UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

NEWTON MARTINS DE ALMEIDA JUNIOR

USO DE DISPOSITIVOS MÓVEIS PARA O CONTROLE DE PEDIDO EM RESTAURANTE DE PEQUENO PORTE

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO - PR

2015

NEWTON MARTINS DE ALMEIDA JUNIOR

USO DE DISPOSITIVOS MÓVEIS PARA O CONTROLE DE PEDIDO EM RESTAURANTE DE PEQUENO PORTE

Trabalho de Conclusão de Curso, apresentado ao III Curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, campus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Vinicius Pegorini

PATO BRANCO-PR

2015

USO DE DISPOSITIVOS MÓVEIS PARA O CONTROLE DE PEDIDO EM RESTAURANTE DE PEQUENO PORTE

Por

Newton Martins De Almeida Junior

Esta monografia foi apresentada às 16h00 do dia 08 de dezembro de 2015 como requisito parcial para a obtenção do título de ESPECIALISTA, no III Curso de Especialização em Tecnologia Java, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Msc. Vinicius Pegorini
Orientador
UTFPR – Campus Pato Branco

Profa. Dra. Beatriz Terezinha Borsoi
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Banca
UTFPR – Campus Pato Branco

Prof. Msc. Robison Cris Brito
Coordenador do curso de Especialização
UTFPR – Campus Pato Branco

AGRADECIMENTOS

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

RESUMO

ALMEIDA JUNIOR, Newton Martins de. Uso de dispositivos móveis para o controle de pedido em restaurantes de pequeno porte. 2015. 44 f. Monografia de Especialização. III Curso de Especialização em Tecnologia Java. Universidade Tecnológica Federal do Paraná. Pato Branco, 2015.

Os hábitos das pessoas têm sofrido mudanças gerando novos estilo de vida. Uma dessas mudanças é em relação à alimentação e principalmente ao hábito de alimentar-se fora do domicílio. Impulsionado pelo aumento da demanda de clientes ocorre a ampliação da rede gastronômica, que deixou de ser apenas uma opção de lazer para tornar-se questão de comodidade ou até mesmo necessidade. Com uma concorrência cada vez mais acirrada alguns fatores são determinantes para o sucesso do negócio, trabalhar no ramo de alimentação fora de casa parece ser simples, mas, na prática, é complexo e exige uma busca constante pela qualidade dos serviços oferecidos e amplo conhecimento do setor. Diante desta complexidade e com a utilização cada vez mais frequente de sistemas, percebeu-se a possibilidade de desenvolver um sistema específico de gerenciamento de pedidos para restaurantes de pequeno porte, de modo que este seja apenas o primeiro item a ser implantado, apresentando ao gestor facilidades para que futuramente se interesse cada vez mais na sistematização do seu negócio e busque mais ferramentas de gerenciamento a apoio aos processos de administração e tomada de decisão. O aplicativo foi modelado com base nos conceitos da análise orientada a objetos e estruturada para modelagem de dados e desenvolvido utilizando o ambiente de desenvolvimento Java, banco de dados PostgreSQL e o *framework* Spring para o servidor, além do Android Studio para a plataforma móvel.

Palavras-chave: Restaurantes de pequeno porte. Desenvolvimento de *Software*. Gerenciamento de pedidos.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE LISTAGENS

LISTA DE ABREVIATURAS

AOP	*Aspect-Oriented Programming*
CASE	*Computer Aided Software Engineering*
CDI	*Context Dependency Injection*
DER	Diagrama de Entidade e Relacionamento
DFD	Diagrama de Fluxo de Dados
EJB	*Enterprise JavaBeans*
HTML	*HyperText Markup Language*
SGBD	Sistema de Gerenciamento de Banco de Dados
SI	Sistema de Informação
SysML	*Systems Modeling Language*
UML	*Unified Modeling Language*
OHA	*Open Handset Alliance*
PHP	PHP *Hypertext Preprocessor*

SUMÁRIO

1 INTRODUÇÃO

O grande crescimento da rede de gastronomia no Brasil é evidente e pode ser observado com o aumento de estabelecimentos de diferentes características em diversas regiões do mundo. Um exemplo básico que pode ser citado é que ao caminhar em uma rua pode-se deparar com restaurantes servindo pratos de diferentes nacionalidades próximos um do outro.

O setor de alimentação fora do lar vem apresentando nos últimos anos um crescimento vertiginoso. De acordo com Pesquisa de Orçamento Familiar 2008-2009 feito pelo IBGE, o percentual das despesas com alimentação fora do lar já representa 31,1% do total dos gastos com alimentos. Em pesquisa anterior (2002/2003), este número era de 24,1%m, ou seja, mais de um quarto das refeições no Brasil são consumidas fora do lar. Nos grandes centros urbanos passam de um terço (INSTITUTO..., 2015).

Em um mundo no qual a tecnologia se faz cada vez mais presente no cotidiano, o empresário não pode ficar alheio às novidades se quiser permanecer no mercado. De acordo com Lopes (2014) em um restaurante, por exemplo, seu uso por meio da informatização ou automação, pode gerar diversos ganhos empresariais e operacionais e com a alta concorrência no mercado, desenvolver uma boa gestão é o melhor caminho para se destacar.

Os benefícios que o uso de um sistema específico pode gerar posteriormente são recompensadores e farão com que o negócio seja mais próspero. Conforme Gonçalves e Lima (2010), o sucesso de uma organização pode depender de um sistema de informação eficaz, que poderá causar grande impacto nas estratégias da empresa. A empresa, os clientes e/ ou usuários e qualquer indivíduo que interagir com os Sistemas de Informação (SI) pode ser beneficiado com esse impacto.

A não utilização de um sistema para gerenciamento dos restaurantes é decorrente de diversos motivos como: gastos com a aquisição/manutenção do sistema e treinamentos para funcionários; proprietários nunca terem pensado no assunto ou nunca ninguém ter proposto uma ideia; proprietários ou funcionários serem reservados ao novo e preferirem manter do jeito que está, pois funciona.

Assim, verificou-se a possibilidade de desenvolver como trabalho de conclusão de curso um sistema de informação que permita o gerenciamento de

restaurantes de pequeno porte. Para o desenvolvimento da plataforma móvel foi utilizado o ambiente de desenvolvimento Android Studio.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um aplicativo para dispositivos móveis utilizando a tecnologia Android para o gerenciamento de restaurantes de pequeno porte.

1.1.2 Objetivos Específicos

Para atender o objetivo geral deste trabalho, pretende-se:

- Permitir o cadastro de pedidos pelos garçons do estabelecimento;
- Permitir o cadastro de pedidos pelos clientes;
- Permitir a consulta dos pedidos realizados.

1.2 Justificativa

Sabendo que para competir no mercado atual são necessárias constantes atualizações e melhorias contínuas com intuito de propiciar satisfação ao cliente. Teve-se como motivação inicial para trabalhar neste projeto, desenvolver um sistema cujo objetivo é agilizar e melhorar a organização dos processos de estabelecimentos de pequeno porte que atuam no ramo da gastronomia por meio de um sistema de informação. Sendo o sistema o responsável por gerenciar os pedidos por garçons e clientes do restaurante.

Para o desenvolvimento deste sistema, foram empregados e postos em prática os conhecimentos obtidos durante o curso de especialização, além disto, também foram necessários recursos aprendidos na graduação e até mesmo no ambiente de trabalho, pois foram necessários conhecimentos desde a análise de

dados e documentação de sistemas até a implementação de um *software* e banco de dados bem estruturados.

1.3 Estrutura do Trabalho

A estrutura deste trabalho foi dividida em 5 capítulos principais. O primeiro capítulo é composto pela justificativa, os objetivos geral e específicos, e a estrutura do trabalho.

No Capítulo 2 é apresentado o referencial teórico que compreende conceitos relacionados a gestão de restaurantes.

No Capítulo 3 são apresentados os materiais e o método utilizado para documentação, modelagem e implementação do *software*.

No Capítulo 4 apresentam-se os resultados obtidos com o trabalho, detalhando a modelagem do *software* e a descrição dos processos principais do sistema desenvolvido.

Por fim está o Capítulo 5 com a conclusão deste trabalho.

2 REFERENCIAL TEÓRICO

O desenvolvimento deste trabalho foi dividido em duas partes. A primeira tratou de realizar pesquisa sobre as regras de negócio de restaurantes de pequeno porte. Com isso foi possível analisar as vantagens em utilizar um *software* para gerenciar este tipo de negócio, principalmente no que diz respeito aos pedidos realizados. A segunda parte correspondeu ao desenvolvimento de um sistema para gerenciamento de pedidos de pequenos restaurantes.

