CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

Faculdade de Tecnologia da Baixada Santista "Rubens Lara"

Curso Superior de Tecnologia em Sistema para Internet

DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID UTILIZANDO A TECNOLOGIA MVC. Estudo de caso: acompanhamento de entrega.

Santos

Junho/2015

Ely Assumpção Ndiaye

Lucas Domingues Limeres

Renan Augusto Muniz Alves

DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID UTILIZANDO A TECNOLOGIA MVC. Estudo de caso: acompanhamento de entrega.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia da Baixada Santista “Rubens Lara”, como exigência parcial para obtenção de título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Jorge Luiz Chiara

Santos

Junho/2015

ALVES, Renan Augusto Muniz; LIMERES, Lucas Domingues; NDIAYE, Ely Assumpção;

DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID UTILIZANDO A TECNOLOGIA MVC. Estudo de caso: acompanhamento de entrega. / Ely Assumpção Ndiaye; Lucas Domingues Limeres; Renan Augusto Muniz Alves; Orientador: Prof. Jorge Luiz Chiara. Local: Faculdade de Tecnologia "Rubens Lara", 17 de julho de 2015.

43 p.

Monografia (Trabalho de Conclusão de Curso) – Centro de Educação Tecnológica Paula Souza, Faculdade de Tecnologia "Rubens Lara", Curso Superior de Tecnologia em Sistemas para Internet.

1. Introdução 2. *Android* 3. Engenharia de Software 4. Desenvolvimento de uma aplicação *android* utilizando a tecnologia mvc

ALVES, Renan Augusto Muniz; LIMERES, Lucas Domingues; NDIAYE, Ely Assumpção;

AGRADECIMENTO

Primeiramente a Deus.

Aos nossos pais e noivas, pelo apoio, incentivo e orientação.

A todos os nossos professores, pelo apoio e incentivo.

E, principalmente, ao nosso orientador, Professor Jorge Luiz Chiara, que nos auxiliou do início ao fim do nosso projeto.

Somente obedecendo, somente tendo o orgulho humilde, mas sagrado, de obedecer, é que se conquista então o direito de comandar. Benito Mussolini – 41º Ministro da Itália.

RESUMO

ALVES, R. A. M.; LIMERES, L.D.; NDIAYE, E. A. **Desenvolvimento de uma aplicação *Android*utilizando atecnologia *MVC*. Estudo de caso: Acompanhamento de entrega.**2015. 43páginas. Trabalho de Conclusão de Curso de Graduação de Tecnólogo em Sistemas para Internet, Centro Estadual de Educação Tecnológica Paula Souza, Faculdade de Tecnologia da Baixada Santista “Rubens Lara”, Santos, 2015.

O aumento exponencial de clientes que realizam compras através de sistemas *delivery* vemgerando diversas frustrações.Pelo lado do cliente, por conta do longo prazo no aguardo na espera da entrega e a falta de informação sobre o pedido.Ao estabelecimento, por conta do mau atendimento, se torna mal visto pelo cliente não conseguindo obter uma relação de fidelização com o mesmo. Em busca de soluções para maximizar o *delivery* e eliminar estasfalhas, o presente projeto tem por objetivo permitir ao usuário, através de um aplicativo, realizar seu pedido e acompanhá-lo, desde a preparação até a entrega, trazendo-lhe segurança, interatividade e transparência na relação cliente-empresa e levar mais conforto e satisfação para o cliente na hora de realizar um pedido.

Palavras-chave: Entrega.*Android*. Tecnologia.

ABSTRACT

ALVES, R. A. M.; LIMERES, L.D.; NDIAYE, E. A. **Development of an*Android* application using the*MVC*. Case of study: Delivery Monitoring.**2015. 43 pages.Project of Technologist Graduate Course Completion in Internet Systems, State Center of Technological Education Paula Souza, Faculty of BaixadaSantista Technology "Rubens Lara", Santos, 2015.

