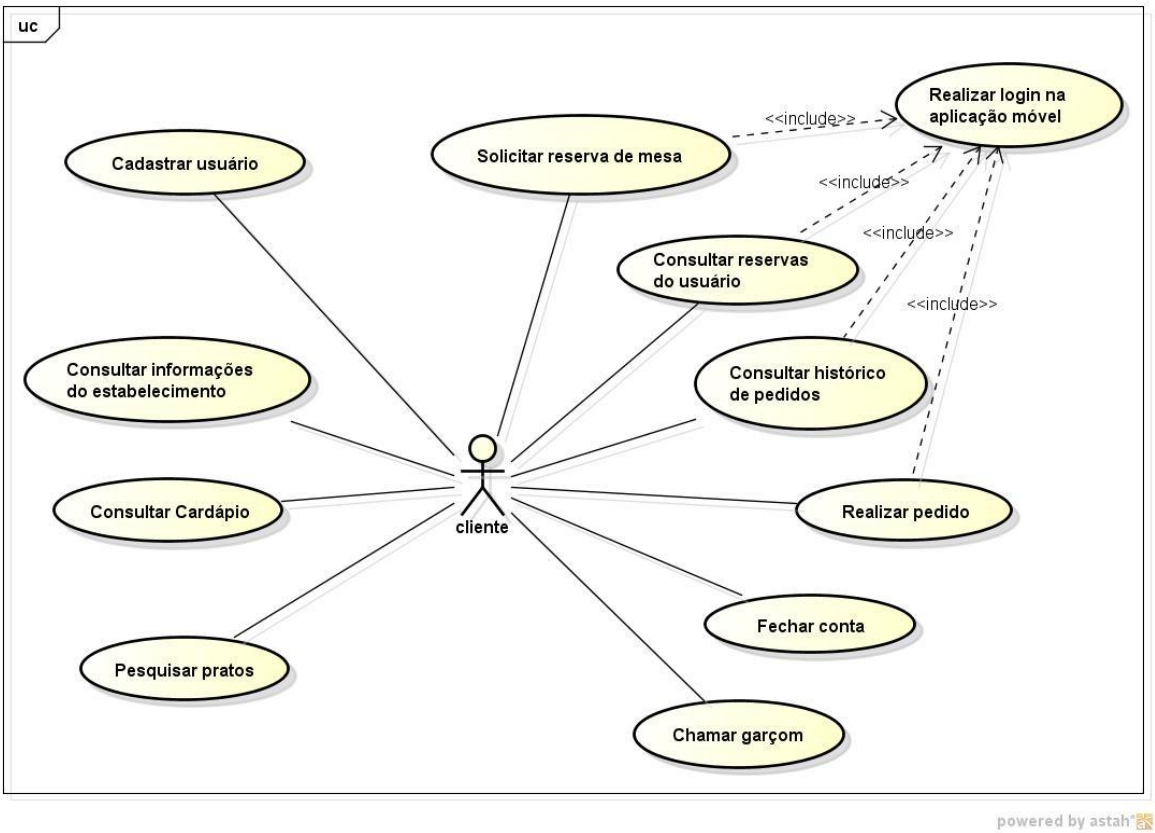

Figura 11 - Casos de uso da aplicação móvel

Fonte: Autor

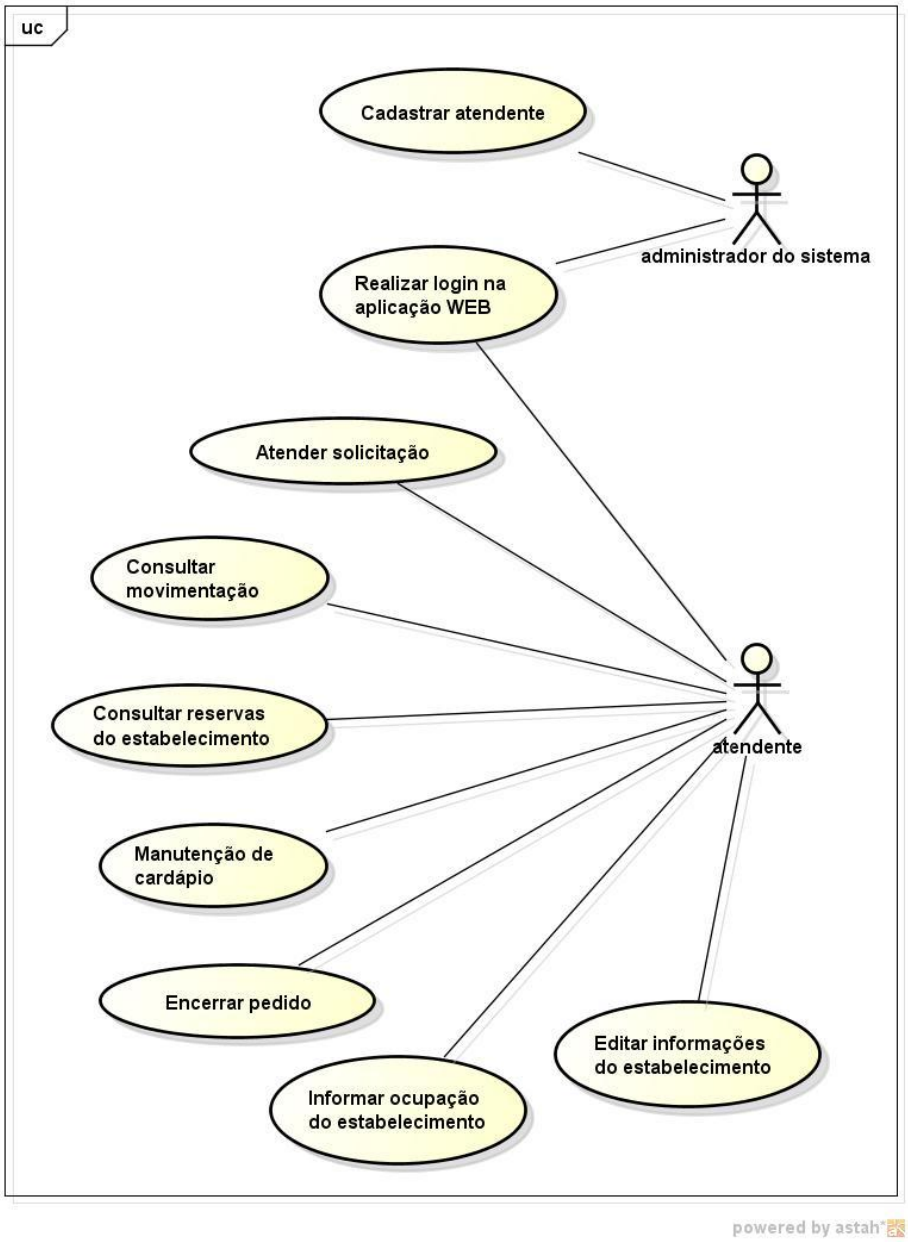

Figura 12 - Casos de uso da aplicação servidor

Fonte: Autor

A seguir, cada caso de uso será apresentado detalhadamente com uma tabela descritiva e o protótipo de tela do sistema que irá atender o caso de uso em questão. Os casos de uso apresentados estão divididos por aplicativo: Móvel e WEB.

4.3.1 Casos de uso do aplicativo móvel

A partir da Tabela 4 serão apresentados os casos de uso do aplicativo móvel, utilizado pelos clientes. Neste projeto será desenvolvido apenas o aplicativo para o sistema operacional Android, mas os mesmos casos de uso podem ser utilizados para uma continuidade do projeto, considerando o desenvolvimento para outras plataformas móveis.

Tabela 4 - Caso de uso narrativo: Realizar login na aplicação móvel

Identificador:	Realizar login na aplicação móvel
Descrição:	Permite ao cliente realizar o login na aplicação informando email e senha para acessar determinadas funcionalidades.
Ator Principal:	Cliente
Fluxo Principal:	1. O cliente acessa uma opção do sistema que exige um usuário cadastrado (solicitar reserva, realizar pedido, consultar histórico de pedidos). 2. O aplicativo exibe a tela de login e solicita que o cliente informe o email e a senha. 3. O cliente informa o email e a senha e confirma a operação. 4. O aplicativo valida os dados e retorna para o cliente a opção desejada.
Fluxo Secundário 1:	2.1 O aplicativo reconhece que o usuário já realizou *login* em outro momento. Nenhum dado é solicitado e o aplicativo segue para a opção desejada.
Fluxo secundário 2:	3.1. O cliente não possui cadastro e clica na opção de cadastrar usuário. 3.2. O aplicativo segue para o caso de uso de cadastro de usuário.
Fluxo secundário 3:	4.1. O aplicativo exibe a mensagem que o usuário e a senha não conferem e permanece na página de login.

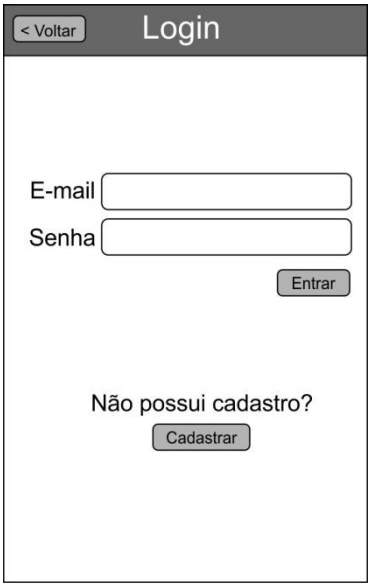

Figura 13 - Aplicativo móvel: Login
Fonte: Autor

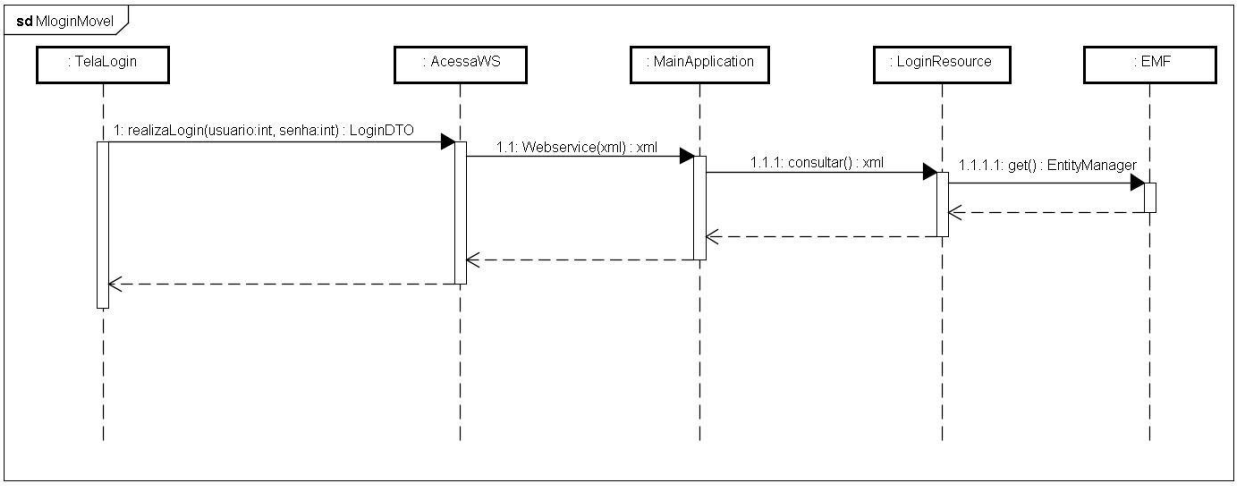

Figura 14 - Diagrama de sequência: Login Móvel

Fonte: Autor

Tabela 5 - Caso de uso narrativo: Cadastrar usuário

Identificador:	Cadastrar usuário
Descrição:	O cliente cadastra seu próprio usuário para acessar funcionalidades que necessitam de registro no sistema.
Ator primário:	Cliente.
Fluxo principal:	1. O cliente tenta acessar uma funcionalidade que necessita de registro. 2. O aplicativo exibe a tela para informar usuário e senha ou opção para cadastrar usuário. 3. O cliente clica no botão de cadastrar usuário. 4. O aplicativo exibe uma tela e solicita o preenchimento dos seguintes campos: nome, email, telefone e senha. 5. O cliente preenche os dados e confirma o cadastramento. 6. O sistema envia um email para o cliente, confirmando o cadastro.
Fluxo alternativo:	5.1 O email informado já está cadastrado no sistema ou é inválido. 5.2 O aplicativo exibe uma mensagem de erro e permanece na tela de cadastro.

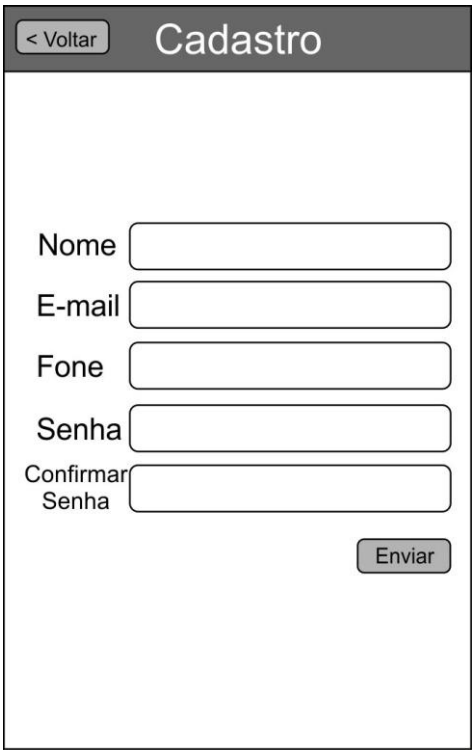

Figura 15 - Aplicativo móvel: Cadastrar usuário
Fonte: Autor

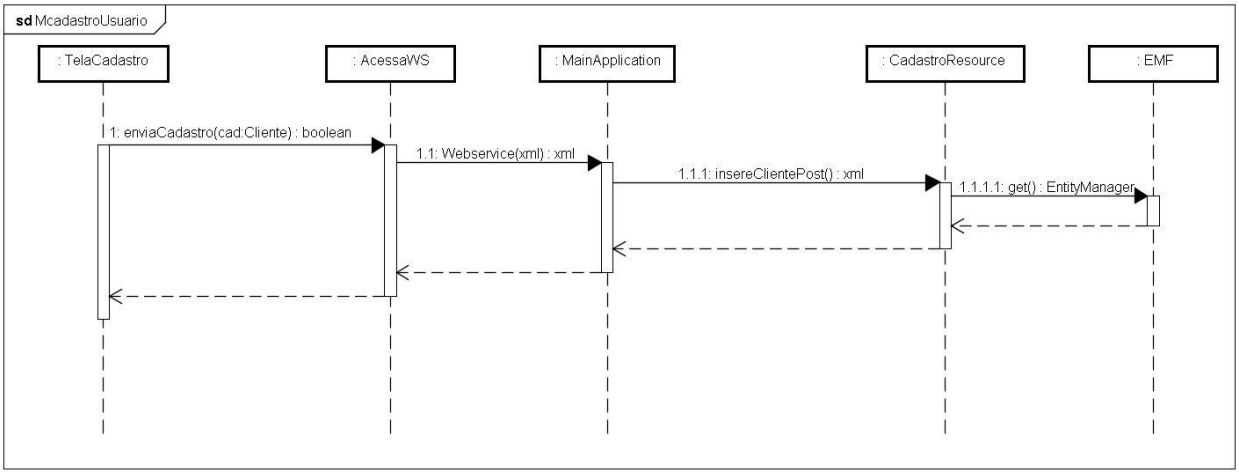

Figura 16 - Diagrama de sequência: Cadastrar usuário

Fonte: Autor

Tabela 6 - Caso de uso narrativo: Consultar informações do estabelecimento

Identificador:	Consultar informações do estabelecimento
Descrição:	Exibe informações dos estabelecimentos como cardápio, endereço, telefone, horário de funcionamento e um texto que pode ser cadastrado pelo próprio estabelecimento.
Ator primário:	Cliente
Fluxo principal:	1. O cliente acessa o aplicativo 2. O aplicativo busca via *web service* os dados de todos os estabelecimentos e atualiza o banco de dados local. 2. É exibida para o usuário a lista dos estabelecimentos cadastrados no sistema e junto de cada estabelecimento é fornecido o endereço, telefone e avaliação dos clientes. 3. O cliente clica sobre um determinado estabelecimento. 4. O aplicativo apresenta uma tela exclusiva do estabelecimento selecionado exibindo o endereço, telefone, lotação, horário de funcionamento e um texto cadastrado pelo próprio estabelecimento.