Esse capítulo explicará o motivo da utilização de cada prática empregada nas duas partes citadas, para posteriormente nos demais capítulos descrever a aplicação destas.

2.1 Gestão de restaurantes

Restaurantes são estabelecimentos comerciais nos quais são preparadas e servidas refeições. No dicionário Aurélio, um restaurante é definido como “lugar onde se servem refeições avulsas a certo número de pessoas”. Desta maneira, o objetivo do restaurante é a preparação de alimentos e também a disponibilização destes aos clientes.

De acordo com Venturini (2010), as origens dos restaurantes ainda não foram totalmente desvendadas, uma vez que a ideia de profissionalizar sua gestão ainda é recente. Entretanto, é possível afirmar que os locais para a venda de alimentos têm uma história secular e bastante interessante.

Conforme Cadorin (2010):

> “Pequenos restaurantes tem uma maneira de trabalhar bastante diferente dos demais comércios, principalmente por serem empresas de pequeno porte. Devido ao grande fluxo de pessoas, vendas de baixo valor e sem obrigatoriedade de registro em nota fiscal, essas empresas apresentam a necessidade de trabalhar com sistemas informatizados, mas sistemas pequenos, de uso rápido e fácil. A utilização de um *software* nestas empresas tem como principais objetivos controlar os registros financeiros, organizar, agilizar os processos e os pedidos de venda”. (CADORIN, 2010).

Entender do ramo não assegura ao empresário que irá conseguir administrar o seu negócio de forma eficiente, uma vez que são necessárias habilidades no

gerenciamento e movimentação de produtos e atendimento aos clientes. Utilizar-se de recursos tecnológicos que ajudem na coordenação de atividades relativas ao negócio, pode ser uma ferramenta de extrema importância para o sucesso da empresa.

Segundo Maricato (2005):

> "Se a casa estiver cheia, é importante apressar o fluxo de serviços: agilizar os pedidos, o serviço e a elaboração da conta, no sentido de obter maior rotatividade das mesas. Obviamente, isso deve ser feito com elegância, procurando servir melhor o cliente, que preferivelmente não deve perceber que se quer apressar sua saída." (MARICATO, 2005, p.97).

É comum o estabelecimento encontrar-se com grande fluxo de clientes, e em paralelo estão os fluxos de informações. E mesmo o gerente sendo capaz, o auxílio para controlar os processos é indispensável. Assim, percebe-se a importância de se informatizar os processos, garantindo a segurança das informações e fazendo com que o trabalho fique menos pesado. E para ajudar o gestor existe o sistema de informação.

Segundo O'Brien e Marakas (2010) há três razões fundamentais comuns a todas as aplicações empresariais da tecnologia da informação, as quais são encontradas nos três papéis vitais que os sistemas de informação podem exercer em uma empresa:

- Suporte de processos e operações de negócios;
- Suporte da tomada de decisão pelos seus empregados e gerentes;
- Suporte das suas estratégias para vantagem competitiva.

Sabendo disso, a empresa estará melhor preparada para atender adequadamente seus clientes, e de maneira organizada possuir um controle interno eficiente e com menos falhas.

3 MATERIAIS E MÉTODOS

Este capítulo apresenta as ferramentas utilizadas durante a análise, modelagem e desenvolvimento do sistema de gerenciamento de restaurantes de pequeno porte e o método aplicado para elaboração do trabalho.

3.1 Materiais

Os seguintes materiais foram utilizados ao longo da construção do sistema:

a) Visual Paradigm 12.0 – ferramenta para modelagem dos diagramas.
b) Java Enterprise Edition – linguagem de programação usada na codificação do sistema, na versão 7.
c) PostgreSQL – banco de dados utilizado, na versão 9.4.
d) Android Studio 1.0 – ambiente de desenvolvimento para plataformas móveis.
e) Framework Spring – *framework* utilizado para o desenvolvimento do servidor.

A seguir são detalhadas as ferramentas utilizadas no trabalho e como foram aplicadas na modelagem ou desenvolvimento do sistema.

3.1.1 Visual Paradigm

Visual Paradigm for UML (VISUALPARADIGM, 2013) é uma ferramenta *ComputarAided SoftwareEngineering* (CASE[1]) com várias opções de modelagem com diagramas da UML e que também oferece suporte a diagramas de requisitos *Systems ModelingLanguage* (SysML) e a diagramas DER.

A ferramenta possui um bom ambiente de trabalho, o que facilita a visualização e manipulação do projeto de modelagem. É uma ferramenta comercial e também oferece transformações específicas para códigos-fonte de algumas linguagens de programação como, por exemplo, C++ e Java.

[1] Uma ferramenta CASE é um software que auxilia no ciclo de desenvolvimento de um sistema, desde a fase de análise, fase de testes, desenvolvimento e apoiam na manutenção de metodologias.

Como citado anteriormente o *Visual Paradigm* é uma ferramenta comercial que possui várias versões, as quais variam de valor, porém há também uma versão gratuita que suporta todos os diagramas da UML, como o de classes, casos de uso, sequência e comunicação, elaborados neste trabalho.

Esta ferramenta é ideal para engenheiros de *software*, analistas de sistemas, e arquitetos de sistemas que estão interessados em criação de sistemas em larga escala e necessitam de confiabilidade no desenvolvimento orientado a objetos.

3.1.2 Java

Java é uma linguagem de programação que começou a ser desenvolvida nos anos 1990 por uma equipe de programadores chefiada por James Gosling, na Sun Microsystems, conglomerado norte-americano da área de informática. Por trás de todo o sucesso do Java está um projeto de pesquisa corporativo batizado de Green, que apostava na convergência dos computadores com outros equipamentos eletrodomésticos. Todavia, conforme Croce Filho e Ribeiro (2010), este projeto passou por dificuldades, uma vez que o mercado de dispositivos eletrônicos inteligentes, direcionado ao consumo popular, no início da década de 1990, não progrediu tão rápido como a Sun havia previsto, correndo risco de ser cancelado. Entretanto, por uma feliz coincidência a Word Wide Web se popularizou e a Sun viu a linguagem Java ser utilizada para adicionar conteúdo dinâmico às páginas da web, como interatividade e animações.

As aplicações Web estão evidentemente mais presentes no mercado e atualmente o seu desenvolvimento já representa uma fatia significativa da produção em organizações de desenvolvimento de softwares. O crescimento vertiginoso de aplicações Web interferem diretamente em vários aspectos da vida das pessoas, uma vez que a linguagem Java conforme cita Cadenhead e Lemay (2005), inicialmente usada para criar programas simples em páginas Web, pode ser encontrada hoje em dia em: servidores Web, banco de dados relacionais, computadores mainframe, telefones, telescópios orbitais, assistentes digitais pessoais, “smartcards” do tamanho de um cartão de créditos.

Conforme Cadenhead e Lemay (2005), originalmente a Java foi considerada uma tecnologia de melhorias a sites Web e ainda é utilizada assim, pois várias páginas na Web contém programas Java que são executados automaticamente quando as páginas são carregadas no navegador.

De acordo com Pressman (2002), as aplicações *web* configuram-se como evolução dos softwares convencionais, ou seja, mantiveram os mesmos princípios de engenharia no desenvolvimento de aplicações *web* para que continuassem produzindo aplicações de qualidade.

Um fator determinante para ampliação da utilização de Java se deve principalmente pela mobilidade desta em diferentes dispositivos, conforme a própria Oracle - empresa detentora dos direitos Java destaca que o Java foi projetado para possibilitar o desenvolvimento de aplicações portáteis de alto desempenho para a mais ampla variedade possível de plataformas de computação. Pois, ao conceder aplicações entre ambientes diferentes, as empresas passam a ofertar mais serviços e aumentar a produtividade, a comunicação e a colaboração do usuário final — além de reduzir consideravelmente o custo de propriedade das aplicações da empresa e do consumidor.

De acordo com a Oracle (2015), o Java tornou-se inestimável para os desenvolvedores, possibilitando que eles:

> - Gravem software em uma plataforma e o executem virtualmente em qualquer outra plataforma;
> - Criem programas que podem ser executados dentro em um *web* browser e acessem *webservices* disponíveis;
> - Desenvolvam aplicações do servidor para fóruns *on-line*, armazenamentos, pesquisas, processamento de formulários HTML, etc.;
> - Combinem aplicações ou serviços usando a linguagem Java para criar aplicações ou serviços altamente personalizáveis.
> - Crie aplicações potentes e eficientes para telefones celulares, processadores remotos, microcontroladores, módulos sem fio, sensores, *gateways*, produtos de consumo e praticamente qualquer outro dispositivo eletrônico.