The exponential increase in customers who perform procurement through delivery systems has generated many frustrations. On the client side, by the long-term account awaiting the expected delivery and the lack of information on the application. The establishment, due to the poor service becomes frowned upon by the customer failing to get a loyalty relationship with it .In search of solutions to maximize delivery and eliminate these flaws, this project aims to allow the user, through an application, place your order, accompany you from preparation to delivery, bringing you security, interactivity and transparency in the customer-company relationship, and bring more confort and satisfaction for the customer when carrying hold request.

Palavras-chave*:* *Delivery.Android.Technology.*

LISTA DE ABREVIATURAS E SIGLAS

ADT	*Android Development Tools*
APIs	*Application Programming Interface*
CPU	*Central Processing Unit*
DSDM	*Dynamic Systems Development Method*
EDGE	*Enhanced Data Rates for GSM Evolution*
GPS	*GlobalPositioning System*
GSM	*Global System for Mobile*
IDE	*Integrated Development Environment*
IOS	Sistema Operacional *Apple*
MMS	*Multimedia Message System*
MVC	*ModelViewController*
OHA	*Open Handset Alliance*
RAM	*RandowAcess Memory*
SDK	*Software Development Kit*
SMS	*Short Message Service*
SO	Sistemas Operacionais
VM	*Virtual Machine*