Figura 17 - Aplicativo móvel: Tela principal
Fonte: Autor

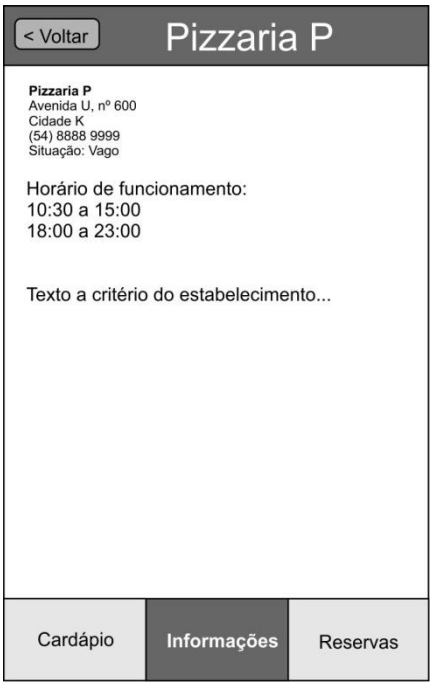

Figura 18 - Aplicativo móvel: Consultar informações do estabelecimento
Fonte: Autor

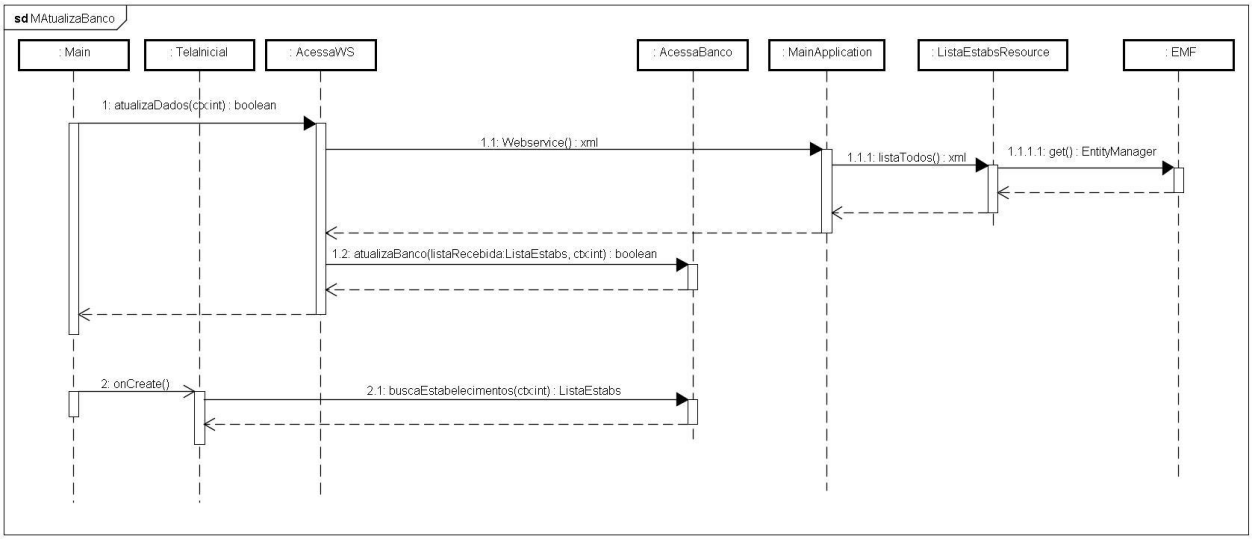

Figura 19 - Diagrama de sequência: Atualiza banco de dados

Fonte: Autor

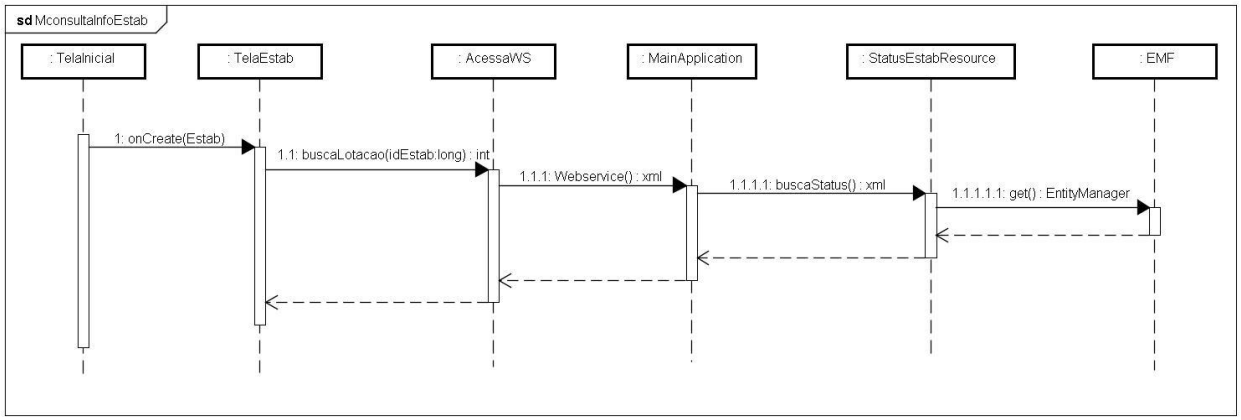

Figura 20 - Diagrama de sequência: Consultar informações do estabelecimento

Fonte: Autor

Tabela 7 - Caso de uso narrativo: Consultar cardápio

Identificador:	Consultar cardápio
Descrição:	Exibe o cardápio de um determinado estabelecimento. São disponibilizadas informações como nome do prato, preço e ingredientes. A consulta pode ser realizada *offline* e sem a necessidade de cadastrar usuário.
Ator primário:	Cliente
Fluxo principal:	1. O cliente acessa o aplicativo 2. O aplicativo exibe a lista dos estabelecimentos cadastrados no sistema e junto de cada estabelecimento é fornecido o endereço, telefone e avaliação dos clientes. 3. O cliente clica sobre um determinado estabelecimento. 4. O aplicativo apresenta uma tela exclusiva do estabelecimento selecionado exibindo o endereço, telefone, lotação, horário de funcionamento e um texto cadastrado pelo próprio estabelecimento. 5. O cliente clica no botão "Cardápio" exibido no rodapé da página. 6. O aplicativo exibe o cardápio do estabelecimento selecionado.

Figura 21 - Aplicativo móvel: Consultar cardápio
Fonte: Autor

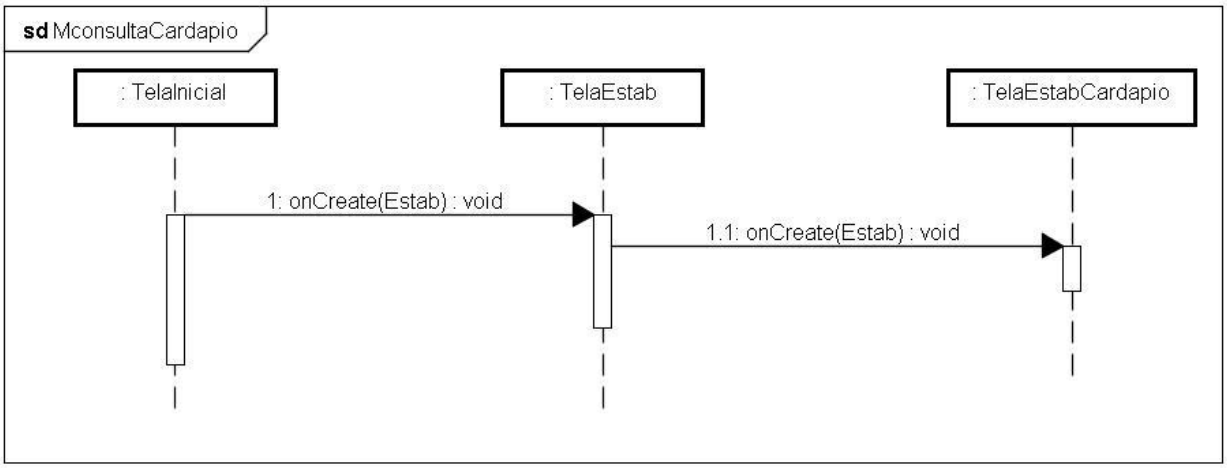

Figura 22 - Diagrama de sequência: Consultar cardápio

Fonte: Autor

Tabela 8 - Caso de uso narrativo: Consultar reserva do usuário

Identificador:	Consultar reserva do usuário
Descrição:	Lista todas as reservas solicitadas pelo cliente e a situação das mesmas (em espera, confirmadas ou canceladas). Pode ser usado pelo cliente para saber se determinada reserva foi confirmada.
Ator primário:	Cliente
Pré-condição:	Estar *online* e logado no sistema.
Fluxo principal:	1. A partir da tela de informações do estabelecimento, o usuário clica no botão de reservas. 2. O aplicativo exibe uma tela com um botão para solicitar reserva e lista todas as solicitações já realizadas, independente do estabelecimento.

Figura 23 - Aplicativo móvel: Listar reservas

Fonte: Autor

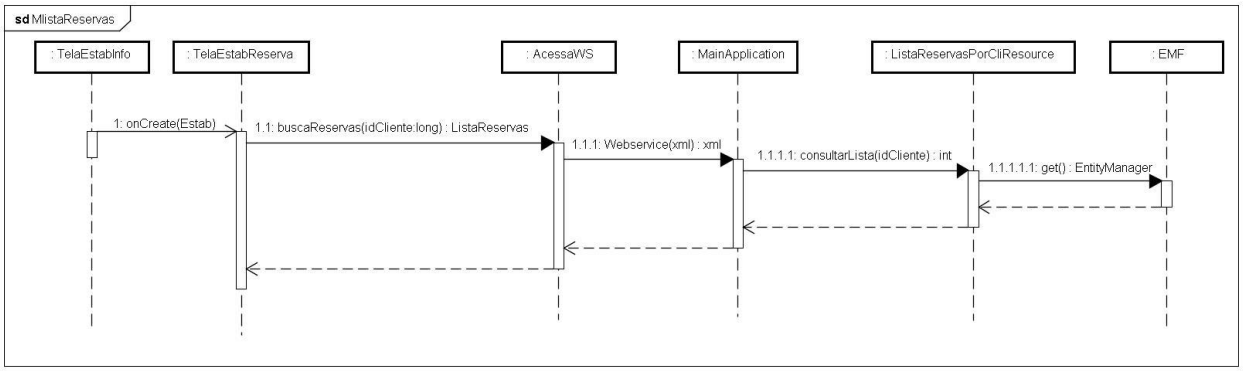

Figura 24 - Diagrama de sequência: Listar reservas do usuário

Fonte: Autor

Tabela 9 - Caso de uso narrativo: Solicitar reserva de mesa

Identificador:	Solicitar reserva de mesa
Descrição:	O cliente realiza uma solicitação para reserva de mesa em determinado estabelecimento, informando o horário da reserva e o número de pessoas.
Ator primário:	Cliente
Pré-condição:	Estar *online* e logado no sistema.
Fluxo principal:	1. A partir da tela que lista as reservas solicitadas, o usuário clica no botão de solicitar nova reserva. 2. O aplicativo exibe uma nova tela com os campos de data da reserva, horário da reserva e número de pessoas. 3. O cliente preenche os campos e confirma a solicitação. 4. O aplicativo envia a solicitação para o estabelecimento.

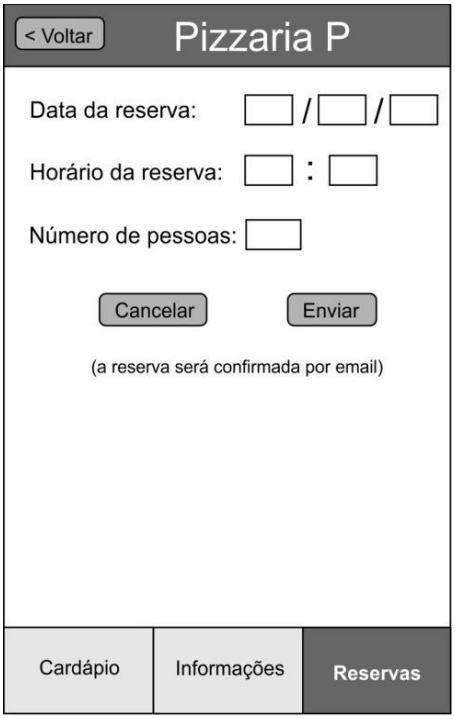

Figura 25 - Aplicativo móvel: Solicitar reserva

Fonte: Autor

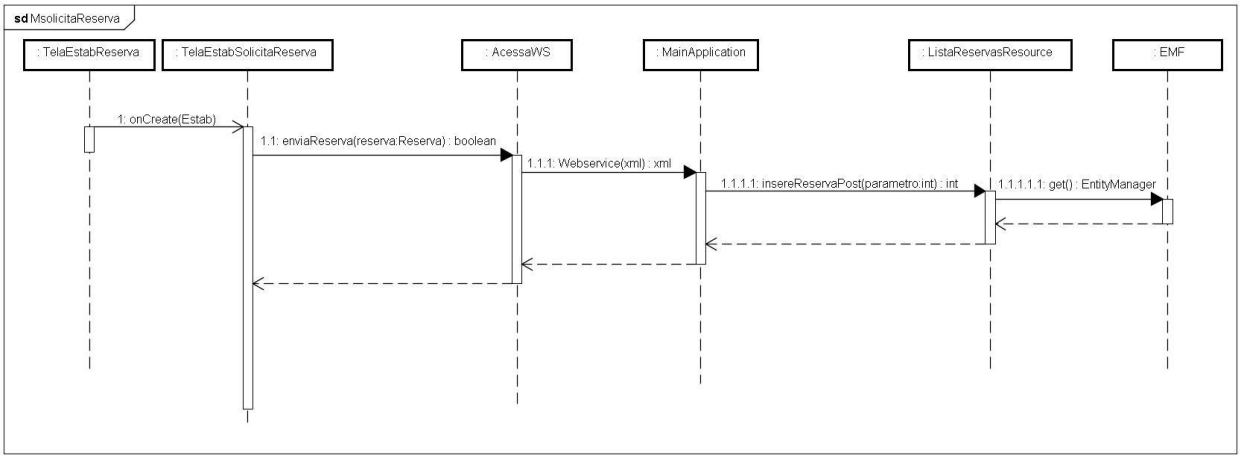

Figura 26 - Diagrama de sequência: Solicitar reserva

Fonte: Autor

Tabela 10 - Caso de uso narrativo: Realizar pedido

Identificador:	Realizar pedido
Descrição:	Permite ao cliente realizar o autoatendimento nos estabelecimentos cadastrados. O processo inteiro compete em realizar um *check in* no restaurante, escolher os produtos através do cardápio e ao final solicitar o fechamento da conta. Nesta etapa é possível também solicitar a presença do garçom através do aplicativo.
Ator primário:	Cliente
Pré-condição:	Estar *online* e logado no sistema.
Fluxo principal:	1. Chegando ao estabelecimento o cliente informa ao aplicativo o código para o *check in* e o número da mesa que irá ocupar, ambas as informações devem ser disponibilizadas pelo estabelecimento. 2. O aplicativo valida o código e o número da mesa junto ao servidor e abre um novo pedido para o cliente, liberando as opções de incluir produtos ao pedido, chamar o garçom e fechar a conta. 3. O cliente acessa a opção de incluir item ao pedido. 4. O aplicativo exibe ao cliente o cardápio do estabelecimento. 5. O cliente escolhe um produto, informa a quantidade desejada, insere observações de preparo e adiciona ao pedido. 6. O aplicativo envia ao servidor a solicitação do produto e volta para a tela de pedido em aberto já com o novo item incluído na lista. O novo item aparece com o status “Incluído” até que o atendente do estabelecimento confirme o pedido. 7. O cliente clica no botão “atualizar”. 8. O aplicativo busca no servidor a situação atual dos pedidos solicitados e exibe a informação atualizada na tela (ao clicar sobre algum pedido, o aplicativo mostra a situação).
Fluxo Secundário 1:	2.1. O código de *check in* é considerado inválido pelo servidor, o aplicativo exibe uma mensagem de erro e volta para a tela de *check in*.

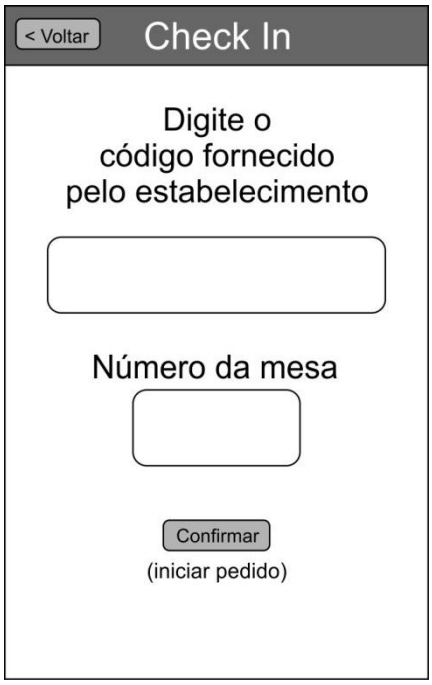

Figura 27 - Aplicativo móvel: Check In
Fonte: Autor

Figura 28 - Aplicativo móvel: Pedido em aberto
Fonte: Autor

Figura 29 - Aplicativo móvel: Adicionar item ao pedido
Fonte: Autor

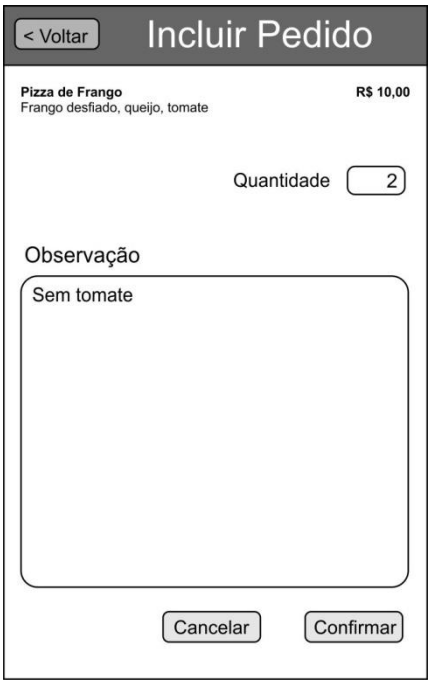

Figura 30 - Aplicativo móvel: Observações do item
Fonte: Autor

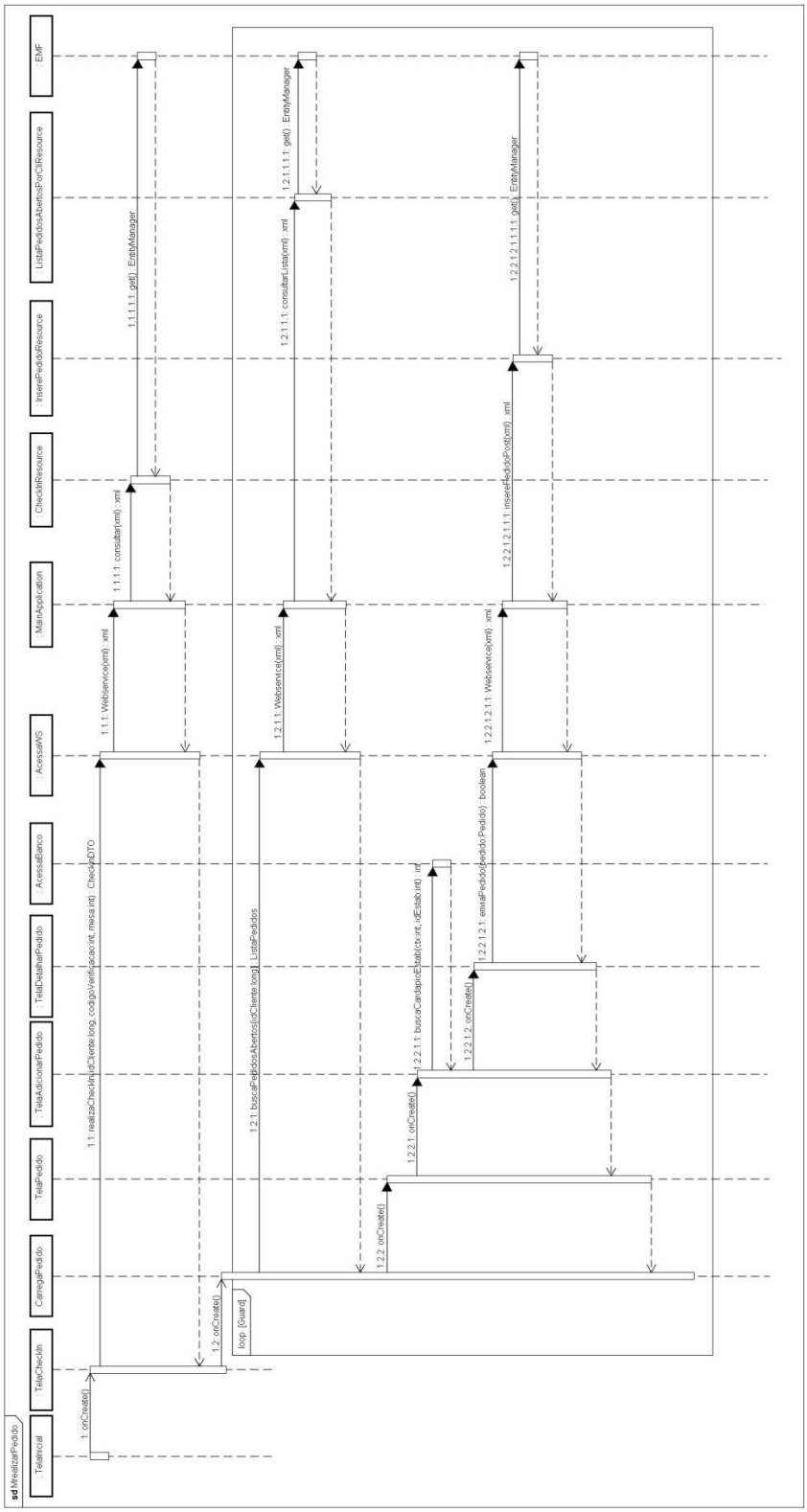

Figura 31 - Diagrama de sequência: Inserir pedido

Fonte: Autor

Tabela 11 - Caso de uso narrativo: Chamar garçom

Identificador:	Chamar garçom
Descrição:	Possibilita ao cliente solicitar a presença de um garçom à mesa diretamente pelo aplicativo.
Ator primário:	Cliente
Pré-condição:	Estar com um pedido em aberto.
Fluxo principal:	1. A partir da tela de pedido em aberto, o usuário clica no botão "Chamar garçom". 2. O aplicativo envia uma solicitação para o servidor, exibe uma mensagem de confirmação para o usuário e permanece na tela de pedido em aberto.