3.1.3 PostgreSQL

Um dos itens primordiais para o funcionamento de um sistema é a escolha de um Sistema Gerenciador de Banco de Dados (SGBD), afinal é este quem garante que os dados sejam armazenados de maneira correta.

Segundo Lobo (2008), os SGBDs foram criados para aprimorar o gerenciamento dos dados, proporcionando melhorias significativas em relação aos arquivos de armazenamento. Os SGBDs são verdadeiros sistemas que têm a função de armazenar os dados e, também, oferecer mecanismos para o gerenciamento das informações.

Alguns itens devem ser levados em consideração no momento da escolha de um SGBD, tal como a segurança, uma vez que ela impede a violação de consistência dos dados, acesso indevido (usuários e aplicações), além de prover controle de concorrência (conflito de acesso a dados simultâneos).

Para realização deste projeto o SGBD escolhido foi o PostgreSQL, por ser um banco de dados de nível corporativo com a vantagem de ser gratuito.

De acordo com Colares (2007), o PostgreSQL teve seu início na:

> [...] Universidade de Berkeley, na Califórnia, em 1986. À época, um programador chamado Michael Stonebraker liderou um projeto para a criação de um servidor de banco de dados relacionais chamado Postgres, oriundo de um outro projeto da mesma instituição denominado Ingres. Essa tecnologia foi então comprada pela Illustra, empresa posteriormente adquirida pela Informix. Porém, mesmo diante disso, dois estudantes de Berkeley (Jolly Chen e Andrew Yu) compatibilizaram o Postgres à linguagem SQL. Este projeto recebeu o nome de Postgres95.(COLARES,2007,p.36)

Em 1996, quando o projeto estava estável, o banco de dados recebeu o nome de PostgreSQL. No entanto, enquanto ainda possuía o nome Postgres95, o banco de dados teve várias mudanças. O seu código foi totalmente revisado e a linguagem SQL foi definida como padrão (COLARES, 2007).

O PostgreSQL é um SGBD - objeto relacional de código aberto, com mais de 15 anos de desenvolvimento. De acordo com Cruvinel (2007), ele é extremamente robusto e confiável, com uma gama generosa de recursos que também o torna flexível e adaptável. É considerado objeto-relacional por implementar, além das características de SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados.

Entre as principais características apresentadas pelo PostgreSQL, estão a: compatibilidade multi-plataforma, ou seja, pode ser utilizado e executado em vários sistemas operacionais, como Windows, Mac OS X, Linux e outras variantes de Unix; compatibilidade com diversas linguagens de programação, entre elas, Java, PHP, e

C/C++; possui a base de dados com tamanho ilimitado; Tabelas com tamanho de até 32 TB; Quantidade de linhas de até 1.6 TB ilimitada; Campos de até 1 GB (COLARES, 2007).

Uma desvantagem apresentada pelo PostgreSql era o fato de não ter uma versão nativa para a plataforma Windows, precisando assim de uma camada de emulação, lançada apenas a partir da versão 8.0. Outra é não ter suporte a envio de e-mails integrado ao banco.

3.1.4 Android

O Android é o sistema operacional do Google para dispositivos móveis, e que atualmente é o líder mundial nesse segmento. De acordo com Lecheta (2015), o sucesso do Android está por trás do seu desenvolvimento, que conta com gigantes do mercado de mobilidade.

O grupo Open HandsetAlliance (OHA) conta com empresas como a Intel, Samsung,LG, Motorola, ASUS, Dell, entre outros nomes de peso que auxiliam no desenvolvimento da plataforma. O Google representa uma grande parte da força Android, porém existe muito interesse no desenvolvimento de uma plataforma móvel poderosa e flexível, de código aberto e que atenda a necessidade de todos, ou seja, as grandes potências do mercado móvel contribuem significativamente e ativamente para os bons resultados apresentados pelo Android.

O mercado de dispositivos móveis vem crescendo admiravelmente. De acordo com matéria do jornal O Globo (2012), a penetração de celulares no mundo é de 87%. E conforme pesquisa, como o serviço de telefonia móvel vem barateando gradativamente, existem atualmente cerca de 8 bilhões de linhas móveis ativas no mundo, ou seja, praticamente existe um aparelho móvel para cada pessoa no mundo, incluindo crianças.

Conforme aponta Deitel (2015), as vendas de aparelhos Android estão crescendo vertiginosamente, originando enormes oportunidades para os desenvolvedores de aplicativos Android.

> - A primeira geração de telefones Android foi lançada em outubro de 2008. Em outubro de 2013, um relatório da StrategyAnalytics mostrou que o Android tinha 81,3% da fatia de mercado global de

> smartphones, comparados com 13,4% da Apple, 4,1% da Microsoft e 1% do Blackberry.
> - •De acordo com um relatório do IDC, no final do primeiro trimestre de 2013, o Android tinha 56,5% de participação no mercado global de *tablets*, comparados com 39,6% do iPad da Apple e 3,7% dos *tablets* Microsoft Windows.
> - Em abril de 2013, mais de 1,5 milhão de aparelhos Android (incluindo smartphones, *tablets*, etc.) estava sendo ativado diariamente.
> - Quando esta obra estava sendo produzida, havia mais de um bilhão de aparelhos Android ativado.
> - Atualmente, os dispositivos Android incluem smartphones, *tablets*, *e-readers*, robôs, motores a jato, satélites da NASA, consoles de jogos, geladeiras, televisões, câmeras, equipamentos voltados à saúde, relógios inteligentes (*smartwatches*), sistemas automotivos de "*infotainment*" de bordo (para controlar rádio, GPS, ligações telefônicas, termostato, etc.) e muitos outros. (DEITEL, 2015, p.3)

Os avanços alcançados pela OHA e a plataforma Android, acabam beneficiando tanto os fabricantes de celulares, os usuários comuns quanto às empresas de desenvolvimento e desenvolvedores.

Atualmente os usuários comuns têm acesso a aparelhos que anteriormente tinham custo elevado por apresentarem diferenciais, hoje os consumidores estão buscando cada vez mais celulares com recursos como câmeras de alta definição, músicas, *bluetooth*, capacidade para jogos, GPS, acesso à internet e principalmente a redes sociais e TV digital.

Na área corporativa a procura por dispositivos móveis, como tabletes ou smartphones, vem crescendo também. Diversas empresas procuram incorporar suas aplicações convencionais para dispositivos móveis, tornando seus negócios mais ágeis e flexíveis e alimentando cada vez mais o conceito "mobilidade" (FARTO, 2010).

Conforme Lecheta (2015) o celular cada vez mais ocupa um espaço importante na vida das pessoas e o Android procura justamente agradar o usuário, fazendo com que o usuário encontre tudo que procura em um único aparelho.

Para os fabricantes o fato de ter uma plataforma única e consolidada é uma vantagem, pois podem customizar os seus produtos de maneira independente sem ter que compartilhar essas alterações. Já para desenvolvedores eles podem usufruir de uma plataforma de desenvolvimento moderna e com vários recursos.

O mundo da tecnologia está sempre em evolução e a OHA tem como objetivo principal manter uma plataforma-padrão na qual todas as novas tendências do mercado estejam englobadas em uma única solução.

3.1.5 SpringFramework

Spring é um *framework* de código aberto que foi criado por Rod Johnson para lidar com a complexidade de desenvolvimento de aplicativos corporativos. É um dos *frameworks* líderes do mercado *full-stack* Java/JEE. De acordo Pacheco (2007), Spring fornece diversos benefícios para diferentes projetos, aumentando a produtividade de desenvolvimento e a performance em tempo de *runtime* em quanto ao mesmo tempo prove uma cobertura para testes e muita qualidade para a aplicação. Foi criado com o intuito simplificar a programação em Java, possibilitando construir aplicações que antes só era possível utilizando EJBs.

De acordo com Gentil (2006), o Spring atualmente possui diversos módulos como Spring Data (trata da persistência), Spring Security (trata da segurança da aplicação) entre outros módulos.

Mas o principal (*core*) pode ser utilizado em qualquer aplicação Java, as principais funcionalidades são a Injeção de Dependência, do inglês Context Dependency Injection (CDI), e a Programação Orientada a Aspectos, do inglês *Aspect-Oriented Programming* (AOP). Conforme Gentil (2006), cabe ao desenvolvedor dizer ao Spring o que quer utilizar, tornando-a uma poderosa ferramenta, por não existir a necessidade de se arrastar todas as ferramentas do *framework* para criar uma aplicação simples.