LISTA DE FIGURAS

Figura 1–Estrutura *Android*..... 15

Figura 2 –Ambiente de desenvolvimento *Eclipse SDK*..... 20

Figura 3 –Comparação das metodologias..... 21

Figura 4 –Descrição do processo *Scrum*..... 23

Figura 5 –Modelo *MVC*..... 26

Figura 6 –*MVC* no *Android*..... 27

Figura 7–*Scrum: ProductBacklog*..... 30

Figura 8–Código *View*..... 31

Figura 9 –Tela de *Login* (Aplicativo)..... 32

Figura 10 –Código *Model*..... 32

Figura 11 –Código*Controller*..... 33

Figura 12 –Código Mapa..... 34

Figura 13 –Tela Mapa (Aplicativo)..... 34

Figura 14 –Código Notificação..... 35

SUMÀRIO

1 INTRODUÇÃO 12

1.1 JUSTIFICATIVA DO TEMA 12

1.2 PROBLEMAS DE PESQUISA 13

1.3 OBJETIVO GERAL 13

1.4 OBJETIVOS ESPECÍFICOS 13

1.5 PROCEDIMENTOS METODOLÓGICOS 14

2 *ANDROID* 15

2.1 *ANDROID* EM NÚMEROS 15

2.1.1 Arquitetura 16

2.1.2 Aplicação 17

2.1.3 *Framework* 17

2.1.4 Bibliotecas 17

2.1.5 *AndroidRuntime* 18

2.1.6 *LinuxKernel* 18

2.2 RECURSOS DO ANDROID 19

2.3 DESENVOLVEDORES PARA ANDROID 20

2.4 VANTAGENS EM ANDROID 20

2.5 AMBIENTE DE DESENVOLVIMENTO 21

3 ENGENHARIA DE *SOFTWARE* 22

3.1 METODOLOGIAS ÁGEIS 22

3.1.1 Scrum 23

3.2 PADRÕES DE PROJETOS 25

3.2.1 Introdução a padrões de projetos 25

3.2.2 Padrões Compostos 26

3.2.3 Padrões de projetos e suas utilidades 26

3.2.4 *MVC* 26

3.2.5 *MVC* no *Android* 27

4 DESENVOLVIMENTO DE UMA APLICAÇÃO ANDROID UTILIZANDO A TECNOLOGIA MVC 29

4.1 A EMPRESA 29

4.2 OS APLICATIVOS 29

4.3 *SCRUM* 30

4.4 *MVC* 32

4.4.1 *View* 32

4.4.2 *Model* 33

4.4.3 *Controller* 34

4.5 *GOOGLE MAPS* 34

4.6 NOTIFICAÇÕES 36

5 CONSIDERAÇÕES FINAIS 37

REFERÊNCIAS 38

APÊNDICE A – Pesquisa de Campo (Empresa) 40

APÊNDICE B – Pesquisa de Campo (Usuários) 41

1 INTRODUÇÃO

O mercado de *Smartphones*no Brasil vem crescendo a cada ano com destaque ao sistema operacional *Android*, liderando esse mercado nos últimos 3 anos. Segundo a pesquisa realizada pela empresa americana *Gartner*, demonstra que do ano de 2011 até o ano de 2013 houve um crescimento de 47.8% nas vendas no mercado brasileiro.

Com esse crescimento, os usuários cada vez mais procuram utilizar os recursos tecnológicos através de aplicativos. Desde as necessidades relacionadas ao trabalho, vestuário, bancários, hoje também, o alimentício.

Atualmente, muitas pessoas se utilizam do sistema de *delivery* para poder receber sua alimentação sem necessitar ir até o local, perdendo tempo com o deslocamento, trânsito e etc. O sistema de *delivery, de* modo geral, o cliente liga para o estabelecimento, faz seu pedido e informa o endereço de entrega, forma de pagamento e telefone para contato. O funcionário do estabelecimento, por sua vez, recebe o pedido, cadastra no sistema do estabelecimento e o envia para a cozinha. Após o pedido ficar pronto é destinado ao entregador que, junto com outros pedidos, sai para realizar a entrega. Ao chegar no local de destino o cliente realiza o pagamento para o entregador e recebe seu pedido. Desta forma o processo do pedido é finalizado.

1.1 JUSTIFICATIVA DO TEMA

De acordo com a pesquisa de campo, Apêndice B, realizada na região da Baixada Santista*, 95% de um grupo de aproximadamente 128 pessoasgostaria de poder acompanhar o seu pedido pelo celular até o destino final de sua entrega lhe trazendo a confiança e a interatividade.

Desta forma, o presente projeto permite ao usuário, através de um aplicativo, realizar seu pedido e acompanhá-lo desde a preparação até a entrega.

*Conjunto de cidades localizadas na região do litoral sul do estado de São Paulo.

1.2 PROBLEMAS DA PESQUISA

Considerando o substancial aumento de clientes que utilizamo sistema de entrega, nota-se que a demanda máxima dos estabelecimentos que realizam esse serviço tem excedido seu limite. Isso acaba gerando diversas insatisfações ao clienteem relação ao longo prazo de entrega do produto ou serviço e a falta de informação, além de afetar de maneira direta o próprio estabelecimento.

O fator tempo é de intensa valia e determinante não somente para o prestador de serviço como também para o cliente. A demora excessiva para entrega do produto no ponto de destino causa um transtorno em cadeia que atinge a todos os participantes no sistema de entrega. O cliente insatisfeito com o tempo de espera entra em contato com o estabelecimento em buscade informações de seu produto e realizar reclamações, afetando a credibilidade do prestador de serviço, além do fato de que o tempo utilizado para atender a reclamação do cliente insatisfeito poderia ser utilizado para uma nova venda e fidelização e satisfação de mais um cliente.

1.3 OBJETIVO GERAL

Desenvolver um aplicativo, utilizando o sistema operacional *Android*, permitindo, assim, mais interatividade e transparência na relação empresa-cliente e poder levar mais conforto e satisfação para o cliente na hora de realizar um pedido.

1.4 OBJETIVOS ESPECÍFICOS

a) Utilizar a tecnologia *MVC*para o desenvolvimento da aplicação *Android*;
b) Promover uma interação entre aplicativo do cliente e o aplicativo do entregador, utilizando ferramentas de mapas e *GPS*;
c) Desenvolver um sistema de alerta para que o cliente seja avisado assim que o entregador saia do estabelecimento e quando estiver próximo ao local da entrega.