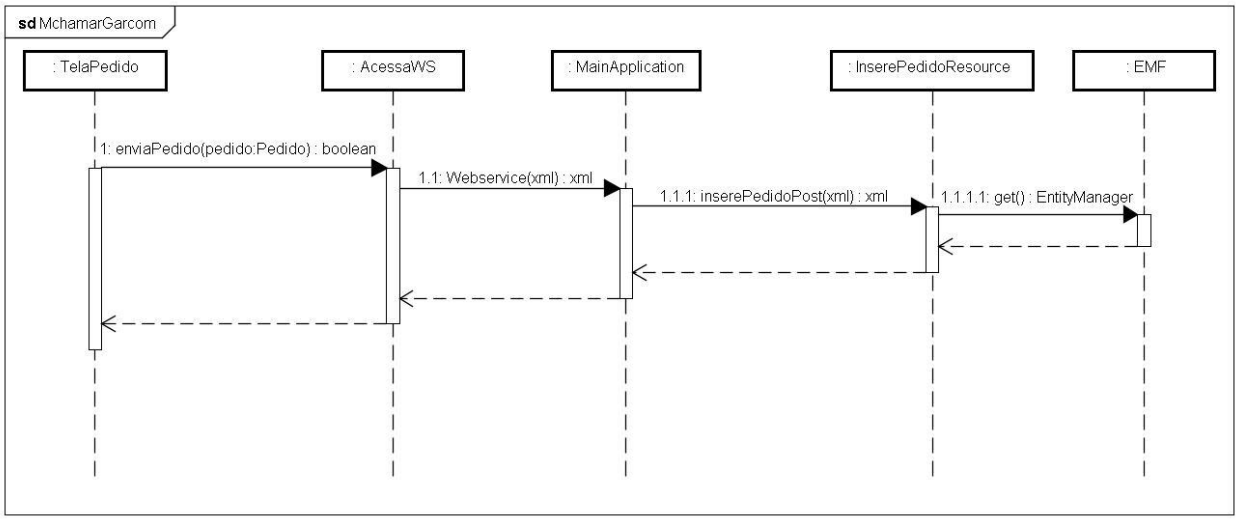

Figura 32 - Diagrama de sequência: Chamar garçom

Fonte: Autor

Tabela 12 - Caso de uso narrativo: Fechar conta

Identificador:	Fechar conta
Descrição:	Possibilita ao cliente solicitar o fechamento da conta diretamente pelo aplicativo.
Ator primário:	Cliente
Pré-condição:	Estar com um pedido em aberto.
Fluxo principal:	1. A partir da tela de pedido em aberto, o usuário clica no botão "Fechar conta". 2. O aplicativo envia uma solicitação de encerramento de conta para o servidor e exibe para o cliente uma tela para preenchimento da nota de avaliação do estabelecimento. 3. O cliente informa a nota e confirma. 4. O aplicativo volta para a tela inicial.

Fluxo secundário:	1.1. O cliente não encerra o pedido e sai do aplicativo. No momento em que retornar ao aplicativo, na tela inicial haverá em destaque uma mensagem avisando que há um pedido em aberto. O cliente não poderá iniciar outro pedido até que encerre o pedido anterior.

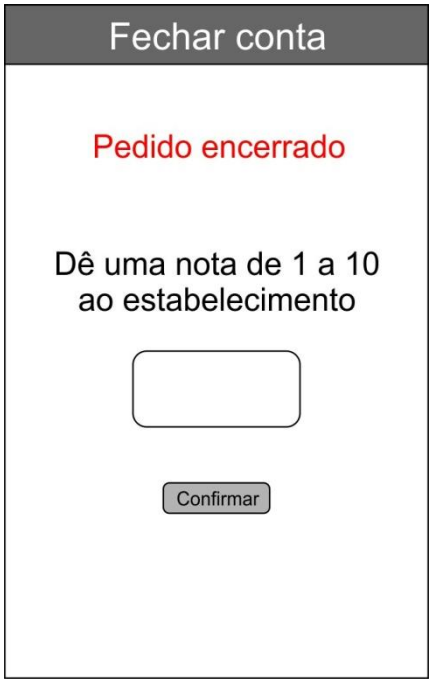

Figura 33 - Aplicativo móvel: Fechar Conta
Fonte: Autor

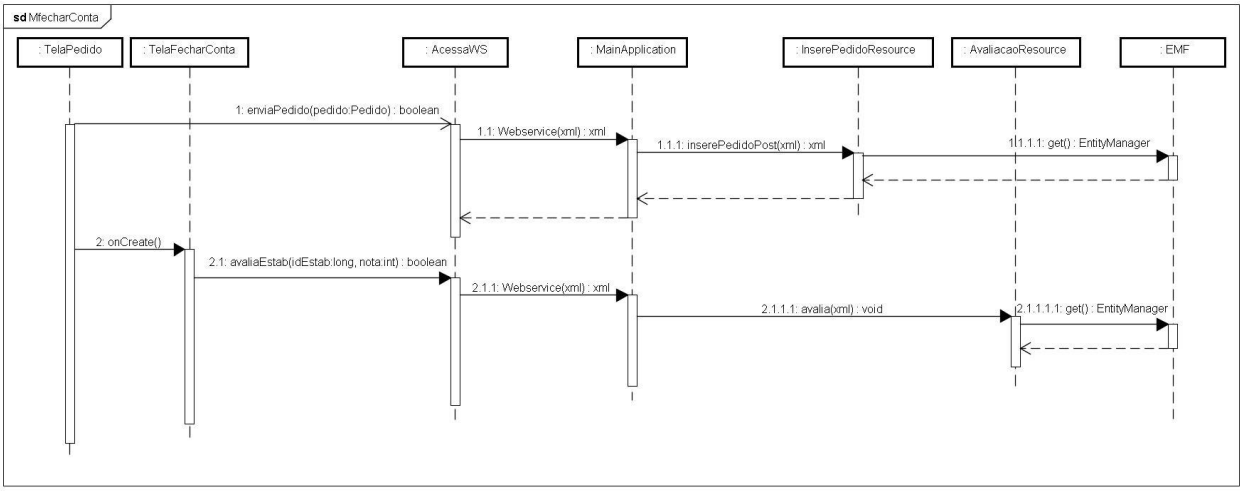

Figura 34 - Diagrama de sequência: Fechar Conta

Fonte: Autor

Tabela 13 - Caso de uso narrativo: Consultar histórico de pedidos

Identificador:	Consultar histórico de pedidos
Descrição:	Possibilita ao cliente acessar a lista com todos os pedidos realizados por ele na aplicação. É mostrado o estabelecimento do pedido, a data e os produtos consumidos.
Ator primário:	Cliente
Pré-condição:	Estar *online* e logado no sistema.
Fluxo principal:	1. A partir da tela inicial, o usuário clica no botão de histórico de pedidos. 2. O aplicativo busca no servidor e exibe na tela a lista de todos os pedidos realizados por este usuário no sistema.

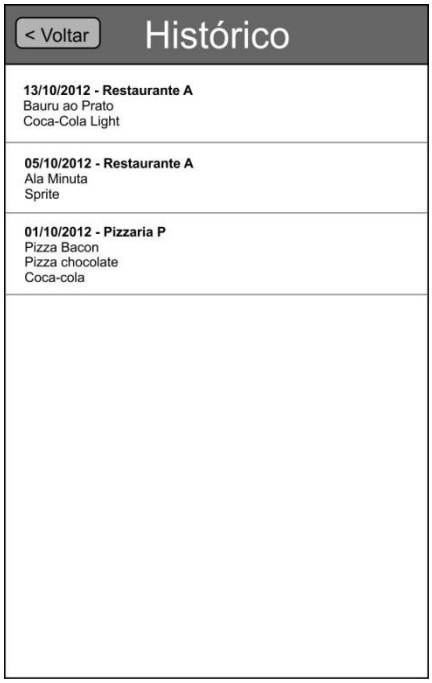

Figura 35 - Aplicativo móvel: Consultar histórico de pedidos
Fonte: Autor

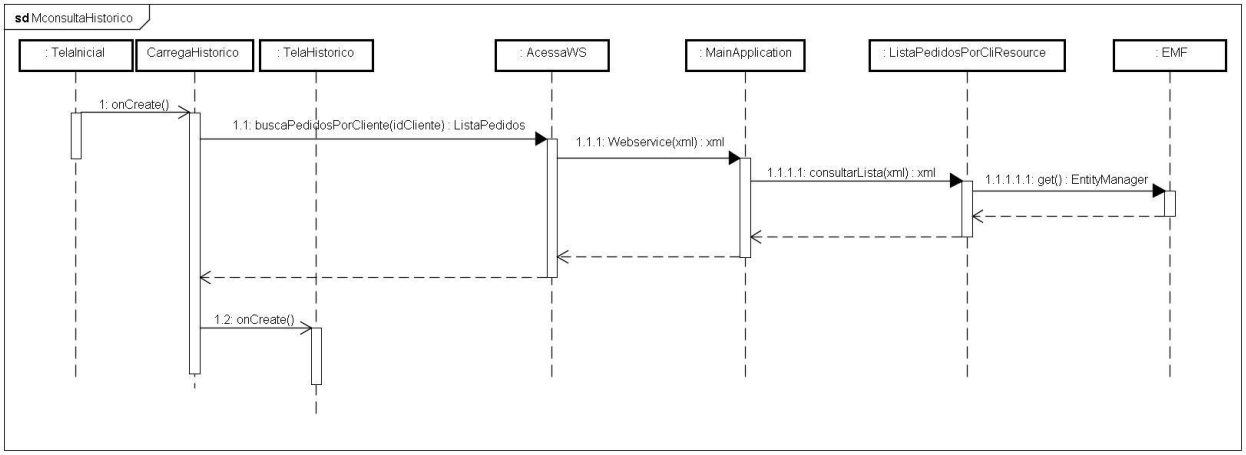

Figura 36 - Diagrama de sequência: Consultar histórico de pedidos

Fonte: Autor

Tabela 14 - Caso de uso narrativo: Pesquisar pratos

Identificador:	Pesquisar pratos
Descrição:	Permite ao usuário uma pesquisa pelo nome do prato. O termo buscado será pesquisado no cardápio de todos os estabelecimentos e resultará em uma lista de pratos.
Ator primário:	Cliente
Fluxo principal:	1. A partir de um botão na tela principal, o usuário clica na opção “Pesquisar pratos”. 2. O aplicativo exibe uma tela onde é possível inserir uma chave de pesquisa. 3. O cliente informa uma chave para pesquisa e confirma a consulta. 4. O aplicativo realiza uma consulta na base de dados local e retorna uma lista com todos os pratos que atendem à pesquisa, junto de seus respectivos estabelecimentos. São listados todos os pratos que contém o texto informado. Junto à informação dos pratos há um botão que serve de link para acessar a tela do estabelecimento, na intenção de buscar mais informações.

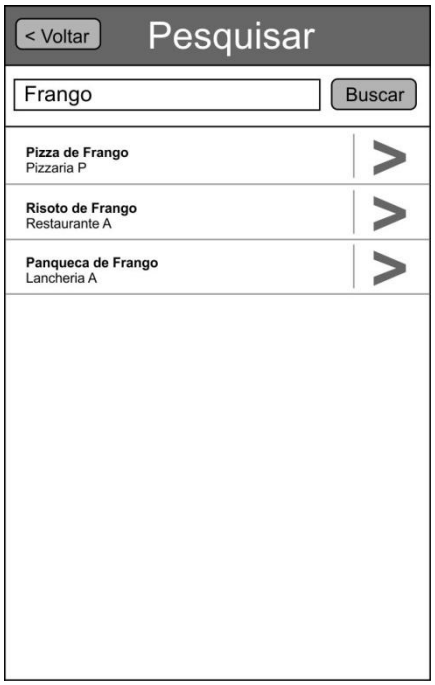

Figura 37 - Aplicativo móvel: Pesquisar prato
Fonte: Autor

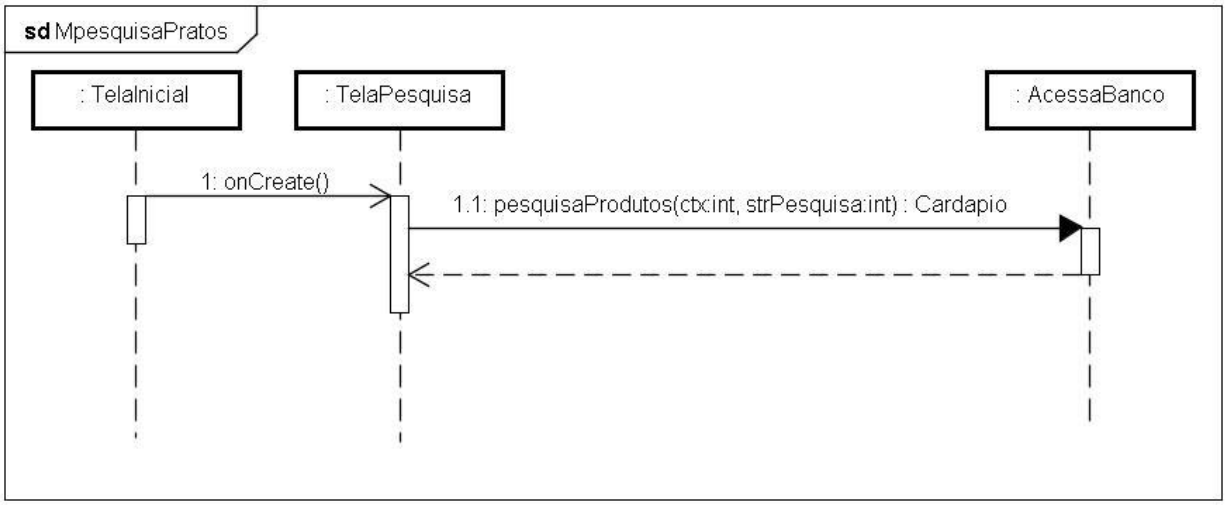

Figura 38 - Diagrama de sequência: Pesquisar pratos

Fonte: Autor

4.3.2 Casos de uso do aplicativo WEB

A partir da Tabela 15 serão apresentados os casos de uso do aplicativo WEB que será utilizado pelos estabelecimentos. A principal funcionalidade deste aplicativo é apresentar as solicitações e permitir ao atendente confirmar a solicitação ou realizar o cancelamento, caso algo não esteja de acordo.

Tabela 15 - Caso de uso narrativo: Realizar login na aplicação WEB

Identificador:	Realizar login na aplicação WEB
Descrição:	Permite ao atendente ou ao administrador realizar login na aplicação WEB a fim de obter acesso às funcionalidades do sistema.
Ator primário:	Administrador, Atendente
Pré-condição:	Possuir um cadastro, realizado pelo administrador do sistema.
Fluxo principal:	1. Ao acessar o site da aplicação, é solicitado o email e a senha do usuário. 2. O atendente (ou administrador) informa o email e a senha e confirma. 3. O sistema valida os dados de login e exibe a tela inicial da aplicação WEB.
Fluxo secundário:	3.1. O sistema exibe a mensagem que o usuário e a senha não conferem e permanece na página de login.

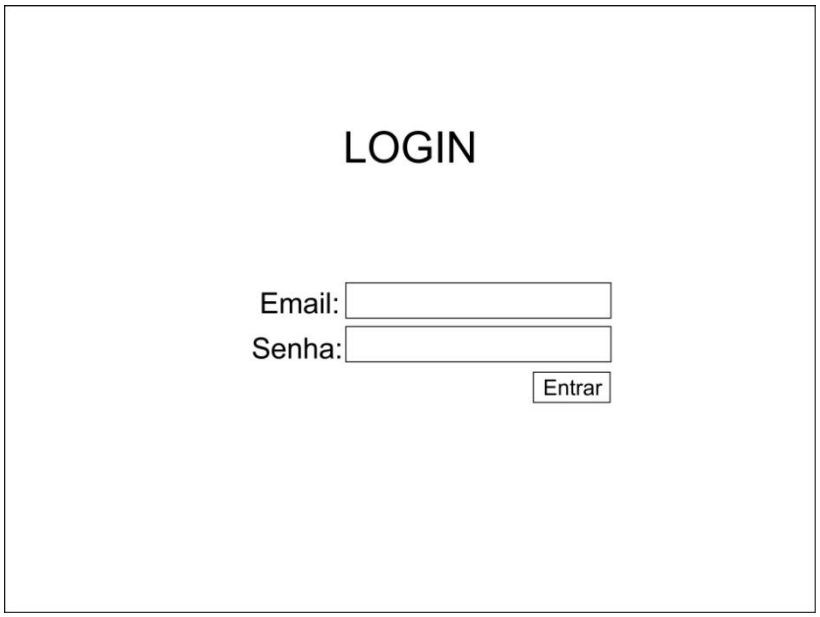

Figura 39 - Aplicativo WEB: Login

Fonte: Autor

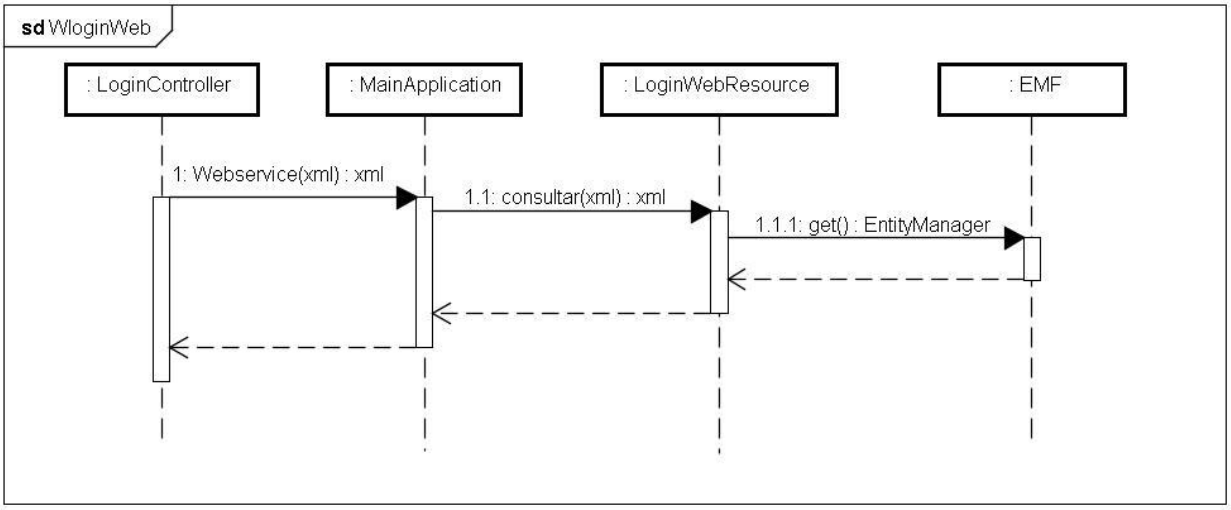

Figura 40 - Diagrama de sequência: Login Web

Fonte: Autor

Tabela 16 - Caso de uso narrativo: Cadastrar estabelecimento

Identificador:	Cadastrar estabelecimento
Descrição:	Ao realizar *login* no aplicativo WEB, é disponibilizado ao administrador uma opção exclusiva para cadastrar estabelecimentos.
Ator primário:	Administrador
Pré condição:	Estar logado na aplicação com o usuário administrador.
Fluxo principal:	1. Na tela principal o administrador acessa a opção de cadastro de atendentes. 2. O sistema retorna uma tabela com os estabelecimentos e seus atendentes cadastrados, possibilitando editar, excluir ou incluir novos registros. 3. O administrador clica no link para incluir novos atendentes, preenche os campos com os dados solicitados e confirma a inclusão.