3.2 Método

Esse trabalho foi realizado com base no modelo de ciclo de vida Clássico ou Cascata que segundo Audy e Prikladnicki (2008) é um ciclo de vida que utiliza um

método sistemático e sequencial, em que o resultado de uma fase constitui a entrada de outra, a qual neste trabalho respeita as seguintes etapas:

Levantamento de requisitos: o processo de captura de requisitos pode ser suportado por diversas técnicas, como entrevistas, aplicação de questionários ou observação direta.

Neste trabalho, o levantamento de dados para análise de requisitos foi realizado por meio de observação em diversos estabelecimentos do ramo gastronômico e por conhecimento do desenvolvedor por ter relação com pessoas que possuem pequenos restaurantes, buscando identificar as principais necessidades do segmento.

Análise e Projeto: na elaboração dos diagramas da UML para as fases de análise e projeto, foi utilizada como base a obra do autor Wazlawick (2011). Primeiramente foi desenvolvido o diagrama de casos de uso, pois é necessário definir os processos que o sistema irá controlar antes de detalhar o sistema através de outros diagramas como o modelo conceitual que já inclui atributos, propriedades e relações entre as classes. Em seguida foi realizado o caso de uso expandido do método cadastrar pedido. E posteriormente foram elaborados os demais diagramas, modelo conceitual, sequência, comunicação e entidade-relacionamento.

Implementação e Testes: para a implementação do sistema foram utilizadas as ferramentas Java epgAdmin. A codificação seguiu o conceito de programação orientada a objetos dando ênfase em características como: polimorfismo, herança, encapsulamento e desenvolvimento de classes. E por fim, a fase de testes foi realizada apenas pelo desenvolvedor, ou seja, o sistema não foi implantado em nenhum estabelecimento do ramo de pequenos restaurantes.

4 RESULTADO

Neste capítulo apresenta-se o sistema desenvolvido, exibindo telas, trechos de códigos, modelagem, análise de requisitos, estrutura do banco de dados e todos os detalhes referentes à estruturação do sistema desenvolvido.

Primeiramente o sistema é descrito em uma visão do lado do cliente. A seguir são apresentados os resultados obtidos com a coleta e análise dos requisitos, e a modelagem gerada sobre o sistema, incluindo os diagramas da UML: diagrama de casos de uso, conceitual, sequência e comunicação. É apresentada ainda a modelagem do banco de dados, por meio de um diagrama de entidade e relacionamento.

Por fim é exibido o sistema, sua interface visual e o código gerado, destacando os principais trechos de código desenvolvidos utilizando conceitos de herança, polimorfismo e demais características da orientação a objetos.

4.1 Modelagem do Sistema

Para desenvolvimento de um software*,* o primeiro passo, e um dos mais importantes, é a realização da análise, que engloba entre outras coisas, o processo de levantamento de requisitos. Este processo permitirá definir quais são os problemas ou necessidades do cliente, e quais são as possibilidades de resolvê-los por meio do software que será desenvolvido. Tendo os requisitos definidos é possível então modelar o sistema.

O sistema desenvolvido foi modelado utilizando o paradigma de orientação a objetos, baseando-se na UML, criando diagramas de sequência e comunicação, de casos de uso e de classes.

Esta seção irá abordar os requisitos levantados e classificados como funcionais e não-funcionais, e posterior modelagem gerada para o sistema e para o banco de dados com base nestes requisitos.

4.2 Requisitos de usuário para o Sistema

O software resultado deste trabalho visa manipular, de maneira simples e organizada, os processos de restaurantes de pequeno porte, como cadastros de produtos, pedidos, clientes e categorias a fim de auxiliar de forma simples os gestores da organização.

O usuário deverá cadastrar alguns dados essenciais ao uso do sistema, para tanto ele deverá fazer um levantamento de seu estoque de produtos com seus respectivos valores. De posse destes dados, inicialmente será necessário realizar o cadastro dos produtos disponíveis para utilização/venda, por meio de uma tela de produtos. Posteriormente, com os produtos cadastrados, já será possível realizar as vendas, utilizando uma tela de pedidos.

4.2.1 Requisitos Funcionais e Não-Funcionais

A coleta de requisitos foi definida através de observações em diferentes estabelecimentos e pelo fato do aluno ter ligação com pessoas que atuam nesse ramo.

Desta forma foi possível detectar as dificuldades presentes na gerência dos estabelecimentos que não utilizam um sistema informatizado. Dentre os quais controle de entradas, pedidos, clientes. Com isso foi possível desenvolver o levantamento de requisitos apresentado na Tabela 1 e na Tabela2.

Os requisitos funcionais levantados são apresentados utilizando a abreviatura RFX para identificar cada requisito, sendo X um identificador numérico. E os requisitos não-funcionais, caracterizados como restrições ou complemento às funcionalidades que o *software* deve possuir, são apresentados utilizando-se a abreviatura RNFX para identificar cada requisito, sendo X o identificador numérico do requisito não funcional.

Nos quadros gerados, todos os requisitos serão implementados e terão prioridade, pois caso algum requisito não seja implementado o sistema não funcionará de maneira a atender as regras de negócio.

O quadro 1 apresenta os requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF1	*Cadastrar Produtos*	O usuário do sistema deve cadastrar os produtos informando a sua descrição e valor.
RF2	*Cadastrar Pedido*	O sistema deverá registrar os pedidos efetuados pelos clientes, bem como os produtos consumidos, quantidade e o valor total.

Quadro 1 - Requisitos Funcionais do Sistema

O Quadro 2 apresenta os requisitos não-funcionais identificados para o sistema. Os requisitos não-funcionais explicitam regras de negócio, restrições ao sistema de acesso, por exemplo, requisitos de qualidade, desempenho, segurança e outros. Os requisitos não-funcionais do sistema estão relacionados aos requisitos funcionais pelos nomes.

Identificação	Nome	Descrição
RNF1	*Sincronização com o servidor*	A plataforma móvel do sistema deve sincronizar com o servidor para que seja possível a realização dos pedidos.
RNF2	*Calcular valor total*	O sistema deverá calcular o valor total de cada pedido.

Quadro 2 - Requisitos Não Funcionais do Sistema

4.2.2 Diagrama de Caso de Uso

Na Figura 1 apresenta-se o diagrama de casos de uso que aborda as funcionalidades do sistema, e a interação do usuário com estas funcionalidades. O ator apresentado no diagrama é o usuário operador do estabelecimento que é responsável pela realização dos cadastros do sistema.

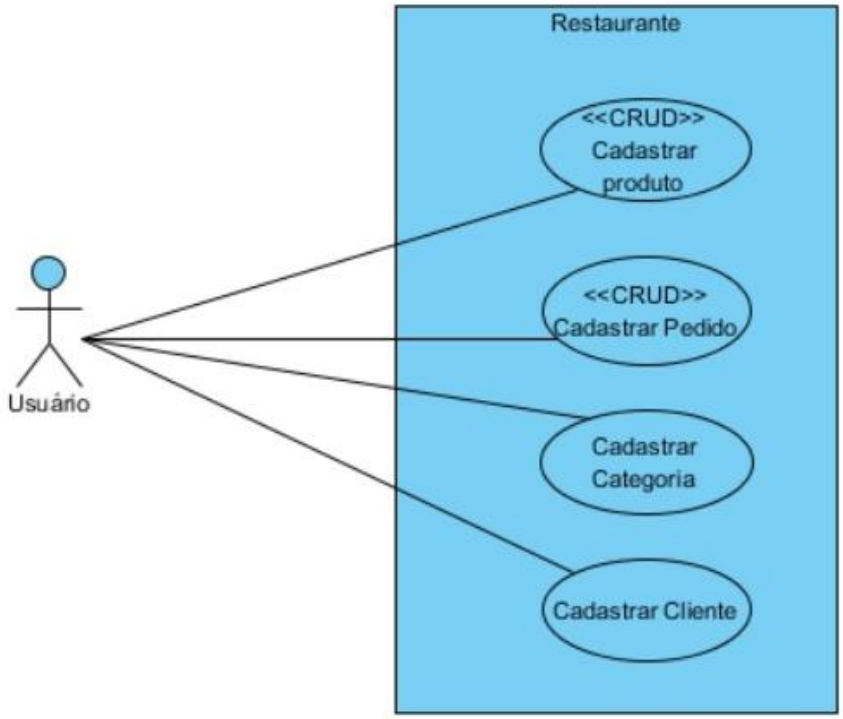

Figura 1 - Documentação: Diagrama de Casos de Uso

O Quadro 3 apresenta o caso de uso expandido do processo de cadastrar pedido.