1.5 PROCEDIMENTOS METODOLÓGICOS

Como suporte as escolhas tecnológicas para o desenvolvimento do projeto, o presente trabalho utiliza-se de uma pesquisa bibliográfica, pesquisa de campo com questionário online, elaborado e aplicado para obtenção de informações sobre osusuários e estudo de caso.

2 ANDROID

O *Android* é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, *middleware*, aplicativos e *interface* do usuário. O *Android* foi construído com a intenção de permitir os desenvolvedores criar aplicações móveis que possam tirar total proveito do que um aparelho portátil possa oferecer. Foi construído para ser verdadeiramente aberto. Por exemplo, uma aplicação pode apelar a qualquer uma das funcionalidades de núcleo do telefone, tais como efetuar chamadas, enviar mensagens de texto ou utilizar câmera, que permite aos desenvolvedores adaptarem e evoluírem cada vez mais estas funcionalidades. (PEREIRA; SILVA, 2009, p. 3)

De acordo com (Lúcio & Michel 2009), por ser *open source*, pode ser adaptado a fim de incorporar novas tecnologias, conforme estas forem surgindo. A plataforma vai estar sempre em evolução, já que as comunidades de desenvolvedores estarão trabalhando em conjunto para construir aplicações móveis inovadoras. *Android* é o primeiro projeto de uma plataforma *open source* para dispositivos móveis em conjunto com a *OHA*. O *AndroidSDK* é o kit de desenvolvimento que disponibiliza as ferramentas e *APIs* necessárias para desenvolver aplicações para a plataforma *Android*, utilizando a linguagem *Java*.

2.1 *ANDROID* EM NÚMEROS

Por se tratar de um sistema operacional aberto, o *Android*pode ser utilizado em diversos aparelhos de marcas diferentes. Portanto, marcas como a *Motorola* e *Samsung* resolveram apostar no novo sistema, deixando o *Symbian* para trás. Isso fez com que uma série de celulares com o novo sistema operacional fosse disponibilizada no mercado, com preços que variavam de acordo com o bolso dos interessados.

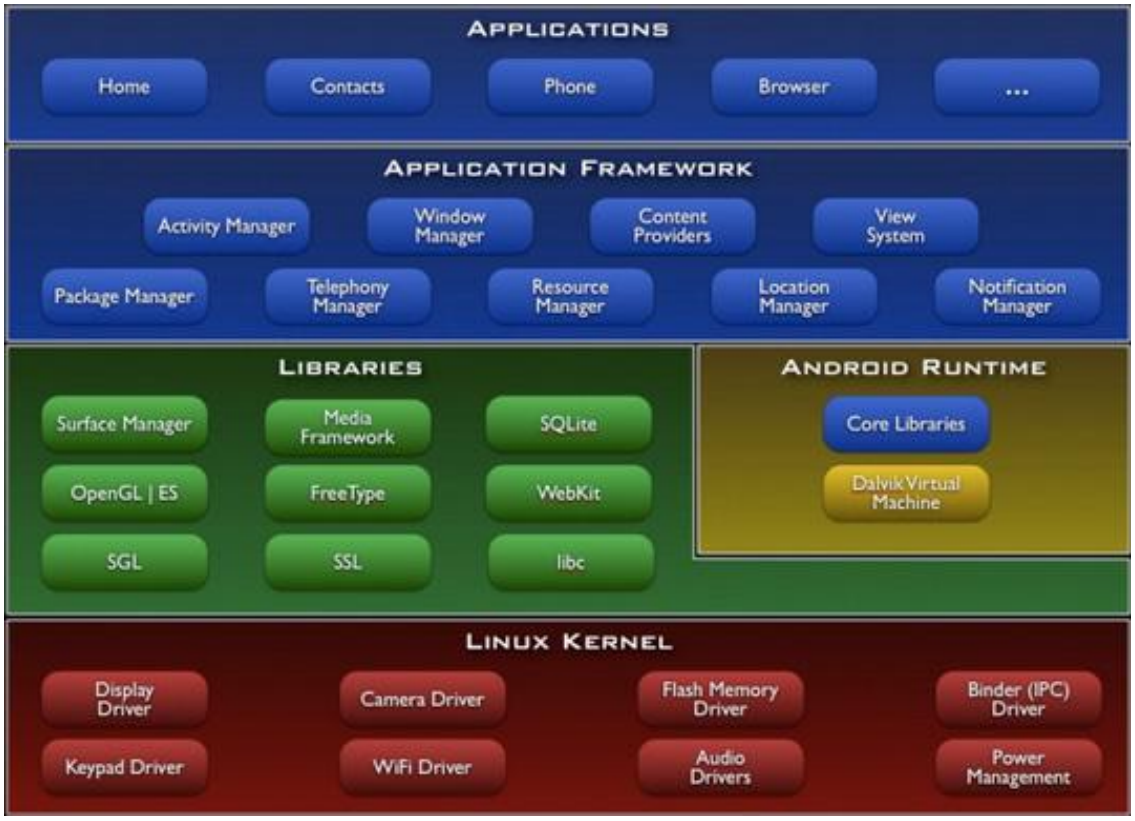
De acordo com a Gartner, Inc, um dos sites de pesquisas mais bem conceituados de tecnologia, mais de 67 milhões de aparelhos com sistema operacional *Android* foram vendidos em todo o mundo, ou seja, mais de 22% domercado mundial, enquanto as vendas do *Symbian* caíram em porcentagem, chegando a 37% do mercado.