Figura 41 - Aplicativo WEB: Estabelecimentos cadastrados

Fonte: Autor

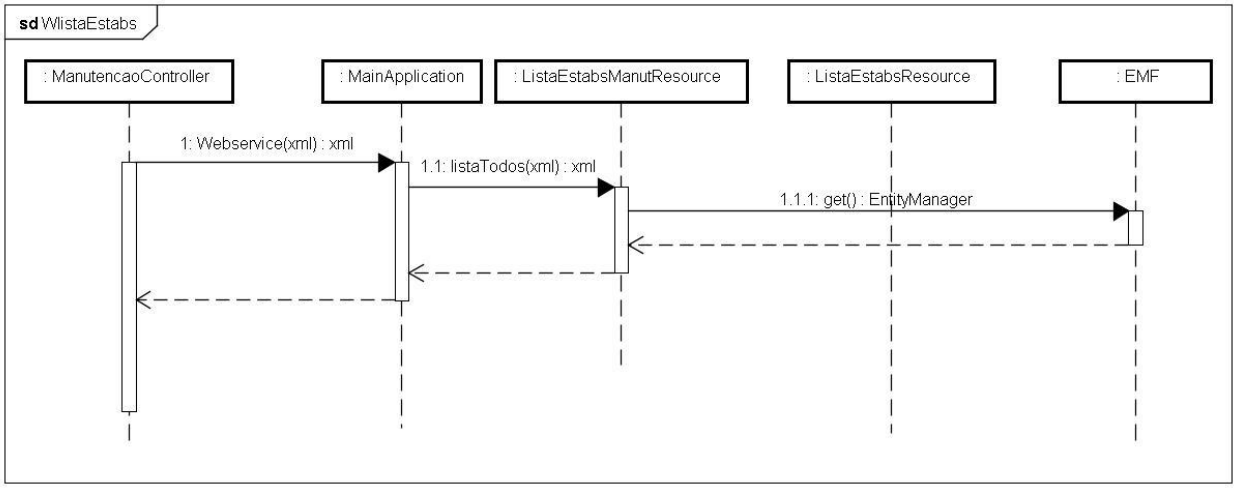

Figura 42 - Diagrama de sequência: Lista estabelecimentos cadastrados

Fonte: Autor

Figura 43 - Aplicativo WEB: Incluir estabelecimento

Fonte: Autor

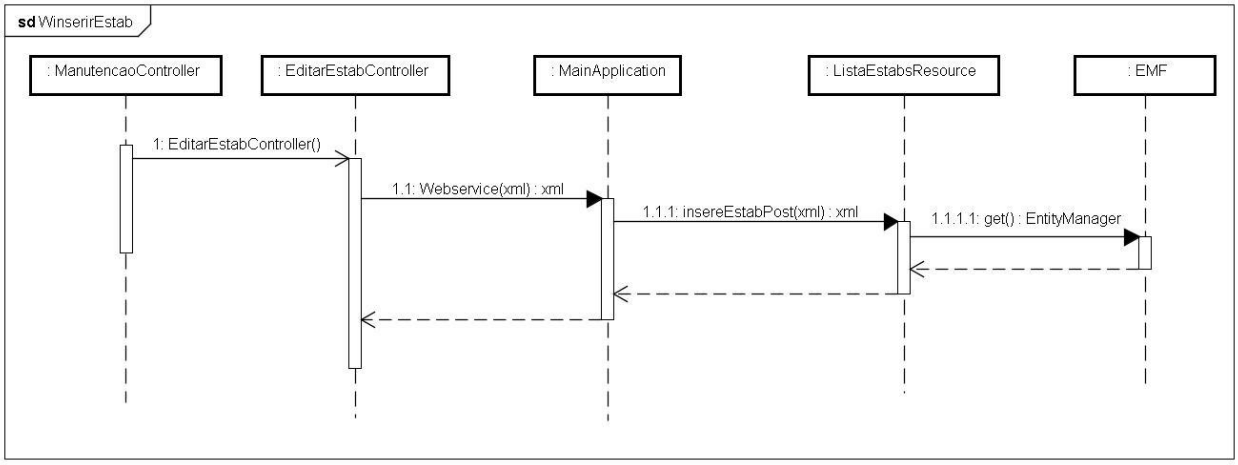

Figura 44 - Diagrama de sequência: Inserir estabelecimento

Fonte: Autor

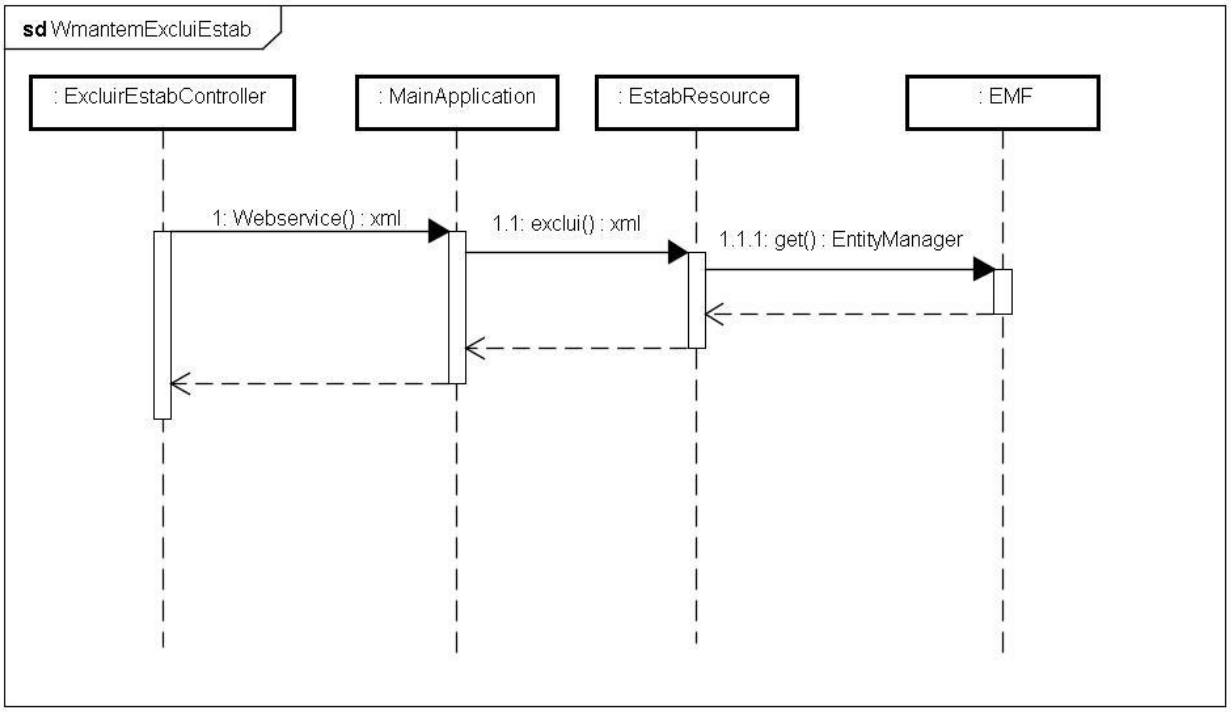

Figura 45 - Diagrama de sequência: Excluir estabelecimento

Fonte: Autor

Tabela 17 - Caso de uso narrativo: Atender solicitação

Identificador:	Atender solicitação
Descrição:	O atendente do estabelecimento confirma uma solicitação realizada por um cliente. Esta solicitação pode ser uma reserva, solicitação de garçom à mesa ou inclusão de um pedido.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. O sistema busca as solicitações realizadas pelos clientes e exibe na tela principal. O atendente analisa a solicitação que está no topo, ou seja, há mais tempo em espera, e confirma a solicitação, dando continuidade ao processo. 2. O sistema elimina a solicitação da lista de solicitações e inclui na lista de "em produção". Caso seja uma solicitação de garçom à mesa ou solicitação de reserva, o sistema não inclui na lista de "em produção".
Fluxo secundário:	1.1. O atendente cancela a solicitação. 1.2. O sistema exibe uma tela para informar o motivo do cancelamento.

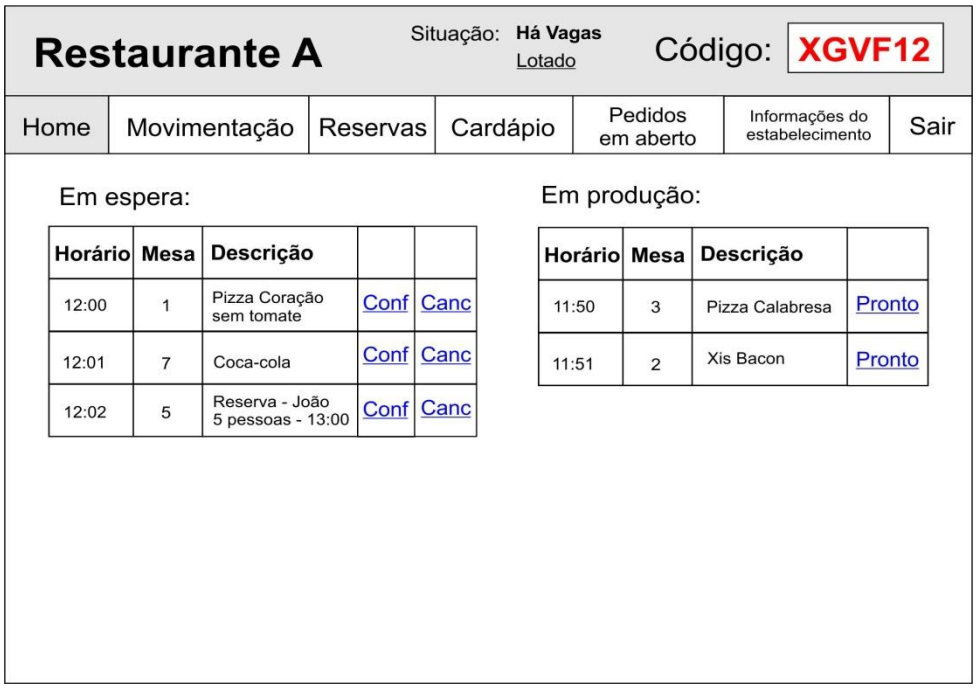

Figura 46 - Aplicativo WEB: Tela principal

Fonte: Autor

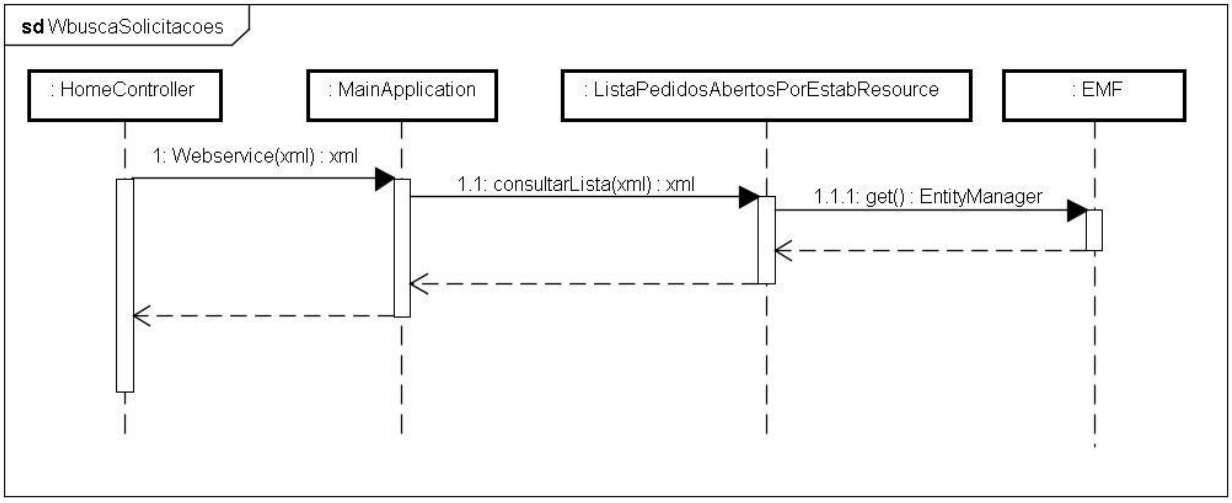

Figura 47 - Diagrama de sequência: Busca solicitações

Fonte: Autor

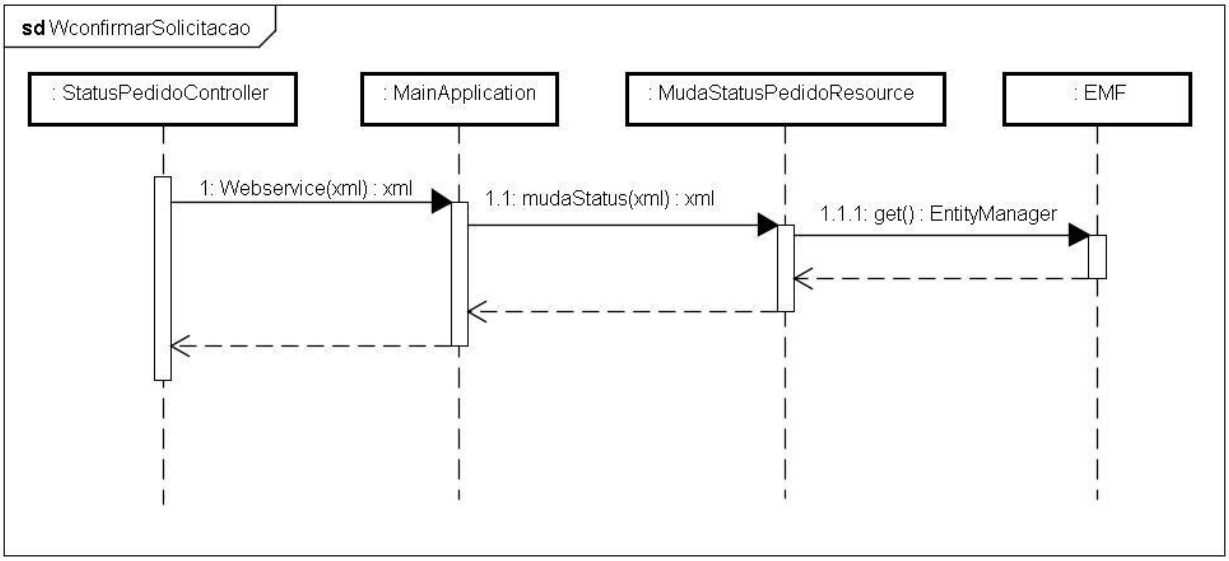

Figura 48 - Diagrama de sequência: Confirmar solicitação

Fonte: Autor

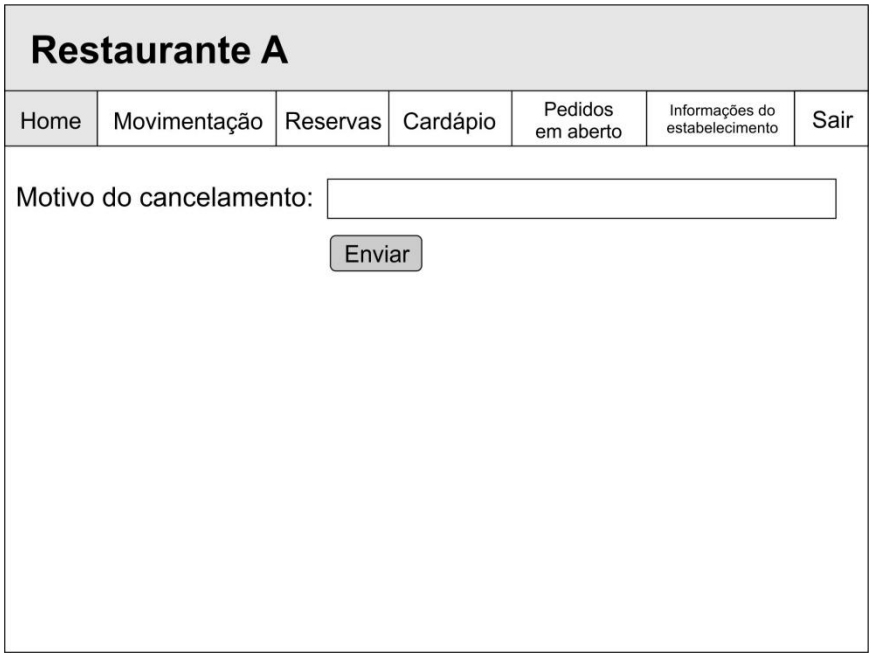

Figura 49 - Aplicativo WEB: Cancelar pedido

Fonte: Autor

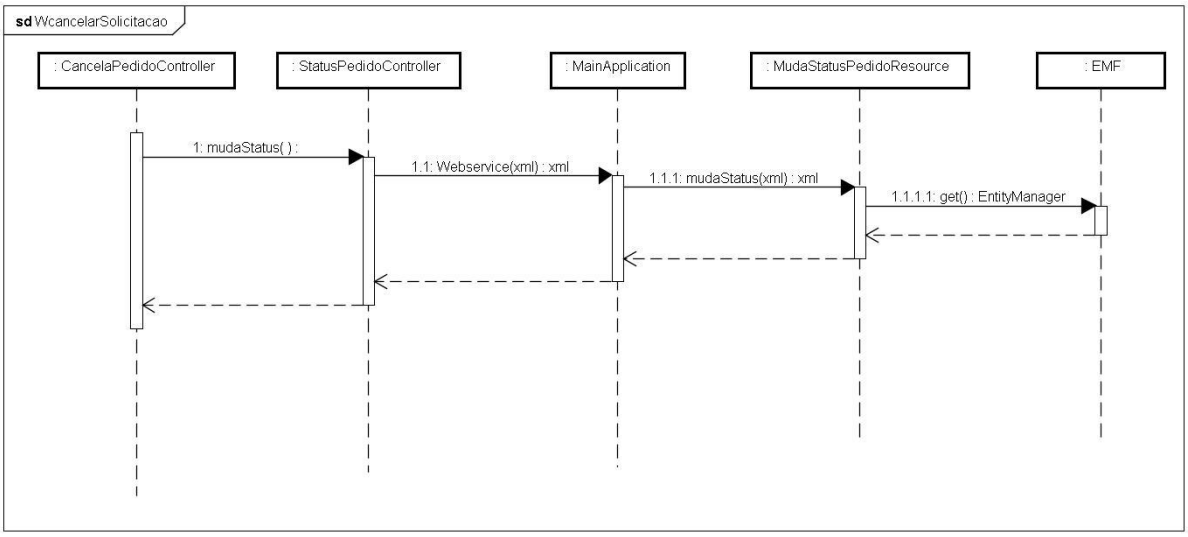

Figura 50 - Diagrama de sequência: Cancelar solicitação

Fonte: Autor

Quando um pedido fica pronto, o atendente realiza o mesmo processo da confirmação de solicitação para informar que o pedido está pronto e será entregue pelo garçom.