Caso de Uso: Cadastrar Pedido
Atores: Usuário
Precondições: Os dados dos produtos devem estar previamente cadastrados
Pós-condições: Pedido cadastrado com produtos inclusos
Requisitos Correlacionados: Cadastrar produtos (RF1)
Variações tecnológicas:
Fluxo Principal: 1. [IN] Este caso de uso inicia quando o usuário vai atender um cliente e lançar seu pedido no sistema. 2. [IN] O usuário informa ao sistema o código do cliente. 3. [OUT] O sistema habilita ao usuário a tela para o lançamento dos produtos do pedido. 4. [IN] O usuário informa ao sistema quais os produtos serão lançados no pedido com suas respectivas quantidades. 5. [OUT] O sistema efetiva os itens do pedido.
Tratamento de Exceções: 3.1a: O código do pedido informado já está cadastrado no sistema. Variante 3.1a: Volta para o início 3.1.1a: Mostra uma mensagem com o erro para o usuário. 3.1.2a: Volta para o passo 2. Variante 3.2a: Cancela a operação 3.2.1a: Mostra uma mensagem ao usuário. 3.2.2a: Cancela o pedido.

Quadro 3 - Caso de Uso Expandido (Cadastrar Pedido)

4.2.3Modelo Conceitual

Na Figura 2 apresenta-se o diagrama conceitual. Este diagrama foi criado como um espelho do banco de dados, ou seja, todas as classes são classes de entidade e, portanto, devem ser persistentes. No sentido de que todas as tabelas possuem uma classe, a qual contém todos os campos da respectiva tabela. Todas as operações de manipulação e acesso ao banco de dados serão realizadas dentro da própria classe e as telas do sistema apenas criam a classe e executam os seus métodos.

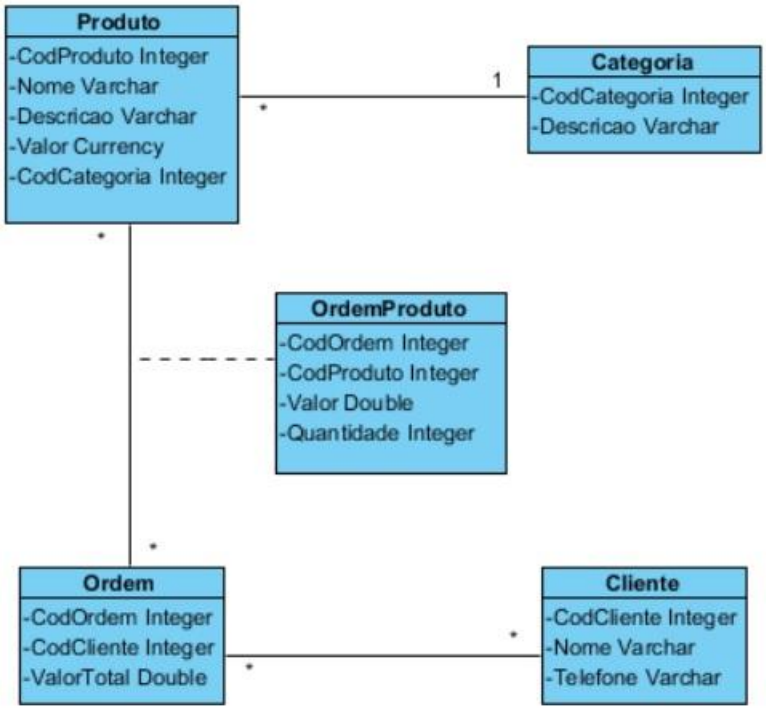

Figura 2 - Documentação: Diagrama Conceitual

Os Quadros 4 a 8 apresentam a descrição de cada uma das classes do diagrama de classes representado na Figura 2.

Identificação:	Produto
Descrição:	É previsto que nesta classe sejam requisitados os dados dos produtos para que assim o cadastro seja efetuado com sucesso.
Requisitos:	RF1
Atributos:	CodProduto (Integer): código de cadastramento do produto. Nome (Varchar): nome do produto Descricao (Varchar): descrição do produto que será

	comercializado. Preco (Currency): preço do produto.
Métodos:	Cadastrar, Alterar, Consultar e Excluir.

Quadro 4 - Descrição da Classe Produto

Identificação:	Ordem
Descrição:	Esta classe é responsável por registrar os pedidos realizados pelos clientes.
Requisitos:	RF4
Atributos:	CodOrdem (Integer): código de cadastramento da ordem. CodCliente (Integer): código do cliente ValorTotal (Double): valor da soma total da ordem Data (Datetime): data do pedido.
Métodos:	Cadastrar, Alterar, Excluir, Consultar.

Quadro 5 - Descrição da Classe Ordem

Identificação:	OrdemProduto
Descrição:	Classe responsável por armazenar os produtos referentes a cada pedido.
Requisitos:	RF1 e FR4
Atributos:	CodOrdem (Integer): código da ordem CodProduto (Integer): código do produto QtdeProduto (Integer): quantidade do produto. Preco (Currency): preço do produto do pedido.
Métodos:	Cadastrar, Alterar, Consultar, Excluir.
Observações:	O cadastro de itens de pedido é feito junto com o cadastro de pedidos, uma vez que apenas a partir do cadastro de um pedido será realizado o cadastro de itens de pedido.

Quadro 6 - Descrição da Classe OrdemProduto

Identificação:	Categoria
Descrição:	Classe responsável por armazenar as categorias dos produtos.
Requisitos:	RF1
Atributos:	CodCategoria (Integer): código da categoria Descrição (Varchar): descrição da categoria
Métodos:	Cadastrar, Alterar, Consultar, Excluir.

Quadro 7 - Descrição da Classe Categoria

Identificação:	Cliente
Descrição:	Classe responsável por armazenar os clientes.
Requisitos:	RF1
Atributos:	CodCliente (Integer): código do cliente

	Nome (Varchar): nome do cliente Telefone (Varchar) telefone de contato do cliente
Métodos:	Cadastrar, Alterar, Consultar, Excluir.

Quadro 8 - Descrição da Classe Cliente

4.2.4 Diagrama de Sequência

Na Figura 3 apresenta-se o diagrama de sequência, que esboça a sequência de acontecimentos no decorrer do processamento do sistema.

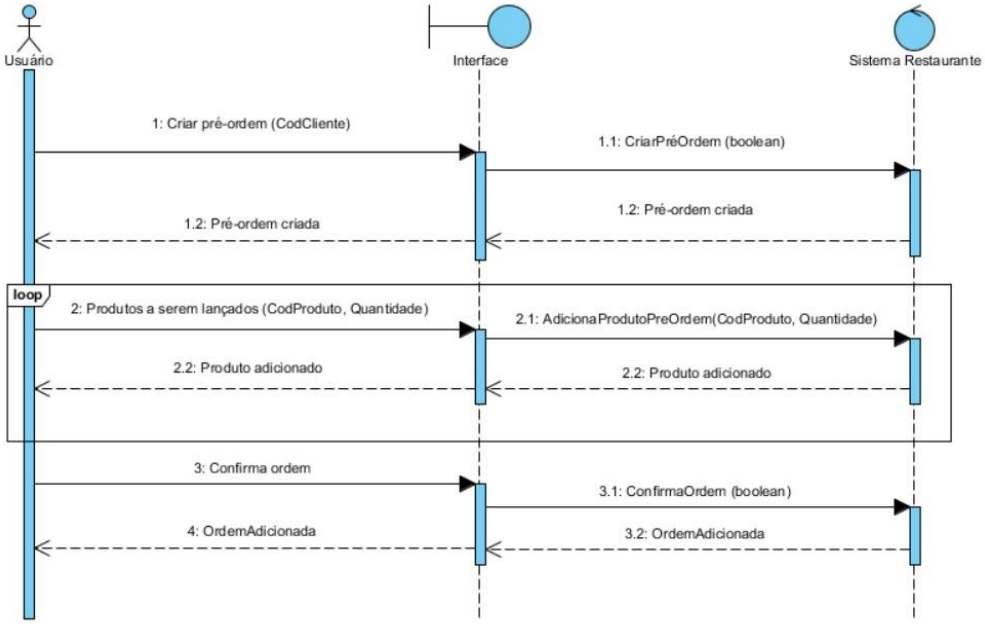

Figura 3 - Diagrama de Sequência (Cadastrar Pedido)

Primeiramente o usuário solicitará a criação de um pré-pedido. Após isso, o usuário entrará em um *looping* para adicionar os produtos do pedido. Ao concluir a inclusão de todos os produtos, o usuário navegará para uma nova tela na qual serão apresentados todos os itens do pedido e o mesmo poderá ser confirmado.