A *Canalysis*, outra empresa conceituada de pesquisas, afirma inclusive que o *Android* ultrapassou a venda de aparelhos com *Symbian* no quarto trimestre do ano passado, ficando com quase 33% das vendas no mercado. Enquanto isso, a Nokia estaria com uma porcentagem de 30%, perdendo ainda mais a dianteira no número de aparelhos comercializados.

2.1.1 Arquitetura

Na figura 1 a seguir, de acordo com Lúcio e Michel (2009), podemos ver como é estruturada a arquitetura da plataforma *Android*. Observa-se um pouco sobre cada uma das camadas disponíveis desta grande plataforma e quais suas funcionalidades.

Figura 1 – Estrutura *Android*



Fonte: http://developer.android.com

2.1.2 Aplicação

Acima de todas as camadas se encontram todos os aplicativos fundamentais (escrito em *Java*) do *Android*, um cliente de e-mail, mapas, navegadores, calendários, programas de *SMS*, gerenciador de contatos, agendas, entre outros que serão desenvolvidos pela comunidade. Isso significa que, para desenvolver programas para a plataforma *Android*, os aplicativos serão criados em *Java* para serem executados na máquina *virtual Dalvik*. Lembrando que uma aplicação não precisa obrigatoriamente ser escrita para *Android*, já que o *Java* possui uma grande portabilidade. Outras aplicações desenvolvidas em Java podem ser incorporadas ao *Android*. (PEREIRA; 2009, p. 6)

2.1.3 Framework

Na camada do *framework* (*Application Framework*), encontramos todas as *APIs* e os recursos utilizados pelos aplicativos, como classes visuais, que incluem listas, grades, caixas de texto, botões e até um navegador *web* embutido, *View system* (componentes utilizados na construção de aplicativos), provedor de conteúdo (*ContentProvider*), que possibilita que uma aplicação possa acessar informações de outra aplicação, ou até mesmo compartilharem as suas informações, possibilitando a troca de informações entre aplicativos e gerenciadores de recursos (permite definir e carregar recursos em *RunTime*), gerenciador de localização (*GPS* e *Cell ID*), gerenciador de notificação (fornece informações sobre eventos que ocorrem no dispositivo), de pacotes e de atividade, que controla todo o ciclo de vida da aplicação e o acesso e navegação entre aplicações. (PEREIRA; 2009, p. 6)

2.1.4 Bibliotecas

Segundo Luiz (2014), as camadas de bibliotecas obtêm consigo um conjunto de bibliotecas C/C++ utilizadas pelo sistema, incluídas nesse conjunto da biblioteca C padrão e possui ainda, aquelas bibliotecas das áreas de multimídia, visualização de camadas 2D e 3D, funções para navegadores web, funções para gráficos, funções de aceleração de *hardware*, renderização 3D, fontes *bitmap* e vetorizadas e funções de acesso ao banco *SQLite*. Todos esses recursos estão disponíveis no *framework* para o desenvolvimento de aplicativos.