Na tela principal do aplicativo Web está o controle do código de verificação que é utilizado pelos clientes na hora de realizar o *check in*. Com uma regra pré-determinada, o aplicativo Web gera um código e envia para o servidor para armazená-lo junto às informações do estabelecimento. Este mesmo código é apresentado na tela para ser repassado aos clientes assim que eles ingressarem no estabelecimento. Na Figura 51 é apresentado o diagrama de sequência desta operação.

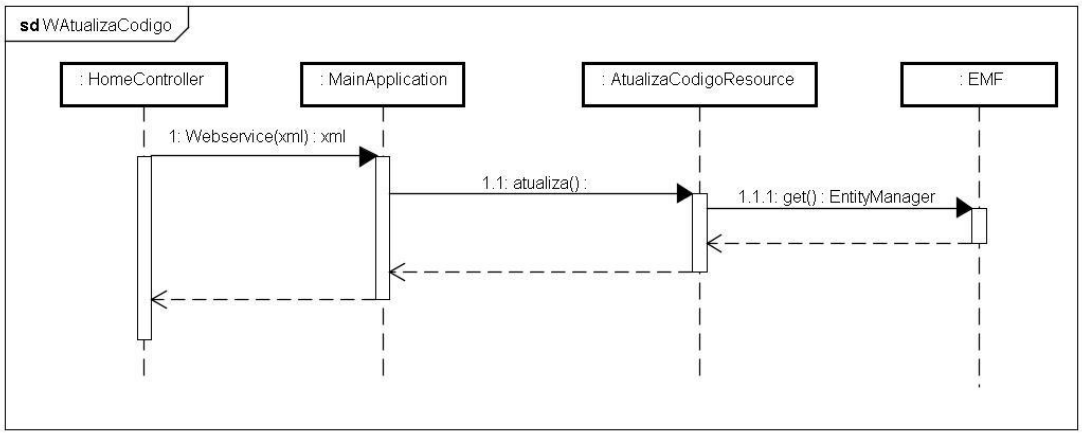

Figura 51 - Diagrama de sequência: Atualiza código de verificação

Fonte: Autor

Tabela 18 - Caso de uso narrativo: Consultar movimentação

Identificador:	Consultar movimentação
Descrição:	Relatório disponível ao atendente com os pedidos realizados no estabelecimento. A seleção pode ser feita por data e são listadas informações sobre cada pedido realizado no período.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. Através de uma opção no menu, o atendente clica no link "Movimentação". 2. O sistema lista todos os pedidos realizados no estabelecimento em questão.

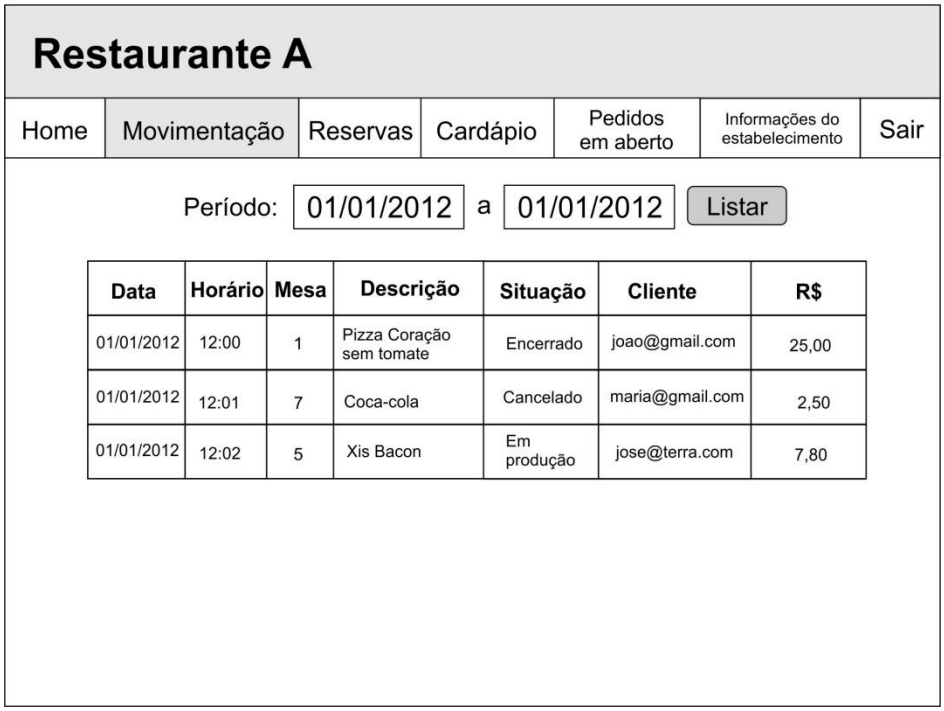

Figura 52 - Aplicativo WEB: Relatório de movimentação

Fonte: Autor

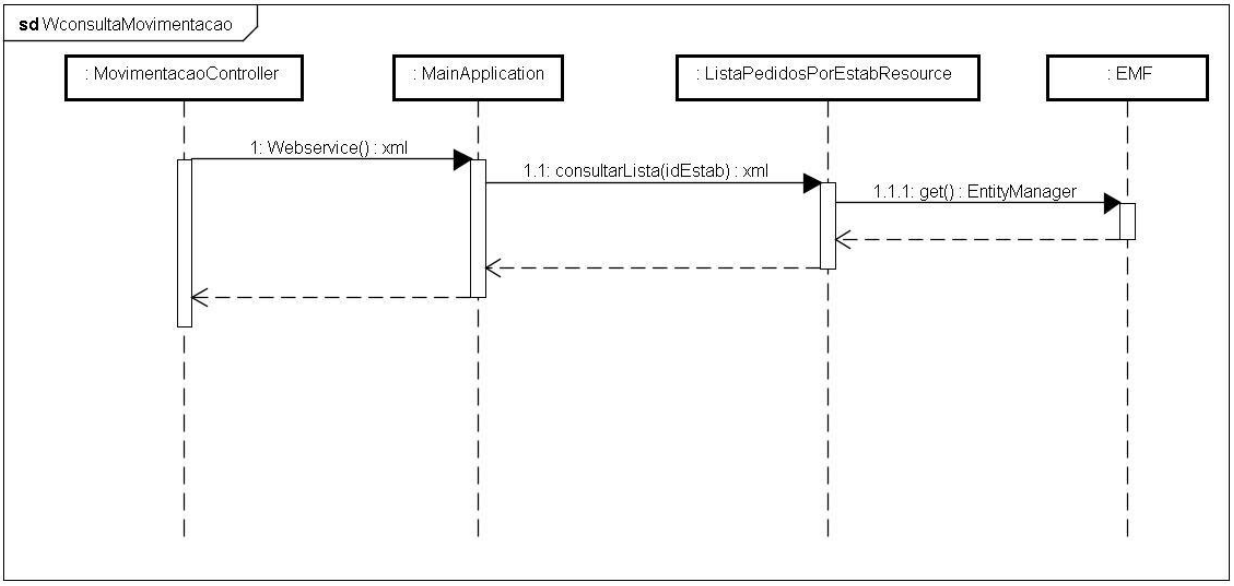

Figura 53 - Diagrama de sequência: Consulta movimentação

Fonte: Autor

Tabela 19 - Caso de uso narrativo: Consultar reservas do estabelecimento

Identificador:	Consultar reservas do estabelecimento
Descrição:	Relatório disponível ao atendente com as reservas realizadas no estabelecimento. A seleção pode ser feita por data e são listadas informações sobre cada reserva solicitada para o período.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. Através de uma opção no menu, o atendente clica no link “Reservas”. 2. O sistema lista todas as reservas realizadas no estabelecimento em questão. 3. O atendente tem a opção de confirmar ou cancelar a solicitação de reserva.

Figura 54 - Aplicativo WEB: Relatório de reservas

Fonte: Autor

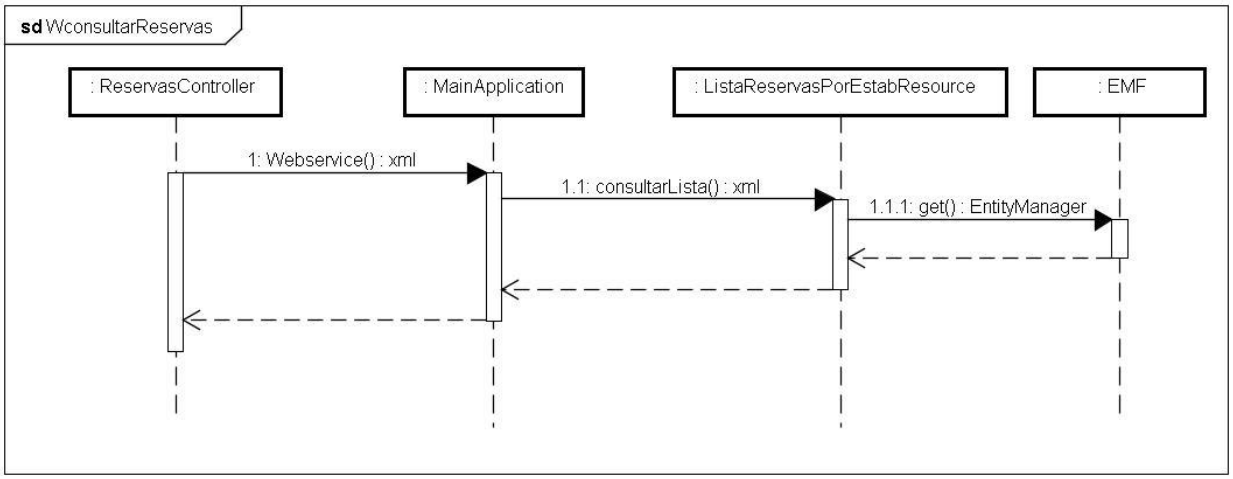

Figura 55 - Diagrama de sequência: Consulta reservas por estabelecimento

Fonte: Autor

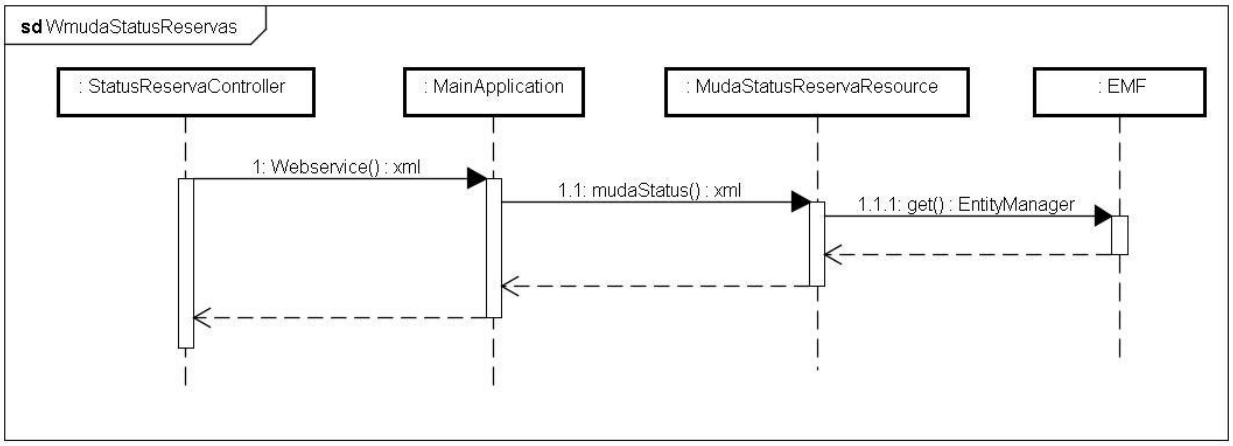

Figura 56 - Diagrama de sequência: Mudar status reserva

Fonte: Autor

Tabela 20 - Caso de uso narrativo: Manutenção do cardápio

Identificador:	Manutenção de cardápio
Descrição:	Permite ao atendente realizar a manutenção do cardápio do seu estabelecimento. É possível incluir, excluir ou modificar itens do cardápio.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. O atendente acessa no menu a opção de manutenção do cardápio. 2. O sistema lista uma tabela com o cardápio cadastrado para aquele estabelecimento. A tabela possui colunas com o nome do prato, os ingredientes e o valor. Todos os campos são editáveis. 3. O atendente modifica o valor de um dos produtos e clica em salvar. 4. O sistema salva os novos dados no servidor e atualiza a tela.

Figura 57 - Aplicativo WEB: Manutenção de cardápio

Fonte: Autor

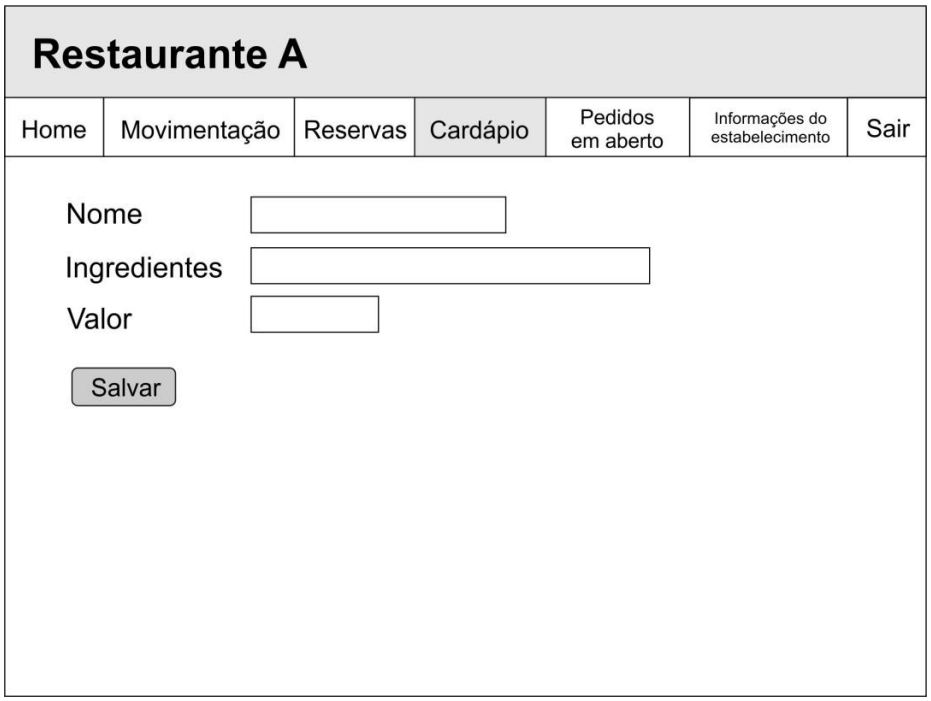

Figura 58 - Aplicativo WEB: Incluir produto

Fonte: Autor

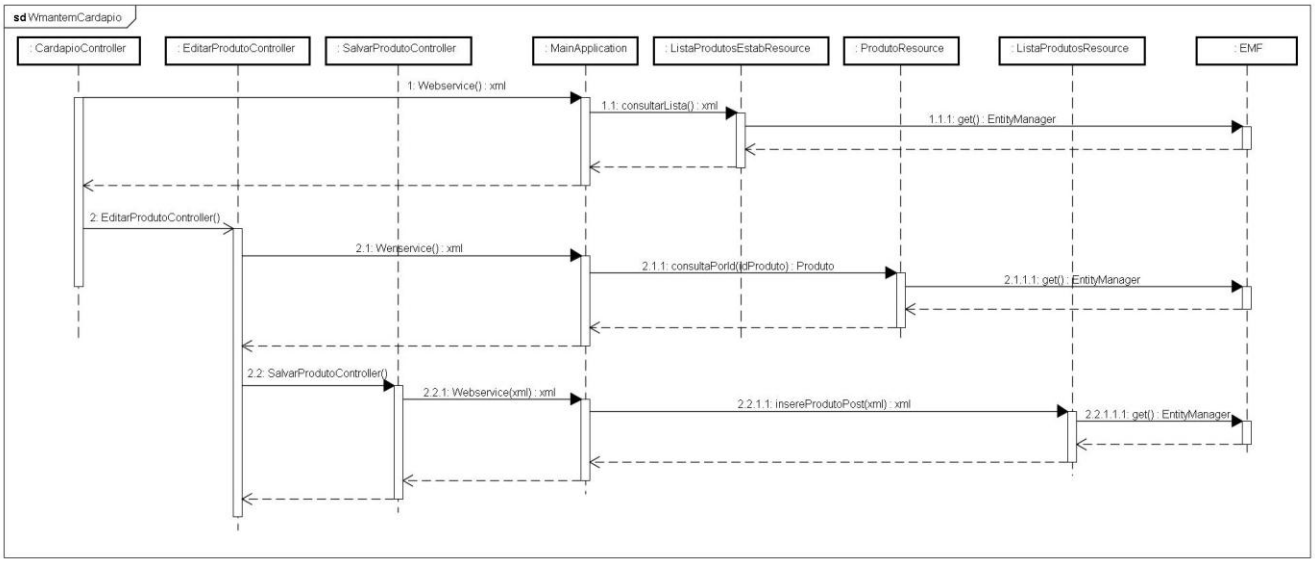

Figura 59 - Diagrama de sequência: Manutenção cardápio

Fonte: Autor

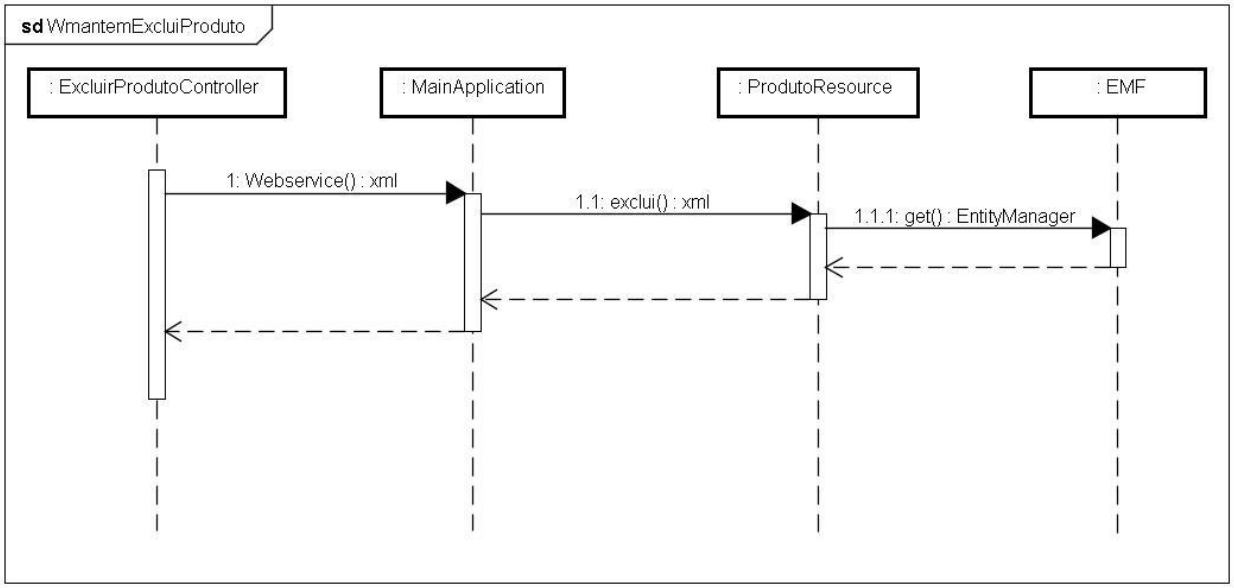

Figura 60 - Diagrama de sequência: Excluir produto

Fonte: Autor

Tabela 21 - Caso de uso narrativo: Encerrar pedido

Identificador:	Encerrar pedido
Descrição:	O atendente encerra o pedido no sistema após uma solicitação do cliente. Com isso o cliente é liberado para iniciar novos pedidos.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. Na tela principal o atendente clica na opção de menu “Pedidos em aberto”. 2. O sistema lista todos os pedidos que estão em aberto no momento, totalizando o valor por usuário. 3. O atendente executa a cobrança junto com o cliente com base no valor apresentado e em seguida clica no botão “Encerrar”.