4.2.5 Diagrama de Comunicação

Na Figura 4 está o diagrama de comunicação, que mostra de maneira semelhante ao diagrama de sequência, a comunicação dinâmica entre os objetos, ou seja, o fluxo das mensagens entre um objeto e outro para um dado processo. O

diagrama de comunicação serve de complemento para o diagrama de sequência, ao passo que decompõe a classe controladora nas classes que estarão envolvidas no processo, sendo possível; identificar os métodos necessários a cada classe.

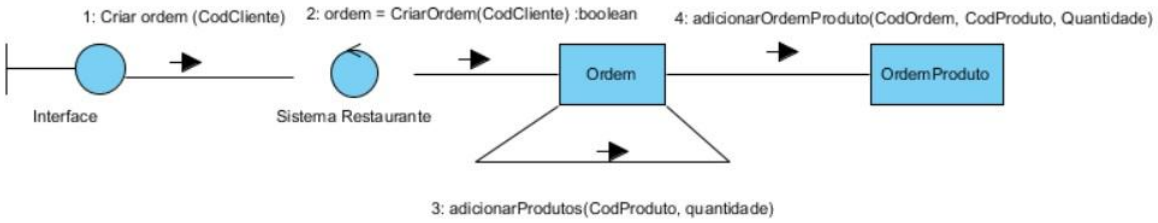

Figura 4 - Diagrama de Comunicação - Cadastrar Pedido

4.2.6 Diagrama de Entidade e Relacionamento

O Diagrama de Entidade e Relacionamento (DER) modela as tabelas de um banco de dados, definindo campos, tipo de dados de cada campo, chaves primárias e relacionamentos entre as tabelas determinando as chaves secundárias ou estrangeiras. Destaca-se que o DER apresentado na Figura 5 apresenta o mapeamento objeto-relacional a partir do modelo conceitual da Figura 2. Na Figura 5 observam-se todas as tabelas que armazenam os dados processados no sistema e seus respectivos campos e interligações.

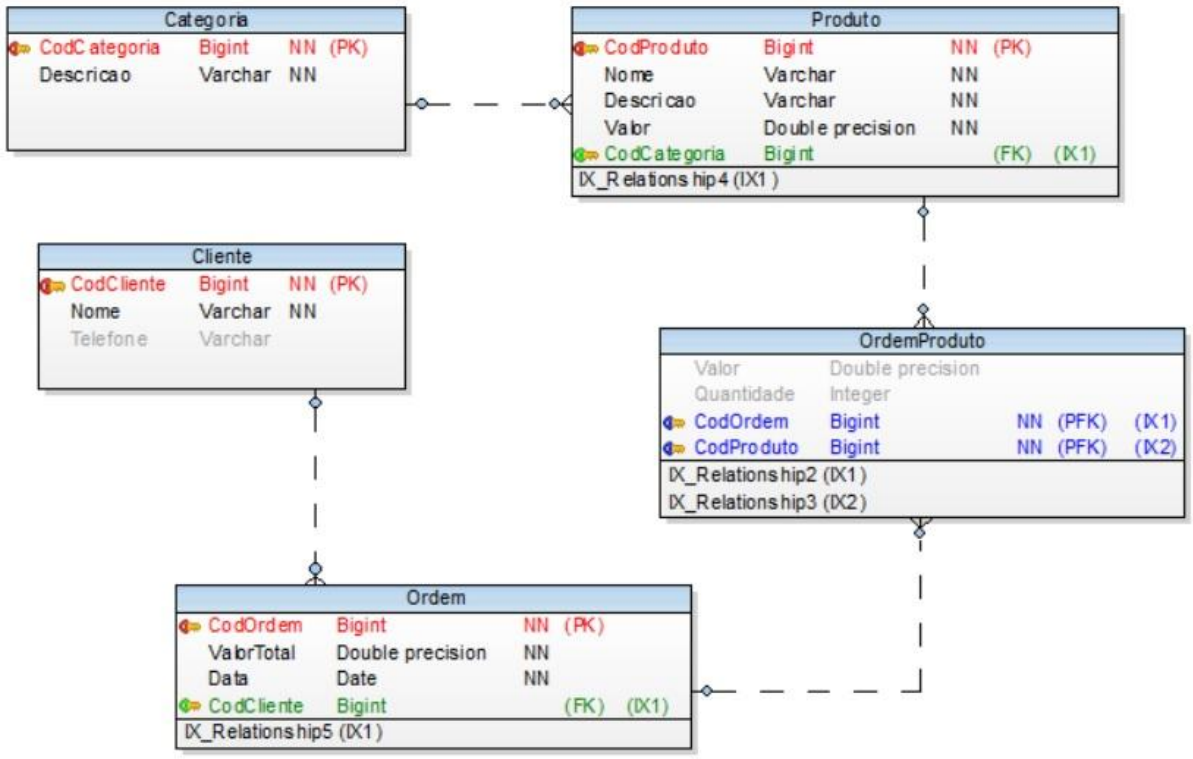

Figura 5 - Diagrama de Entidade e Relacionamento

A seguir são apresentados os quadros de descrições das entidades que compõem o banco de dados, conforme expõem a Figura 5.

O Quadro 9 apresenta a descrição da tabela Produto.

PRODUTO						
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Padrão**	**Observações**
CodProduto	Integer	Não	Sim	Não		Auto-denominado
Nome	Varchar	Não	Não	Não		
Descricao	Varchar	Não	Não	Não		
Preco	Currency	Não	Não	Não		
Login	String	Não	Não	Sim		

Quadro 9 - Descrição da Tabela Produto (DER)

O Quadro 10 apresenta a descrição da tabela Ordem.

ORDEM						
Campo	**Tipo**	**Nulo**	**Chave**	**Chave**	**Padrão**	**Observações**

			primária	**estrangeira**		
CodOrdem	Integer	Não	Sim	Sim		Auto-denominado
CodCliente	Integer	Não	Não	SIm		
DataPedido	Datetime	Não	Não	Não		
ValorTotal	Double	Não	Não	Não		

Quadro 10 - Descrição da Tabela Pedido (DER)

O Quadro 11 apresenta a descrição da tabela OrdemProduto.

ORDEMPRODUTO						
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Padrão**	**Observações**
CodOrdem	Integer	Não	Sim	Sim		
CodProduto	Integer	Não	Sim	Sim		
Quantidade	Integer	Não	Não	Não		
Valor	Currency	Não	Não	Não		

Quadro 11 - Descrição da Tabela OrdemProduto (DER)

O Quadro 12 apresenta a descrição da tabela Cliente.

CLIENTE						
Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Padrão**	**Observações**
CodCliente	Integer	Não	Sim	Não		Auto-denominado
Nome	Varchar	Não	Não	Não		
Telefone	Varchar	Não	Não	Não		

Quadro 12 - Descrição da Tabela Cliente (DER)

4.3 Apresentação do Sistema

Na Figura 6 está a janela principal do sistema.

Figura 6 - Apresentação Inicial do Sistema

Como pode ser visto na Figura 6 existem cinco botões que são:

- Novo pedido, quando o usuário deseja cadastrar um novo pedido;
- Consulta pedidos, ao clique deste botão é possível verificar por meio de uma lista os pedidos anteriormente cadastrados;
- Clientes, nessa tela poderá ser consultado o cadastro dos clientes;
- Produto, nesta tela poderão ser cadastrados os produtos;
- Categoria.

Na Figura 7 pode-se visualizar a tela de Pedidos, a qual o usuário utilizará informando dados, como o número do pedido, a data e os produtos que compõem o pedido.

Figura 7 - Formulário de Cadastro de Pedidos

Para o cadastro dos produtos o sistema altera o conteúdo central, exibindo a tela mostrada na Figura 8.

Figura 8 - Formulário de Cadastro de Produtos

Nesse formulário o usuário deverá fazer a descrição (nome do produto) e o valor a ser comercializado pelo restaurante, bem como a categoria em que ele estará vinculado. Ao realizar a inserção, o produto será automaticamente incluído na lista de produtos.

4.4 Implementação do Sistema

Todas as classes desse sistema foram desenvolvidas baseadas nos conceitos de orientação a objetos, visto ao longo da graduação e especialização. Dentro desta seção serão apresentados alguns trechos considerados mais importantes ao funcionamento geral do sistema.