O *Android* possui bibliotecas desenvolvidas em C/C++, as quais se utilizam por vários recursos do sistema. Estas bibliotecas podem ser acessadas por *frameworks* disponibilizados para os desenvolvedores. (PEREIRA; 2009, p. 7)

2.1.5 *AndroidRuntime*

A pequena camada do ambiente de execução *(AndroidRuntime*) é uma instância da máquina virtual *Dalvik*, criada para cada aplicação executada no *Android*. A *Dalvik* é uma máquina virtual com o melhor desempenho, maior integração com a nova geração de *hardware* e projetada para executar várias máquinas virtuais paralelamente. Foi projetada para funcionar em sistemas com baixa frequência de *CPU*, pouca memória *RAM* disponível e sistema operacional sem espaço de *Swap*. Além disso, é otimizada para consumo mínimo de memória, bateria e *CPU*. (PFALZER, 2010, p. 23)

O *Dalvik* executa arquivoDEX (*DalvikExecutable*), que sãoclasses de *Java* convertidas para a máquina virtual através de ferramenta *DX*, distribuídajuntamente com o *SDK* do *Android*. O *Dalvik VM* baseia-se no *kernel* do *Linux* para funcionalidades subjacentes como o encadeamento e a gestão de baixo nível de memória. (PFALZER; 2010, p. 23)

O *Core Libraries* inclui um grupo de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas de linguagem *Java*, como estruturas de dados, acesso a arquivos, acesso a redes e gráficos. (PEREIRA; SILVA, 2009, p. 9)

Aplicações desenvolvidas em *Java* são compiladas em *bytecodesDalvik* e executadas usando a Máquina Virtual *Dalvik*, que é uma máquina virtual especializada desenvolvida para uso em dispositivos móveis, o que permite que programas sejam distribuídos em formato binário (*bytecode*) e possam rodar em qualquer dispositivo *Android*, independentemente do processador utilizado. (PEREIRA; 2009, p. 9)

2.1.6 *LinuxKernel*

Utiliza a versão 2.6 do *kernel* do *Linux* para os serviços centrais, tais como segurança, gestão de memória, gestão de processos, pilha de protocolos de rede e

modelo de *drives*. O *kernel* também atua como uma camada de abstração entre o *hardware* e o resto da pilha de *software*. Um acessório interessante é o *Blinder*, que é responsável por obter e enviar para a aplicação requerente a *interface* de serviço da aplicação requerida, possibilitando a Comunicação inter processos. (FILHO, 2014, p. 20)

Nesta camada se encontra um poderoso sistema próprio de gerenciamento de energia, onde um aplicativo requisita o gerenciamento de energia e o *driver* de energia do *kernel* passa a chegar periodicamente todos os dispositivos que não estão sendo utilizados por aplicações e os desliga. (PEREIRA; SILVA, 2009, p. 9)

2.2 RECURSOS DO ANDROID

Segundo (PEREIRA; SILVA, 2009, p. 9) os recursos do *Android* são:

A-*Framework* de Aplicação*:* permite a incorporação, a reutilização e a substituição de recursos de componentes;

B-Máquina Virtual *Dalvik*: máquina virtual otimizada para dispositivos móveis;

C-Navegador *Web* Integrado:navegador baseado no*engine open sourceWebkit*;

D-Gráficos Otimizados:gráficos e tratamento de imagens por meio de uma biblioteca 2D e gráficos 3D baseados na especificação *Open GL ES* 1.0 (aceleração de *hardware* é opcional);