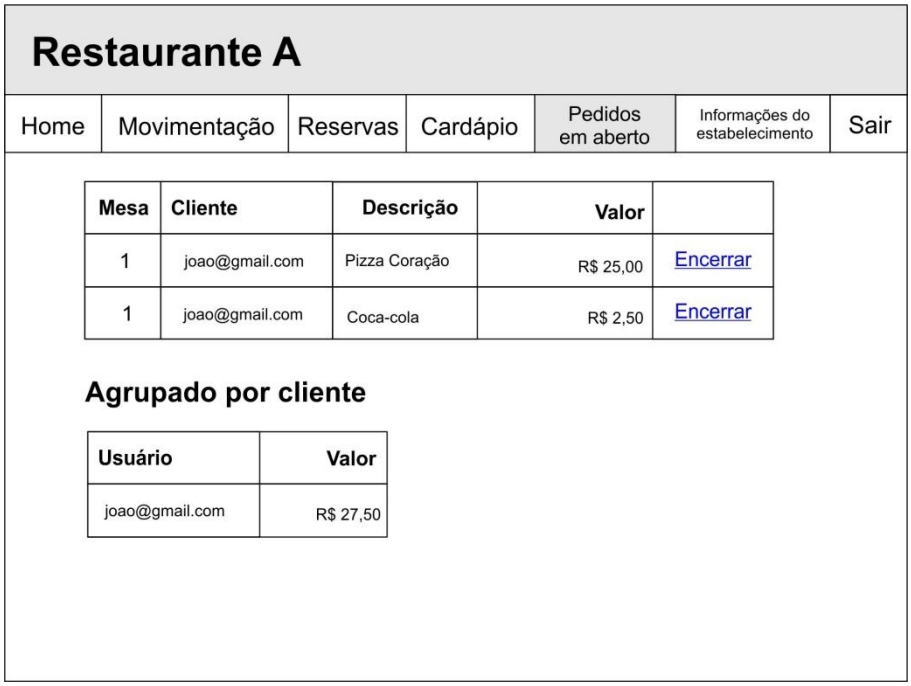

Figura 61 - Aplicativo WEB: Pedidos em aberto

Fonte: Autor

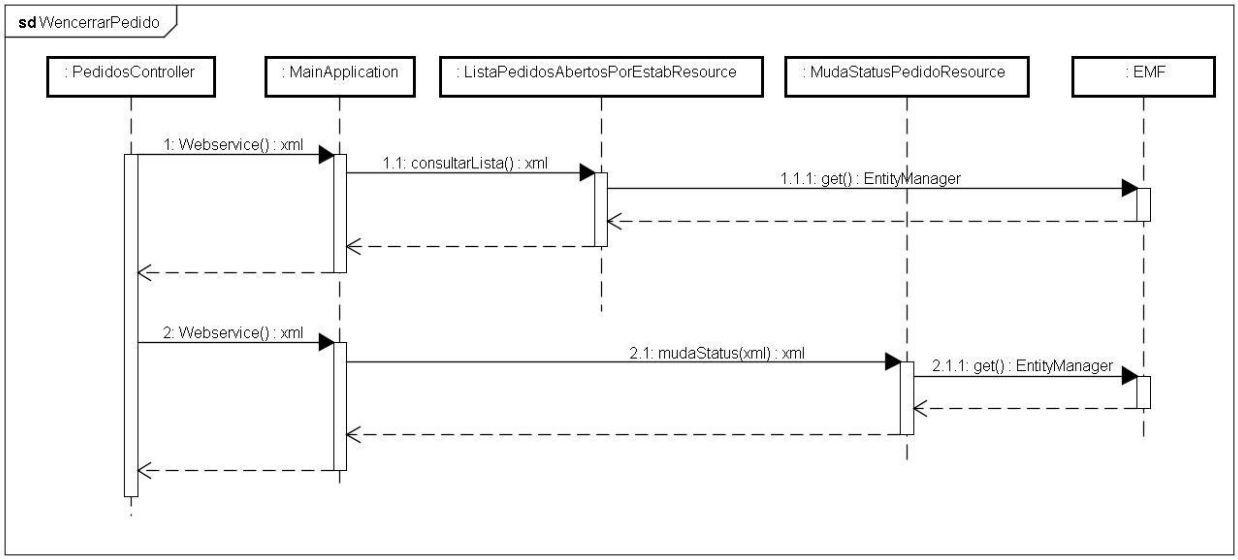

Figura 62 - Diagrama de sequência: Encerrar pedido

Fonte: Autor

Tabela 22 - Caso de uso narrativo: Informar a ocupação

Identificador:	Informar a ocupação
Descrição:	O atendente informa no sistema a situação do estabelecimento quanto à ocupação das mesas. São duas opções, “Há vagas” e “Lotado”. Essas informações estão disponíveis para consulta no aplicativo móvel.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. Na tela principal o atendente clica no link “Lotado” para informar a ocupação máxima do estabelecimento ou “Há vagas” para informar que o estabelecimento possui vagas. 2. O sistema grava essa informação no servidor.

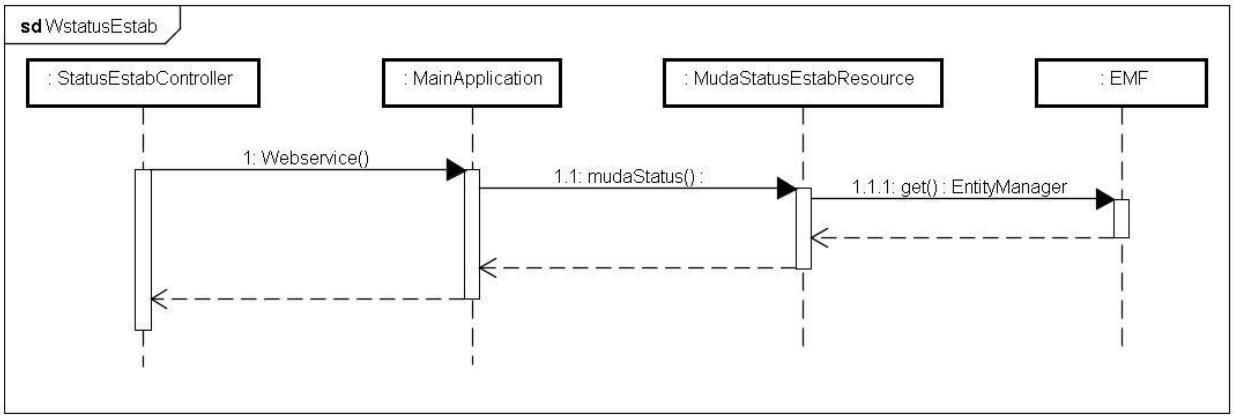

Figura 63 - Diagrama de sequência: Informar ocupação

Fonte: Autor

Tabela 23 - Caso de uso narrativo: Editar informações do estabelecimento

Identificador:	Editar informações do estabelecimento
Descrição:	Permite ao atendente do estabelecimento editar os dados de endereço, telefone, horário de funcionamento e informações gerais que estarão disponíveis para os clientes no aplicativo móvel.
Ator primário:	Atendente
Pré-condição:	Estar logado na aplicação.
Fluxo principal:	1. Na tela principal o atendente clica no link “Informações do estabelecimento”. 2. O sistema retorna uma tela com os dados cadastrados e permite ao atendente alterar os dados. 3. O atendente altera os dados e confirma a edição.

Figura 64 - Aplicativo WEB: Editar informações do estabelecimento

Fonte: Autor

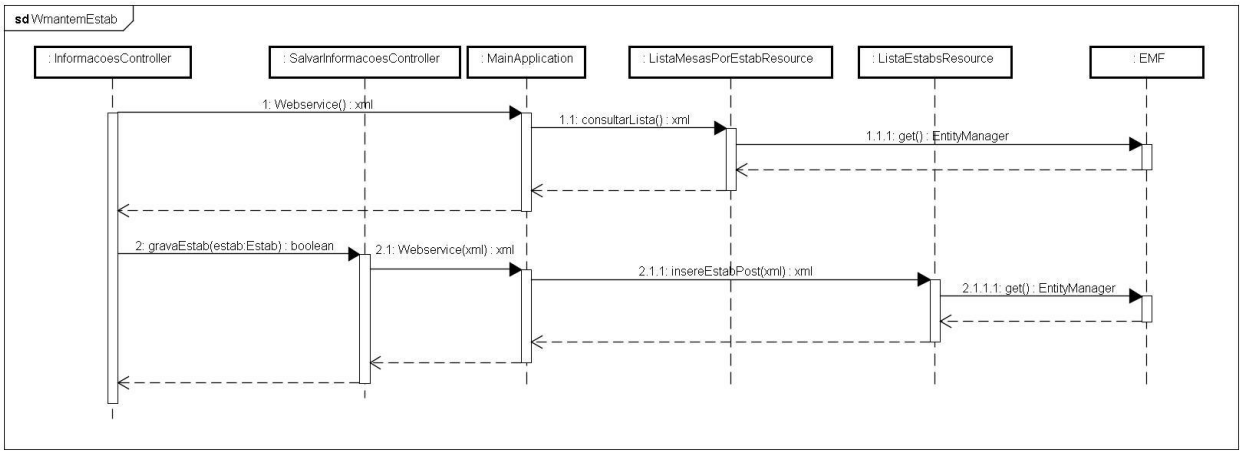

Figura 65 - Diagrama de sequência: Editar informações do estabelecimento

Fonte: Autor

4.4 DIAGRAMA DO BANCO DE DADOS

Como etapa inicial da modelagem do sistema, foram modeladas as classes de dados da aplicação servidor apresentadas na Figura 66 e as classes de dados da aplicação móvel apresentada na Figura 67. Estas informações foram abstraídas do estudo dos casos de uso e servem como base para seguir no projeto do sistema e posterior desenvolvimento.

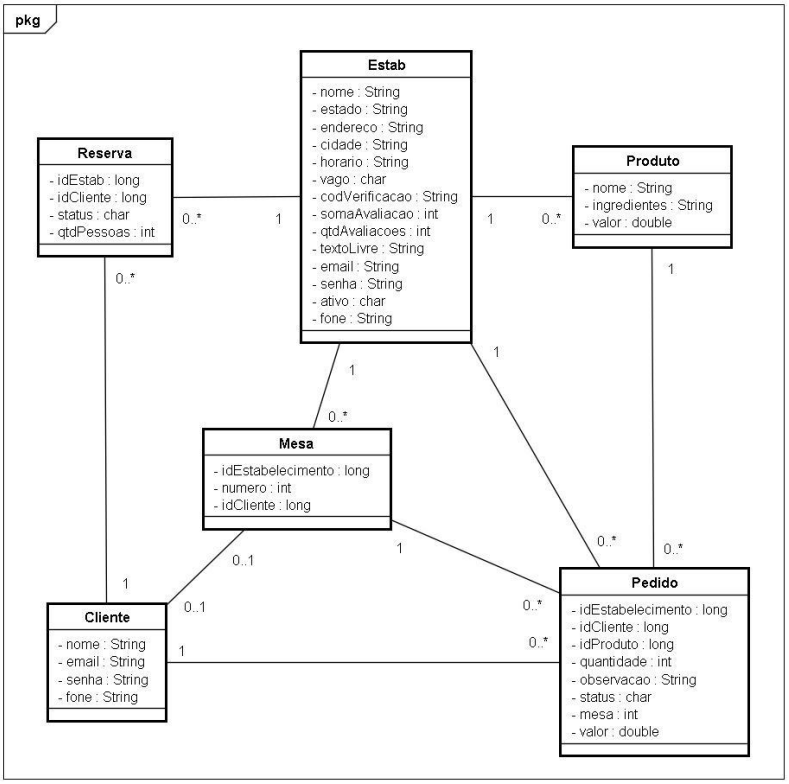

Figura 66 - Diagrama do banco de dados da aplicação servidor

Fonte: Autor

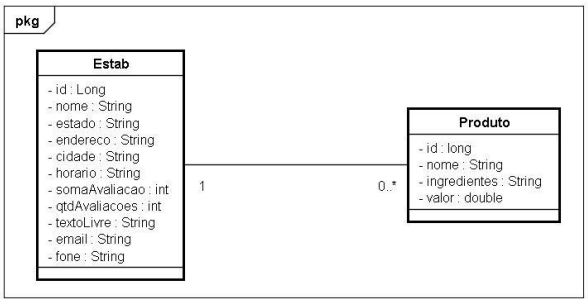

Figura 67 - Diagrama do banco de dados do aplicativo móvel

Fonte: Autor

4.5 DIAGRAMA DE CLASSES

No diagrama de classes da aplicação servidor, apresentada na Figura 68, é demonstrada a classe MainApplication e a sua ligação com o pacote resources. Esta classe recebe todas as solicitações realizadas via *web service* e repassa a requisição para o resource correspondente. Cada classe dentro do pacote resource é responsável por atender a no mínimo um *web service*, buscando as informações na base de dados via classe EMF e retornando um arquivo XML ou simplesmente realizando o processamento solicitado.

Entre as classes resources não há comunicação, elas são invocadas somente pela classe do pacote principal e retornam os dados para esta mesma classe. No diagrama está representado também o pacote entidades, contendo todas as classes responsáveis pelos dados da aplicação. Cada resource possui ligação com uma ou mais classes do pacote entidades, esta ligação está representada de maneira genérica mostrando a ligação de um pacote com o outro.

Na Figura 69 está representado o diagrama de classes do aplicativo móvel, incluindo os pacotes que contém as *activities*, os DAOs, as classes globais, os DTOs e as entidades. *Activity* é uma classe responsável por uma tela da aplicação, realizando algum processamento e possuindo botões ou links que podem ligar a outras telas, formando uma comunicação entre as várias etapas na utilização do aplicativo. No pacote DAO estão as classes utilizadas para acessar o banco de dados local, instalado no próprio dispositivo, na intenção de atualizar os dados ou realizar consultas. A classe AcessaWS do pacote globais é utilizada por diversas *activities* quando há a necessidade de realizar uma comunicação via *web service* com o servidor. Dessa maneira é possível manter as configurações de acesso somente em uma classe, facilitando uma possível manutenção ou alteração na comunicação. A classe cardapioAndroid também é acessada pela maior parte das classes do aplicativo e é essencial para o controle de sessão, mantendo a informação de quando o usuário está logado ou não.

Os DTOs são utilizados pelas *activities* para realizar a transferência de alguns dados específicos entre o aplicativo cliente e o servidor, não caracterizando uma entidade, como as informações de login e *check in*.

No aplicativo WEB, há uma comunicação entre as páginas JSP, responsáveis por exibir as informações em formato HTML, e as classes *controller*, que capturam a informação para ser usada posteriormente pelas páginas JSP. Todas as possíveis comunicações entre as classes estão

representadas na Figura 70. O pacote DTO, assim como no aplicativo móvel, possui as classes que servem para o transporte de dados entre o cliente e o servidor e não são caracterizados como entidades do sistema. As classes do pacote *controller* realizam toda a comunicação com o servidor através do endereço armazenado na classe Global, que é acessada por todos as classes deste pacote. Cada *controller* utiliza uma ou mais classes do pacote entidades, este processo está representada pela ligação geral entre os dois pacotes.