As listagens a seguir exemplificam como foi implementado o sistema, relacionando primeiramente a codificação desenvolvida para a plataforma móvel e depois a estrutura do lado do servidor. A Listagem 1 apresenta como são recuperadas as informações do servidor para o preenchimento dos campos *Spinner*e *ListView* na tela que será feito o pré-pedido. Ao selecionar uma categoria no Spinner das categorias, o ListView dos produtos é atualizado automaticamente e trará somente os produtos relacionados nesta determinada categoria. Quando um determinado produto é selecionado no ListView, uma variável temporária recebe uma identificação do produto selecionado, a quantidade que foi solicitada e preenche um TexTView com essas informações. No momento que o usuário clica para adicionar o produto no pedido, um ArrayList receberá os dados da variável temporária e esta lista será repassada para a tela seguinte na qual serão apresentados os dados do pedido.

```
…
catDao = new CategoryDAO(this);
prodDao= new ProductDAO(this);
res = getResources();

carregar("categoria");
catDao.includeArray(jsonArray);
registrosCat = catDao.list();

SimpleCursorAdaptercatAdapter = new SimpleCursorAdapter( this, android.R.layout.simple_list_item_1, registrosCat,
        new String[] { "name" }, new int[] { android.R.id.text1 }, 0 );
```

```
spCategory.setAdapter(catAdapter);
carregar("produto");
prodDao.includeArray(jsonArray);

spCategory.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
              final Cursor registrosProd = prodDao.listByCategory(id);
              SimpleCursorAdapterprodAdapter = new SimpleCursorAdapter(PreOrderActivity.this,
              android.R.layout.simple_list_item_1, registrosProd, new String[]{"name"}, new int[]{android.R.id.text1}, 0);
              lvProducts.setAdapter(prodAdapter);
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
           adapter = new CustomAdapter(listProductView, listProductViewValueArray, res);
lvProducts.setAdapter(adapter);
        }
      });

lvProducts.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

           Cursor registroById = prodDao.getById(id);
           if (registroById.moveToNext()){
              prodTemp = new Product();
              prodTemp.set_id(registroById.getLong(registroById.getColumnIndex("_id")));
              prodTemp.setName(registroById.getString(registroById.getColumnIndex("name")));
              prodTemp.setDescription(registroById.getString(registroById.getColumnIndex("description")));
              prodTemp.setValue(registroById.getDouble(registroById.getColumnIndex("value")));
              prodTemp.setCategory_id(registroById.getInt(registroById.getColumnIndex("category_id")));
              tvSelectedProduct.setText(prodTemp.getName());
              tvSelectedProductQuantity.setText(" Unitario: " + prodTemp.getValue());
           }
        }
      });
     btAddOnPreOrder.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
             addOnPreOrder();
        }
      });
       btFinalize.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                   finalizeOnClick();
              }
      });
```

```
    }
    private void finalizeOnClick() {
        Intent i = new Intent( this, OrderActivity.class );
        i.putExtra("preOrderList", preOrderList);
        i.putExtra("listProductOnPreOrderArray", listProductOnPreOrderArray);
        startActivity( i );
    }
    private void addOnPreOrder() {
        if (tvSelectedProduct.getText().length()>0 ){
            item = new OrderProduct();
            item.setProduct_id(prodTemp.get_id());
            item.setQuantity(Integer.parseInt(etQuantity.getText().toString()));
            preOrderList.add(item);
           Toast.makeText(this, "Adicionadoaopedido", Toast.LENGTH_SHORT).show();
           etQuantity.setText("");
           lvProducts.clearChoices();
           item = null;
           listProductOnPreOrderArray.add(prodTemp);
        }
    }
...
public void carregar(final String typeData) {
            AsyncTask<Void, Void, JSONArray> at = new AsyncTask<Void, Void, JSONArray>() {
                @Override
                protected JSONArraydoInBackground(Void... params) {
                    try {
                        ConexaoUtilsct = new ConexaoUtils();
                         HttpURLConnectionconexao = ct.conectar("http://192.168.1.5:8085/" + typeData + "/");
                        JSONArray jsonArray = new JSONArray();
                        if (conexao.getResponseCode() == HttpURLConnection.HTTP_OK) {
                                InputStream input = conexao.getInputStream();
                                jsonArray = new JSONArray(ct.byteToString(input));
                        }
                        return jsonArray;
                    } catch (Exception e) {
                          e.printStackTrace();
                    }
                    return null;
                }
            }.execute();
        try {
                jsonArray = at.get();
        } catch (Exception e) {
                e.printStackTrace();
     }
}
```

Listagem 1 - Apresentação da Classe PreOrderActivity

Para a montagem dos *Adapters* que preenchem os campos, são necessários que os dados estejam no formato de *ArrayList*, porém o retorno dos dados do servidor vem em formato *JSONArray.* Na Listagem 2 está o código para a inserção desses dados em uma tabela temporária na plataforma móvel para que possam ser recuperados de forma a montar o *ArrayList* e assim preencher adequadamente os *adapters*.

```
…
public void includeArray (JSONArray json){
limparTabela();
ContentValues register = new ContentValues();
JSONObjectjsonCategory;
      for (inti = 0; i<json.length(); i++) {
         try {
              jsonCategory = json.getJSONObject(i);
              register.put("_id", jsonCategory.getLong("id"));
              register.put("name", jsonCategory.getString("descricao"));
              db.insert( "category", null, register );
         } catch (Exception ex){
               ex.printStackTrace();
         }
      }
   }

publicvoidlimparTabela() {
       db.execSQL( "DELETE FROM category" );
}
```

Listagem 2 - Inclusão do JSONArray em tabela temporária na plataforma móvel

A Listagem 3 apresenta a codificação da tarefa assíncrona que vai enviar os dados que serão inseridos no servidor. Ela é assíncrona, pois não deve interromper o fluxo do software enquanto o servidor estará recebendo os dados. Nesta tarefa, é montada a mensagem de retorno no formato JSON, a qual será enviada e interpretada pelo servidor para posterior inserção no banco de dados.

```
…
public void sendProduct(final Product product){
AsyncTask<Void, Void, String> at =
            new AsyncTask<Void, Void, String>() {
               @Override
               protected String doInBackground(Void... params) {
                  try {
                     HttpURLConnectionhttpcon;
                     String url = "http://192.168.1.5:8085/produto/";
```

```
                String data = "{\"nome\":\"" + product.getName() + "\",\"descricao\":\"" + product.getDescription() + "\",\"valor\":" + product.getValue() + ",\"categoria\":{\"id\":" + product.getCategory_id() + " }}";

                String result = null;
                try{
                        //Connect
                        httpcon = (HttpURLConnection) ((new URL(url).openConnection()));
                        httpcon.setDoOutput(true);
                        httpcon.setRequestProperty("Content-Type", "application/json");
                        httpcon.setRequestProperty("Accept", "application/json");
                        httpcon.setRequestMethod("POST");
                        httpcon.connect();

                        //Write
                       OutputStreamos = httpcon.getOutputStream();
                       BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, "UTF-8"));
                       writer.write(data);
                       writer.close();
                       os.close();

                    //Read
                     BufferedReaderbr = new BufferedReader(new InputStreamReader(httpcon.getInputStream(),"UTF-8"));
                     String line = null;
                     StringBuildersb = new StringBuilder();

                    while ((line = br.readLine()) != null) {
                            sb.append(line);
                    }

                   br.close();
                    result = sb.toString();

                } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                return result;
            } catch (Exception e) {
                 e.printStackTrace();
            }
            return null;
          }
        }.execute();
}
```

Listagem 3 - Tarefa assíncrona de envio de dados para o servidor

Por fim, mas não menos importante do lado do cliente, na Listagem 4 está o código para carregar os dados que vem do servidor no formato JSON.

```
public void carregar(final String typeData){
        AsyncTask<Void, Void, String> at = new AsyncTask<Void, Void, String>() {
            @Override
            protected String doInBackground(Void... params) {
                try {
                    ConexaoUtilsct = new ConexaoUtils();
                    HttpURLConnectionconexao = ct.conectar("http://192.168.1.5:8085/" + typeData + "/");
                    JSONArrayjsonArray = new JSONArray();
                    if (conexao.getResponseCode() == HttpURLConnection.HTTP_OK) {
                        InputStream input = conexao.getInputStream();
                        jsonArray = new JSONArray(ct.byteToString(input));
                    }
                    return jsonArray.toString();
                } catch (Exception e) {
                    e.printStackTrace();
                }
                return null;
            }
        }.execute();
}
```