E-*SQLite*: gerenciador de banco de dados para armazenamento de dados estruturados;

F-Suporte Multimídia: suporta formatos de som (*MP3, AAC, AMR*), vídeo (*MPEG4 e H.264*) e formatos de imagens (*JPG, PNG e GIF*) nativamente;

G-Telefonia *GSM*: permite fácil integração com as tecnologias *GSM*, mas apresenta limitações dependentes do *hardware*;

H-Protocolos de Comunicação *Wireless*: suporte à tecnologia *BluetoothEDGE*, 3G e *Wi-Fi*, porém possui as mesmas limitações de *hardware* do recurso telefonia *GSM*;

I-Câmera, *GPS*, bússola e acelerômetro:fácil integração com *hardwares* para câmeras embutidas e localização via *GPS*, como bússola e acelerômetro, ambos os casos dependem das limitações do *hardware*;

J-Poderoso Ambiente de Desenvolvimento:incluindo um emulador de dispositivo, ferramenta para depuração, analisador de memória e desempenho e um *plug-in* para a *IDE Eclipse* (*ADT*).

2.3 DESENVOLVEDORES PARA *ANDROID*

Segundo uma pesquisa realizada pelo site de pesquisa *Gartner (2013)*, a plataforma *Android* vem revolucionando o mercado para os usuários desenvolvedores, pois possui uma navegação fácil e grande de diversidade de recursos. O *Android*, por ser uma plataforma de código aberto, atrai diversas empresas e programadores independentes que estão desenvolvendo vários aplicativos e adaptando a plataforma de acordo com o *hardware* dos seus celulares, portanto, esta plataforma proporciona grandes inovações em dispositivos móveis.

Segundo (FILHO, 2014, p. 15) *Android* utiliza a linguagem de programação *Java* para a criação de aplicativos. Recentemente, a *Google* lançou a linguagem "*Simple*" que segundo o engenheiro de *software*HelbertCzymontek, o objetivo principal dessa nova linguagem é "facilitar a compreensão e o uso da linguagem para o mundo móvel e para a plataforma *Android*", linguagem essa que também veio para atender "o apelo de programadores não profissionais". Mas a linguagem predominante para a criação de aplicativos para *Android* é *Java*, com bibliotecas construídas pela *Google* especialmente para a programação dessa linguagem.

Segundo um levantamento com 464 desenvolvedores para dispositivos móveis revelou algumas informações curiosas sobre a distribuição de tempo para a criação de aplicativos para as plataformas *Android*, *iOS* e *Windows Phone*.

O sistema operacional da *Microsoft* para portáteis é a plataforma onde os desenvolvedores mais gastam o tempo com o *design* (30,2%), enquanto que para o SO do *Google*, o teste com procura por erros/*bugs* é o que mais consome o tempo dos profissionais (36,1%). Já para o "mundo" *Apple*, a codificação é onde os desenvolvedores mais necessitam de horas de trabalho, com 30,7% de todo o tempo. (LEE, Wei-Meng. Introdução ao Desenvolvimento de Aplicativos para Android. 1. Ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2011).

2.4 VANTAGENS EM *ANDROID*

"*A principal vantagem do Android, é que ele oferece uma abordagem unificada para desenvolvimento de aplicativos."*

"Internamente suporta as seguintes Funcionalidades:"

a) Armazenamento – usa o *SQlite* uma base de dados relacional leve, para armazenamento de dados;
b) Conectividade – suporta *WIFI*, *Bluetooth*, *GSM* e outras;
c) Troca de mensagens – suporta tanto *SMS* quanto *MMS*;
d) Navegador *Web* – baseado no *WebKit* de código aberto, juntamente com *JavaScript* V8 do *chrome*;

Suporte a meios – inclui *MP3, MP4, MIDI, JPEG, ETC*;

a) Suporte *a hardware* – sensor de acelerômetro, câmara bussola digital, sensor de proximidade;
b) Multitoques – suportas as telas multitoques;
c) Suporte *flash* – o *android* 2.3 suporta o *flash* 10.1;
d) *Tethering*– suporte ao compartilhamento de conexões de Internet como *hotspot* com e sem fios.