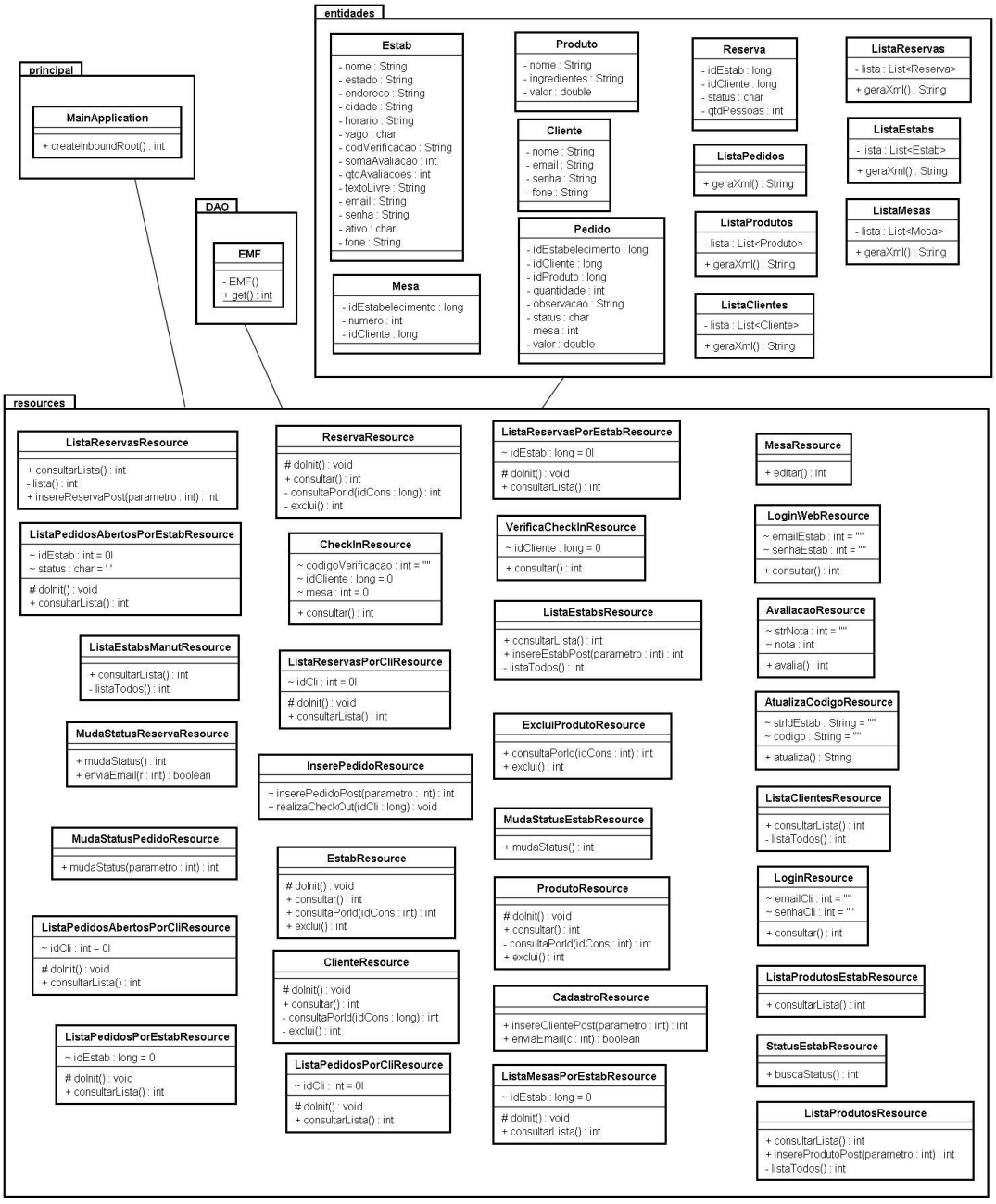

Figura 68 - Diagrama de classes do servidor
Fonte: Autor

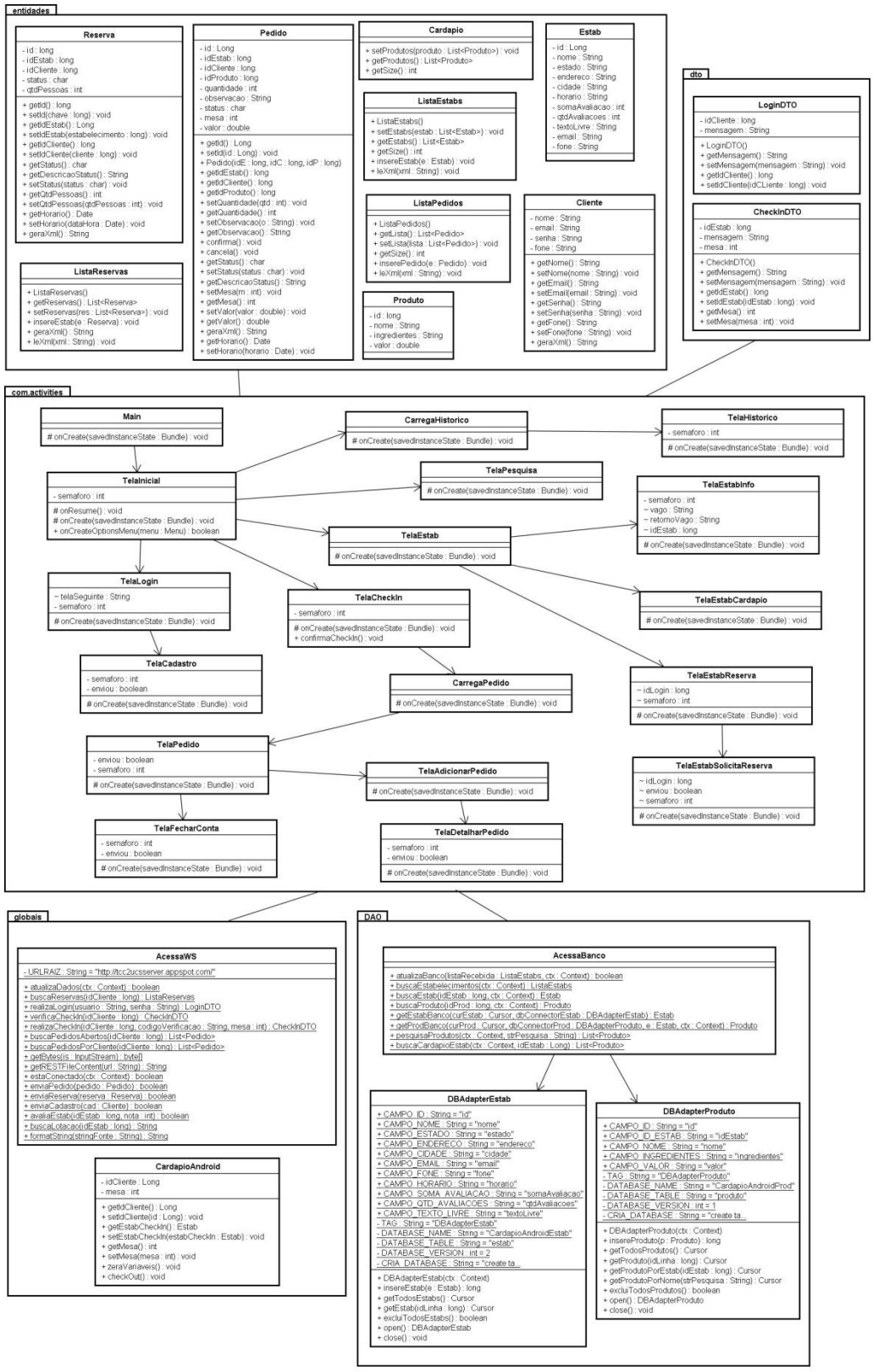

Figura 69 - Diagrama de classes do aplicativo móvel
Fonte: Autor

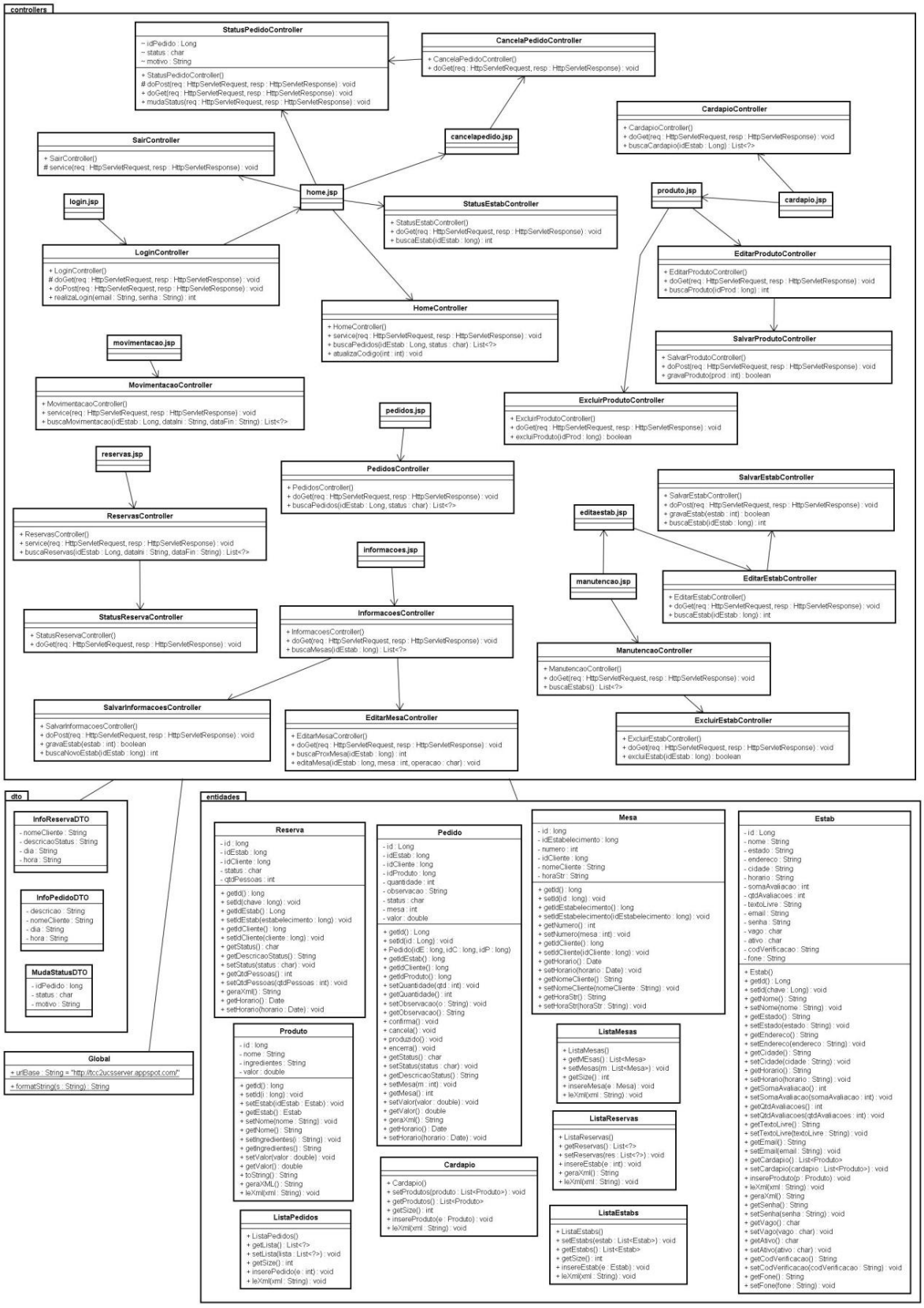

Figura 70 - Diagrama de classes do aplicativo WEB
Fonte: Autor

4.6 ARQUITETURA

O servidor será o repositório de dados e irá gerenciar as informações recebidas ou as consultas solicitadas pelos clientes. O cliente WEB será apenas uma ferramenta para o atendente acessar os dados do seu estabelecimento. Já o cliente móvel contará com uma base de dados local para armazenar determinadas informações dos estabelecimentos. Os dados armazenados no cliente Android, serão uma cópia dos armazenados no servidor, esta duplicação é necessária para possibilitar a consulta *offline* das informações.

Conforme ilustra a Figura 71, a arquitetura física da solução está baseada na conexão dos clientes com o servidor, na intenção de buscar ou inserir dados via *web service*.

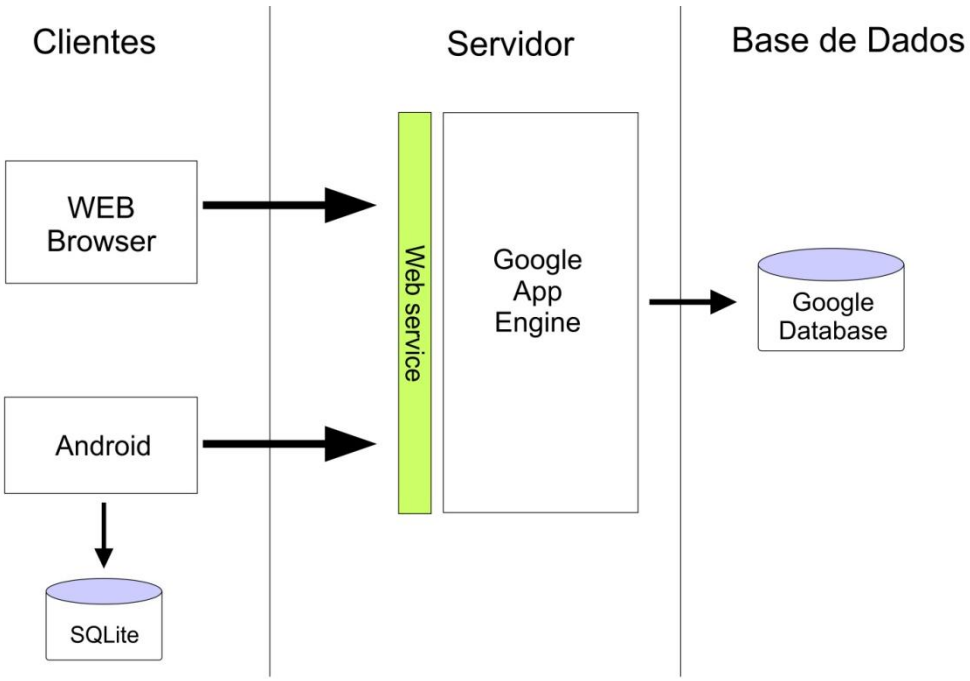

Figura 71 - Arquitetura física da solução

Fonte: Autor

Na Figura 72 está representada a arquitetura com as classes necessárias para a atualização do cardápio no banco de dados local do aplicativo móvel, realizando a consulta dos dados no servidor.

A classe AcessaBanco faz uma solicitação para a classe AcessaWS, informando qual dado quer retornado do servidor. A classe AcessaWS acessa o *web service* disponibilizado pelo servidor, que por sua vez é atendido pela classe MainApplication. Esta classe identifica a URL acessada e repassa a solicitação invocando a classe correta, neste exemplo representada pela classe ProdutoResource. Para buscar os dados na base de dados do Google, é invocada a classe EMF, responsável pela persistência dos dados.

Ao retornar os dados, a classe AcessaWS recebe o arquivo XML e faz a interpretação necessária, passando para a classe Acessabanco o objeto, ou a lista de objetos, neste caso, Cardapio e Produto. Em seguida a classe DBAdapterProduto realiza a persistência desses dados no banco de dados SQLite, contido no Android.

Esta é a representação de uma das várias operações realizadas entre o cliente e o servidor, considerando inclusões e consultas, mas todas seguem o mesmo princípio e os mesmos padrões.

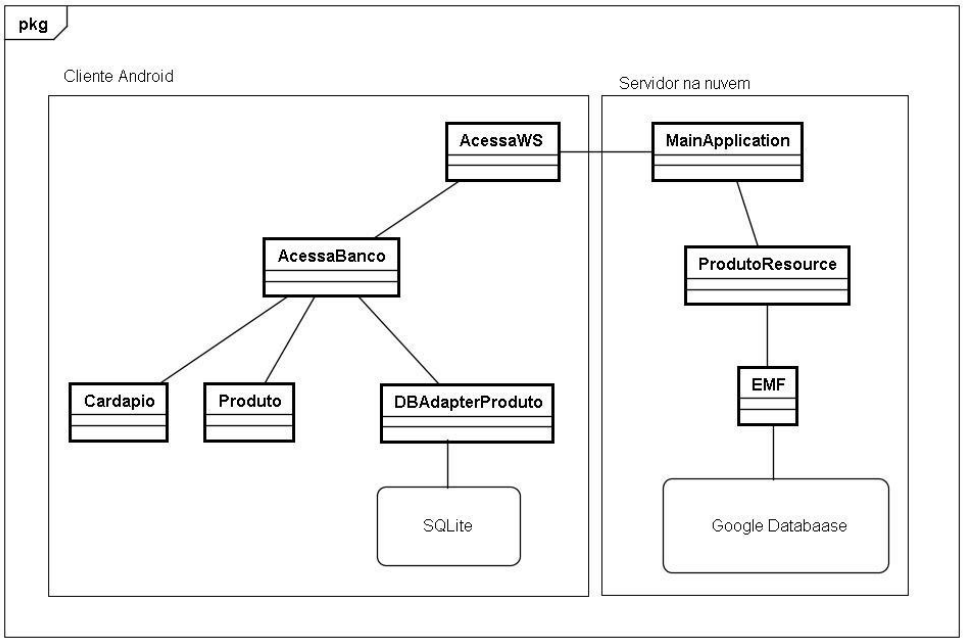

Figura 72 - Arquitetura da comunicação cliente – servidor

Fonte: Autor

5 DESENVOLVIMENTO

Todo o desenvolvimento foi realizado com o uso da IDE Eclipse, aproveitando as ferramentas disponibilizadas pela Google como o simulador de *smartphone*, para testas as aplicações Android, e o simulador do Google App Engine, para testar as aplicações que serão armazenadas na nuvem. O desenvolvimento da solução proposta iniciou pela aplicação servidor, que é responsável pelo gerenciamento dos dados e das requisições realizadas pelas aplicações clientes.

Para atender as requisições via *web service* foi utilizado o framework Restlet (Restlet, 2013) com as bibliotecas necessárias para o ambiente do Google App Engine. Este framework é responsável pelo gerenciamento das várias URLs disponíveis, direcionando cada requisição para uma determinada classe que realiza o processamento e devolve o resultado obtido. Alguns serviços são acessados por meio de consulta, onde é passado por parâmetro uma chave, ou algum outro dado simples, na própria URL. Para inclusões de dados, são esperadas conexões contendo como parâmetro um arquivo XML que é interpretado e suas informações são armazenados no banco de dados do servidor.

Cada solicitação é capturada pela classe MainApplication e conforme a URL acessada, é repassada para uma determinada classe identificada como Resource. A Tabela 24 traz todos os *web services* disponíveis, o cliente que utiliza o serviço e sua respectiva funcionalidade.

Tabela 24 – *Web services* para comunicação

Endereço WEB	**Cliente**	**Funcionalidade**
estab	Móvel/ WEB	Inserir estabelecimento / Listar todos os estabelecimentos ativos
estab_manut	WEB	Lista todos os estabelecimentos cadastrados
estab/{idEstab}	WEB	Consultar/excluir determinado estabelecimento
mesa/{idEstab}/{numero}/{operacao}	WEB	Inlcuir/Excluir/Liberar mesa
mesas_estab/{idEstab}	WEB	Listar as mesas de um determinado estabelecimento
reserva	WEB	Inserir reservas
reserva_estab/{idEstab}/{dataIni}/{dataFin}	WEB	Lista reservas de um estabelecimento

		dentro de um período
reserva_cli/{idCliente}	Móvel	Lista as reservas de um cliente
status_reserva/{idReserva}/{status}	WEB	Altera o status de uma reserva
login/{email}/{senha}	Móvel	Realiza o login do cliente
checkin/{idCliente}/{codVerificacao}/{mesa}	Móvel	Realiza o check in do cliente
verifica_checkin/{idCliente}	Móvel	Verifica se o cliente está com o check in ativo em algum estabelecimento
cadastro	Móvel	Realiza o cadastro do cliente
loginweb/{email}/{senha}	WEB	Realiza o login do estabelecimento
pedidos_cli/{idCliente}	Móvel	Lista todos os pedidos de um cliente
pedidos_estab/{idEstab}/{dataIni}/{dataFin}	WEB	Lista todos os pedidos de um estabelecimento dentro de um período
pedidos_abertos_cli/{idCliente}	Móvel	Lista os pedidos em aberto de um cliente
pedidos_abertos_estab/{idEstab}/{status}	WEB	Lista os pedidos em aberto de um estabelecimento
cardapio/{idEstab}	WEB	Lista o cardápio de um estabelecimento
produto/{idProd}	WEB	Consulta um determinado produto
insere_pedido	Móvel	Insere um pedido
status_pedido	WEB	Altera o status de um pedido
status_estab/{idEstab}/{status}	WEB	Altera o status do estabelecimento (vago ou lotado)
status/{idEstab}	Móvel	Lista o status de um estabelecimento
avaliacao/{idEstab}/{nota}	Móvel	Insere uma avaliação de um cliente para um estabelecimento
codigo_verificacao/{idEstab}/{codigo}	WEB	Altera o código de verificação do estabelecimento

Seguindo o desenvolvimento, foi criado o cliente WEB que será utilizado pelos estabelecimentos, com o objetivo de manter os seus dados e gerenciar as requisições dos clientes.