Listagem 4 - Tarefa assíncrona para recebimento de dados do servidor

Na Listagem 5 é apresentada a utilização do Spring Framework para a persistência de dados já na implementação de uma classe modelo dentro do servidor do sistema. As anotações nos atributos têm por finalidade implementar diretamente no sistema de banco de dados as entidades que serão utilizadas no sistema.

```
import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import lombok.Data;

@Entity
public class Categoria implements Serializable{
        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue
        private Long id;
        @Column(length = 100, nullable=false)
        private String descricao;

        public Long getId() {
                return id;
        }
        public void setId(Long id) {
                this.id = id;
        }
        public String getDescricao() {
```

```
		return descricao;
	}
	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}
	public static long getSerialversionuid() {
		return serialVersionUID;
	}
}
```

Listagem 5 - Classe de entidade no servidor

A Listagem 6 apresenta a codificação do *Controller* de uma entidade. Ela é responsável por gerenciar as solicitações feitas pela plataforma móvel e retorná-las, de modo que os dados sejam apresentados e possam ser utilizados.

```
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import br.edu.utfpr.springaula5.model.Categoria;
import br.edu.utfpr.springaula5.repository.CategoriaRepository;

@RestController
@RequestMapping("/categoria")
public class CategoriaController2 {

	@Autowired
	private CategoriaRepositorycategoriaRepository;

	@RequestMapping(value = "/", method = RequestMethod.GET)
	public List<Categoria> all(){
		return categoriaRepository.findAll();
	}

	@RequestMapping(value = "/", method = RequestMethod.POST)
	public String create(@RequestBodyCategoriacategoria){
		categoriaRepository.save(categoria);

		return "[{\"mensagem\":\"Registroinserido com sucesso!\"}]";
	}

	@RequestMapping(value = "/{codigo}", method = RequestMethod.DELETE)
	public String delete(@PathVariable Long codigo){
		categoriaRepository.delete(codigo);

		return "[{\"mensagem\":\"Registroremovido com sucesso!\"}]";
	}

	@RequestMapping(value = "/{codigo}", method = RequestMethod.GET)
	public Categoria get(@PathVariable Long codigo){
		return categoriaRepository.findOne(codigo);
	}
}
```

Listagem 6 - Classe responsável pelo controle das solicitações feitas ao servidor

5 CONCLUSÃO

O trabalho teve como objetivo desenvolver um *software* simples para informatizar os pedidos de clientes em estabelecimentos comerciais, restaurantes, de pequeno porte.

O intuito do sistema é controlar internamente as operações para melhorar o processo da empresa, considerando a entrada e saída de capital, por meio dos cadastros de pedidos e o controle dos produtos que estão sendo comercializados.

Para modelagem e implementação do sistema, foram utilizadas ferramentas de fácil desenvolvimento, que forneceram uma interação fácil do usuário com o sistema. Ferramentas com uma aceitação de mercado muito grande, produzidas por empresas altamente conceituadas a nível mundial. Com todos esses fatores foi possível desenvolver um sistema, que pretende auxiliar na execução dos processos realizados em estabelecimentos que sempre tem um grande fluxo de atendimentos.

De um modo geral, pode-se afirmar que o desenvolvimento deste trabalho permitiu ao acadêmico adquirir uma experiência maior com o processo de análise e desenvolvimento de *software*.

REFERÊNCIAS

AUDY, Jorge; PRIKLADNICKI, Rafael. **Desenvolvimento Distribuído de Software:** desenvolvimento de software com equipes distribuídas. Rio de Janeiro: Ed. Elsevier, 2008.

CADENHEAD, Rogers; LEMAY, Laura. **Aprenda em 21 dias** Java 2. Rio de Janeiro: Campus, 2005.

CADORIN, Marcos Venicio. **Protótico de Gerenciamento de Bares e Restaurantes**. 2010. 74 f. Monografia apresentada na UTFPR – *Campus* Pato Branco para a obtenção do grau de tecnólogo em Análise e Desenvolvimento de Sistemas.

COLARES, Flávio Martins. **Análise comparativa de banco de dados gratuitos.** 2007. 74f. Monografia apresentada na Faculdade Lourenço Filho para obtenção do grau de Bacharel em Ciência da Computação. Fortaleza, 2007. Disponível em: <http://www.flf.edu.br/revista-flf/monografias-computacao/monografia_flaviocolares.pdf>. Acesso em: 18 nov. 2015

CROCE FILHO, Ralfe D; RIBEIRO, Carlos E. **Informática:** Programação de Computadores. São Paulo: Fundação Padre Anchieta, 2010. Disponível em < http://www.colecaotecnica.cpscetec.com.br/other/originais/4_capitulo4_1268330726.pdf >. Acesso em 18 nov. 2015.

CRUVINEL, Fernando Demartine. **Banco de Dados Vozes.** 2007. 36f. Trabalho de conclusão de curso apresentado à Escola de Engenharia de São Carlos, curso de Engenharia da Computação. São Carlos, 2007. Disponível em: <http://www.tcc.sc.usp.br/tce/disponiveis/97/970010/tce-09042010-145131/publico/Cruvinel_Fernando_Demartine.pdf>. Acesso em: 18 nov. 2015.

DEITEL, Paul; DEITEL, Harvey; DEITEL, Abbey. **Android para programadores:** uma abordagem baseada em aplicativos - Porto Alegre: Brookman, 2015.

FARTO, Guilherme de C. **Abordagem orientada a serviços para implementação de um aplicativo Google Android.** Trabalho de conclusão de curso apresentado ao Instituto Municipal de Ensino Superior de Assis. Assis, 2010. Disponível em <http://fema.edu.br/images/arqTccs/0711270028.pdf>. Acesso em: 19 nov. 2015

FERREIRA, Aurélio Buarque de Holanda. **Dicionário da língua portuguesa**. 5. ed. Curitiba: Positivo, 2010. 2222 p. ISBN 978-85-385-4198-1.

GENTIL, Efraim. **Introdução ao Spring Framework.** Disponível em: < http://www.devmedia.com.br/introducao-ao-spring-framework/26212>. Acesso em: 02 dez. 2015.

GONÇALVES, Gilberto; LIMA, Isaura A. de. **Implantação de um Sistema de Informação - Enterprise Resource Planning (ERP):** estudo de caso em uma indústria eletrônica. Revista de Engenharia e Tecnologia. ISSN 2176-7270. Disponível em < http://www.revistaret.com.br/ojs-2.2.3/index.php/ret/article/viewFile/45/62>. Acesso em: 02 dez. 2015.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Pesquisa de Orçamentos Familiares (POF)2008-2009**. Disponível em < http://www.ibge.gov.br/home/estatistica/populacao/condicaodevida/pof/2008_2009/POFcomentario.pdf>. Acesso em: 02 dez. 2015.

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

LOBO, Edson J.R. **Curso Prático de MySQL.** São Paulo: Digerati Books, 2008.

LOPES, Leonardo G. P. **Informatizar para crescer.** Caderno Especial do 26º Congresso Abrasel. Disponível em <http://static1.squarespace.com/static/535fe569e4b0a3d8cb43c11f/t/54527811e4b06ce38748a2c2/1414690833298/Caderno+Congresso.pdf>. Acesso em: 02 dez. 2015.

MARICATO, Percival. **Como Montar e Administrar Bares e Restaurantes**, 6 ed., São Paulo: Ed. Senac São Paulo, 2005.

O'BRIEN, James A.; MARAKAS, George M. **Administração de Sistemas de Informação**. 15 ed., São Paulo: Ed. Bookman, 2010.

O GLOBO. **Mundo tem 6 bilhões de celulares.** 2012. Disponível em <http://oglobo.globo.com/sociedade/tecnologia/mundo-tem-6-bilhoes-de-celulares-6401960>. Acesso em: 20 nov. 2015

OLIVEIRA, Djalma P. R. **Sistemas de Informações Gerenciais: estratégicas, táticas, operacionais**. São Paulo: Ed. Atlas, 2001.

ORACLE. **Informações sobre Tecnologia Java.** Disponível em < https://www.java.com/pt_BR/about/ >. Acesso em 18 nov. 2015

PACHECO, Diego. **Spring Framework (2.0):** Framework para Desenvolvimento de Aplicações em Java. Disponível em <http://pt.scribd.com/doc/18517573/Spring-Framework-2-0-Diego-Pacheco>. Acesso em: 25 nov. 2015.

PRESSMAN, Roger **Engenharia de Software**, Rio de Janeiro, McGraw Hill, 2002.

VENTURINI, James Luiz. Gerenciamento de Bares e Restaurantes. Porto Alegre: Bookman, 2010.

VISUALPARADIGM. **Visual Paradigm**. Disponível em: <http://www.visual-paradigm.com/>. Acesso em: 13 mar. 2013.

WAZLAWICK, Raul Sidnei. **Análise e Projeto de Sistemas de Informação Orientados a Objetos**. 2 ed., Rio de Janeiro: Ed. Elsevier, 2011.