Para manter a padronização dos serviços WEB, utilizou-se o mesmo framework Restlet implementado no servidor, mas com as bibliotecas voltadas para a criação do cliente na intenção de consumir os serviços disponibilizados. Todas as bibliotecas são compatíveis com o ambiente do Google App Engine, onde está hospedado também o cliente WEB da aplicação. O ambiente na

nuvem do Google permite que seja utilizado qualquer endereço web terminado em ".appspot.com" para o acesso das aplicações nele armazenadas. Também é possível fazer um direcionamento de qualquer domínio pago para a aplicação desenvolvida. Como este projeto tem o objetivo de ser apenas um protótipo, não será criada uma URL comercial para acesso ao aplicativo, ficando assim definida a URL "tcc2ucs.appspot.com" como o endereço para o cliente WEB.

O desenvolvimento do aplicativo WEB foi realizado na linguagem Java utilizando servlets para o controle e preparação dos dados e arquivos JSP para organizar o layout das páginas. Não foi utilizado nenhum framework para o desenvolvimento. Na página consulta de reservas e da movimentação há campos onde é necessário informar uma data. Para facilitar essa entrada de dados foi utilizado o conjunto de bibliotecas jQuery. Utilizando as suas funcionalidades, onde são combinados códigos JavaScript com folhas de estilo, é possível exibir um calendário, o que torna mais intuitiva e fácil a ação de selecionar uma data.

O terceiro e último aplicativo que compõe a solução é o aplicativo móvel para o sistema operacional Android que será utilizado pelos clientes. Segundo informações da Google (Android Developers, 2013), 98,3% dos dispositivos com Android possuem a versão 2.2 ou superior. Com base nesses dados, o aplicativo foi desenvolvido utilizando componentes que possuem suporte nas versões mais utilizadas, não sendo permitida a sua instalação em versões inferiores. Este controle é realizado de forma automática pelo sistema operacional, sendo necessário apenas informar no projeto do aplicativo qual a versão mínima do sistema operacional exigida para a correta execução de todas as funcionalidades.

A transformação dos objetos em uma estrutura XML e sua posterior leitura voltando a ser objetos foram inicialmente projetados para ocorrer com a utilização de um *framework,* onde seu objetivo é reconhecer os atributos do objeto e gerar cada um desses atributos em uma *tag* do arquivo XML. Após alguns testes percebeu-se uma incompatibilidade entre os atributos chave da base de dados do servidor e os dados suportados pelos *frameworks*. Outro problema encontrado foi o alto custo de processamento e o elevado tamanho final da aplicação, quando utilizadas bibliotecas prontas no desenvolvimento do aplicativo móvel. Optou-se então pela criação de um código específico para cada classe, onde há métodos que transformam um objeto em XML e métodos que realizam a leitura de um XML e instanciam uma classe. Dessa forma foi possível

aproveitar o mesmo código nas três aplicações desenvolvidas e eliminar qualquer tipo de incompatibilidade, uma vez que o código foi escrito exclusivamente para este projeto.

Finalizada a etapa de desenvolvimento, foram escolhidos alguns estabelecimentos e seus respectivos cardápios para abastecer o sistema com dados reais, possibilitando assim a realização dos testes que serão apresentados no capítulo seguinte.

6 AVALIAÇÃO DO SISTEMA

Para a realização dos testes de aceitação e avaliação do sistema foram selecionados doze pessoas com o objetivo de utilizar o aplicativo móvel e quatro pessoas relacionadas a algum estabelecimento gastronômico para utilizar o aplicativo WEB. Os utilizadores do aplicativo utilizado pelos estabelecimentos receberam uma breve explicação sobre suas funcionalidades, ao contrário dos usuários do aplicativo móvel, que não receberam qualquer treinamento, simulando uma real situação onde os clientes deveriam se mostrar autossuficientes na utilização do aplicativo.

Durante a utilização do sistema, grande parte das solicitações foi realizada em tempo real, não sendo possível perceber que se tratava de uma aplicação rodando na nuvem, mas em algumas das consultas realizadas pode-se observar uma demora na resposta dos *web services*. Isto acaba se tornando um ponto negativo na utilização de sistemas na nuvem, mas nos testes realizados não representou grande perda quando comparado à quantidade de consultas que retornaram em tempo considerado normal.

Após um período de utilização dos aplicativos, os usuários foram submetidos a um questionário, apresentado no Anexo A (Figura 75), onde constam oito perguntas sobre o sistema, solicitando uma nota de 1 a 5, que buscam avaliar a experiência de utilização de cada indivíduo. Conforme os resultados apresentados na Figura 73 e Figura 74, todas as questões tiveram como maioria notas 4 ou 5, o que mostra a aprovação do sistema. Percebeu-se uma divergência de opinião entre os usuários quanto à velocidade de resposta, o que pode estar ligado com o status da conexão que estão utilizando em determinado momento e pelo fato de o sistema ter a estrutura na nuvem, esse aspecto é notado facilmente.

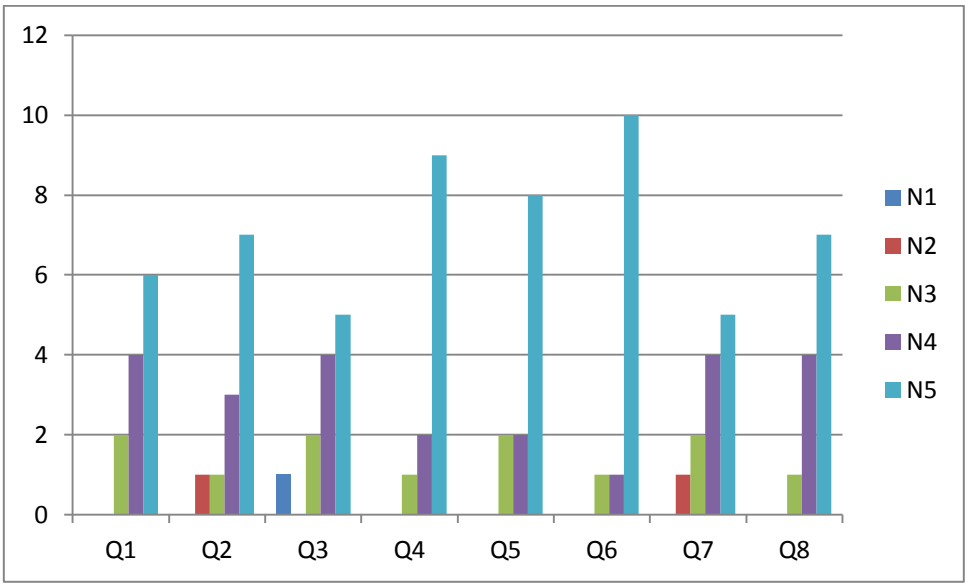

Figura 73 - Resultado da avaliação do aplicativo móvel

Fonte: Autor

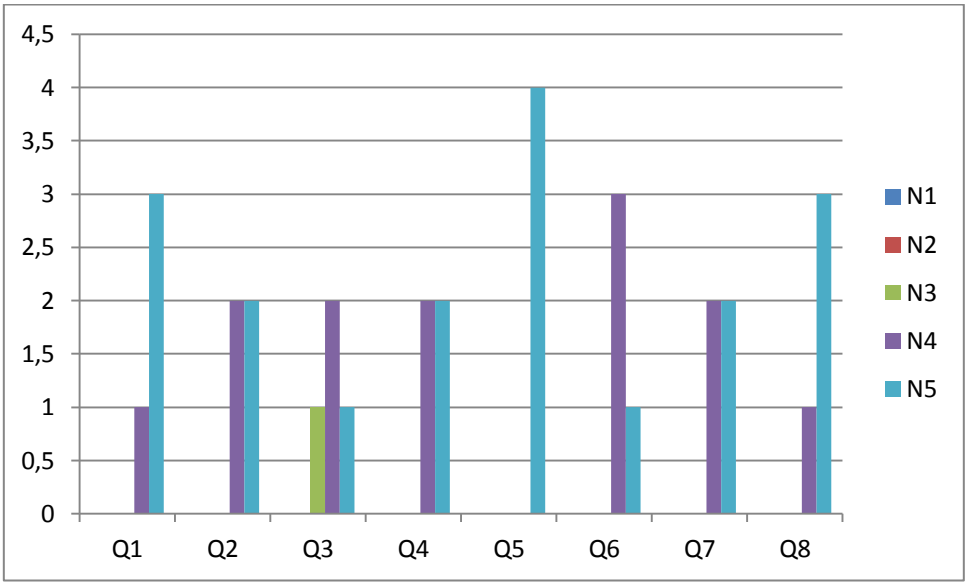

Figura 74 - Resultado da avaliação do aplicativo WEB

Fonte: Autor

7 CONSIDERAÇÕES FINAIS

Novas tecnologias surgem a cada dia, muitas vezes com um propósito definido, mas ainda assim continuam abertas para novas utilizações. Os *smartphones* fazem parte do nosso cotidiano, assim como os inúmeros aplicativos para eles desenvolvidos. Utilizar este meio para auxiliar os usuários em rotinas diárias promete ser uma receita de sucesso para estes aplicativos e é com esta premissa que o projeto em questão ganha confiança para ser desenvolvido.

O autoatendimento em alguns serviços, como os fornecidos em bancos ou aeroportos, já é algo natural e utilizado por grande parte dos clientes desses setores. No setor gastronômico, que envolve um grande número de profissionais, percebe-se um tímido avanço do autoatendimento, apesar do foco desse processo ser justamente a otimização do pessoal empregado e a agilidade no atendimento do cliente. Considerando essa situação, percebe-se um espaço para ser explorado unindo a tecnologia com o atendimento nos estabelecimentos gastronômicos.

Com base nos estudos realizados neste trabalho, concluiu-se que a melhor alternativa para um sistema de autoatendimento em restaurantes é a utilização de processamento na nuvem, com o serviço do Google App Engine para armazenamento e gerenciamento dos dados. A versão de avaliação do produto da Google permite a utilização dos serviços sem custos por um período ilimitado e o espaço de armazenamento desta versão comporta a realização de todos os processos necessários para testar a viabilidade deste projeto. Os aplicativos que irão acessar o servidor de dados serão compostos por um cliente em interface WEB, sendo utilizado pelos estabelecimentos com o objetivo de controlar os pedidos realizados, e um cliente móvel na plataforma Android, para utilização por parte dos clientes.

Na fase de análise foi utilizada a metodologia ICONIX, que se mostrou fundamental na definição do sistema e ao mesmo tempo maleável, pois permite que o analista defina quais etapas são importantes para o projeto em questão. Após algumas etapas já é possível obter uma visão geral e detalhada do sistema, incluindo sua estrutura e suas funcionalidades.

A criação dos protótipos de tela sofreu influência de outros aplicativos disponíveis para diversas plataformas, buscando disponibilizar uma interface intuitiva e simples, sem omitir informações consideradas importantes no processo de autoatendimento. O aplicativo poderá ser utilizado por qualquer pessoa que possua um dispositivo com o sistema operacional Android, e é de suma importância que o aplicativo seja amigável e cause uma boa impressão inicial, fazendo

com que o usuário continue a navegar nas funcionalidades e volte a utilizar o aplicativo em outras ocasiões.

Atualmente já existem no mercado ferramentas disponíveis com aspectos semelhantes aos pretendidos neste projeto. Após os estudos realizados com algumas destas ferramentas, percebeu-se que há vários pontos que podem ser modificados, melhorando a experiência de utilização do usuário. As ferramentas existentes concentram-se em uma utilização totalmente *online* exigindo a criação de um usuário, o que não é necessário quando o cliente deseja apenas realizar uma consulta rápida nas informações dos estabelecimentos. Encontrou-se nessa questão uma deficiência que foi explorada e aperfeiçoada, tornando-se o diferencial em comparação às outras ferramentas. Pretende-se também iniciar a implementação da solução em cidades de menor porte com grande potencial turístico, que atualmente não são contempladas pelas ferramentas estudadas.

O sistema permite também aos estabelecimentos uma utilização parcial do aplicativo, disponibilizando o cardápio e as informações referentes ao estabelecimento, mas sem a opção de autoatendimento. A implementação do autoatendimento pode não ser possível em alguns estabelecimentos, mas isto não os impede de utilizar o aplicativo como uma espécie de cardápio digital, tornando-se uma opção a mais no catálogo de estabelecimentos gastronômicos que o aplicativo irá disponibilizar para seus usuários. Caso o estabelecimento decida adotar o autoatendimento posteriormente, essa funcionalidade pode ser facilmente habilitada.

Para este projeto foram priorizadas funcionalidades que estão ligadas diretamente ao autoatendimento e a fidelização do cliente com o estabelecimento. Confirmando a aceitação do sistema por parte dos usuários e dos estabelecimentos, está previsto como continuidade do projeto a análise e o desenvolvimento de novos recursos para o sistema. Alguns já definidos e listados a seguir:

- Utilização do sinal GPS para localização do estabelecimento mais próximo;
- Integração do sistema com os sistemas ERP de cada estabelecimento para troca de informações referentes ao cardápio e aos pedidos;
- Disponibilização do aplicativo em línguas estrangeiras, visando principalmente eventos internacionais como a Copa do Mundo de 2014;
- Possibilidade de realizar pedidos fora do estabelecimento e solicitar uma tele-entrega;
- Integração com redes sociais;

- Pagamento da conta diretamente pelo aplicativo.

Ao final do projeto, atingiu-se o objetivo de montar um sistema utilizando computação na nuvem e permitir um autoatendimento via *smartphones*. Observando os resultados das avaliações fornecidas pelos usuários, chega-se a conclusão que a solução foi bem aceita e possui os indícios de sucesso, caso seja utilizada para fins comerciais.

REFERÊNCIAS

Amazon. **Amazon Elastic Compute Cloud**. Disponível em <http://aws.amazon.com/pt/ec2/>. Acesso em 01/10/2012.

Android Developers. **Dashboards**. Disponível em <http://developer.android.com/about/dashboards/index.html>. Acesso em 25/04/2013.

BLAHA, Michael; RUMBAUGH, James. **Modelagem e projetos baseados em objetos com UML 2.** Rio de Janeiro: Campus, 2006.

DENNIS, Alan; WIXOM, Barbara Haley. **Análise e projeto de sistemas.** Rio de Janeiro: LTC, 2005.

GOOGLE. O que é o Google App Engine. Disponível em <https://developers.google.com/appengine/docs/whatisgoogleappengine?hl=pt-br>. Acesso em 20/09/2012.

Google Developers. Disponível em <https://developers.google.com>. Acesso em 21/09/2012.

HASHIMI, Sayed; KOMATINENI, Satya; MACLEAN, Dave. **Pro Android 2**. Apress, 2010.

IDC. **Android and iOS Surge to New Smartphone OS Record in Second Quarter, According to IDC.** Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS23638712>. Acesso em 22/09/2012.

IFOOD. Disponível em <http://www.ifood.com.br/delivery/quem-somos/>. Acesso em 01/10/2012.

KOTLER, P. **Administração de marketing:** análise, planejamento, implementação e controle. 4.ed. São Paulo: Atlas, 1995.

LEE, Wei-Meng. **Beginning Android Application Development**. Wiley, 2011

MILLER, Michael. **Cloud Computing. Web-Based Applications That Change the Way You Work and Collaborate Online**. Que, 2009.

NEIL, Teresa. **Mobile Design Pattern Gallery. UI Patterns for iOS, Android, and More.** O'Reilly, 2012.

RESTORANDO. Disponível em <http://porto-alegre.restorando.com.br/pages/about>. Acesso em 29/09/2012.

RICHARDSON, Leonard; RUBY, Sam. **RESTful Web Services**. O`Reilly & Assoc, 2007.
Restlet. Disponível em <http://restlet.org/>. Acesso em 07/01/2013.

ROCHE, Kyle; DOUGLAS, Jeff. **Beginning Java Google App Engine.** Apress, 2009.

ROSENBERG, Doug; STEPHENS, Matt; COPE, Mark C. **Agile Development with ICONIX Process**: People, Process and Pragmatism. Apress, 2005.

SNELL, James; TIDWELL, Doug, KULCHENKO, Pavel. **Programming Web Services with SOAP**. O`Reilly & Assoc, 2002.

SCOTT, Bill; NEIL, Teresa. **Designing Web Interfaces. Principles and Patterns for Rich Interactions.** O'Reilly, 2009.

TABBER. Disponível em <http://www.tabber.com.br/>. Acesso em 12/10/2012.

VELTE, Anthony T.; VELTE, Toby J.; ELSENPETER, Robert. **Cloud Computing. A practical Approach**. McGraw-Hill, 2010.

WAKEFIELD RESEARCH. **Global Survey: Dispelling Six Myths of Consumerization of IT**. Disponível em: <http://www.avanade.com/Documents/Resources/consumerization-of-it-executive-summary.pdf>. Acesso em: 22/09/2012.

WILDE, Erik; PAUTASSO, Cesare. **REST: From Research to practice.** Springer, 2011.

Windows Azure. **O que é o Windows Azure**. Disponível em <http://www.windowsazure.com/pt-br/home/features/what-is-windows-azure/>. Acesso em 02/10/2012.

ANEXO A

Questionário de avaliação da solução

Assinale abaixo como você avalia o sistema utilizado, considerando 0 para a pior nota e 5 para a melhor nota.

1. O sistema é fácil de utilizar?

1	2	3	4	5

2. As funções são apresentadas de forma clara?

1	2	3	4	5

3. Quanto a velocidade de resposta das ações:

1	2	3	4	5

4. As informações disponíveis no sistema foram suficientes para atingir seus objetivos?

1	2	3	4	5

5. O layout utilizado é satisfatório?

1	2	3	4	5

6. A solução apresentada atende as suas necessidades?

1	2	3	4	5

7. Quanto a sua confiabilidade no sistema ao realizar pedidos sem um atendimento pessoal:

1	2	3	4	5

8. Qual a sua avaliação geral para o sistema?

1	2	3	4	5

Figura 75 - Questionário de avaliação da solução

Fonte: Autor