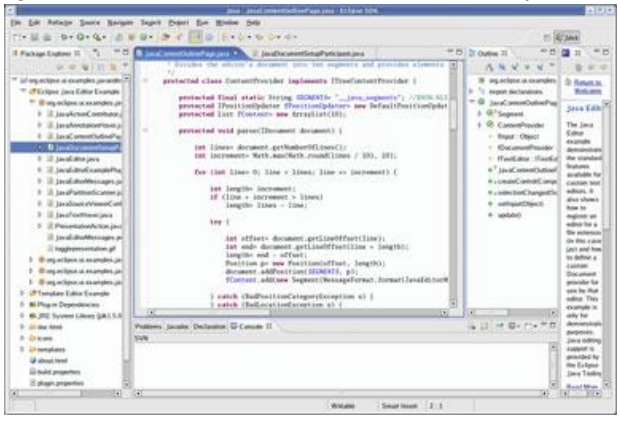
(LEE, Wei-Meng. Introdução ao Desenvolvimento de Aplicativos para Android. 1. Ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2011).

2.5 AMBIENTE DE DESENVOLVIMENTO

O ambiente de desenvolvimento mais utilizado para aplicações *Android*, atualmente éo *Eclipse*.Por ser uma IDE com um ambiente com conceito de Perspectivas, o Eclipse nada mais é do que agrupamento de funcionalidades que estão associadas a determinadas tarefas.

Cada perspectiva apresenta um visual próprio contendo abas que são importantes para determinada tarefa, e ainda podem ser customizadas pelo desenvolvedor da forma como nos agradar mais.

Figura 2 – Ambiente de desenvolvimento *Eclipse SDK*



Fonte:http://www.eclipse.org/ide/

3 ENGENHARIA DE *SOFTWARE*

A engenharia de *software* na prática profissional cresce dia após dia, por meio do somatório das experiências. De forma moderna, a Engenharia de *Software* caminha em paralelo com os Sistemas de Informação, ambos os temas destinados às organizações, para auxiliar as mesmas a tomarem decisões sob o foco do negócio empresarial.

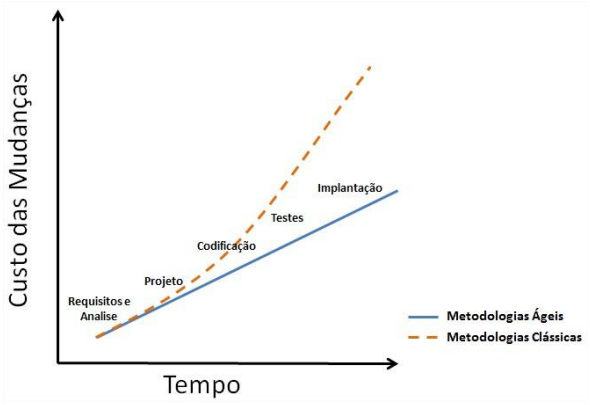
Segundo Sommerville(1992) a engenharia de *software* envolve questões técnicas e não técnicas, tais como a especificação do conhecimento, técnicas de projeto e implementação, conhecimentos dos fatores humanos pelo engenheiro de *software* e ainda gestão de projetos.

3.1 METODOLOGIAS ÁGEIS

Segundo Koscianski e Soares (2007) as metodologias ágeis são adequadas para situações em que a mudança de requisitos é frequente. Para ser realmente considerada ágil, a metodologia deve aceitar a mudança em vez de tentar prever o futuro.

Pode-se traçar uma comparação esquemática entre as metodologias clássicas e ágeis por meio da figura 3 abaixo, onde se analisa a relação entre custo de mudança e a fase em que esta ocorre. A linha pontilhada representa essa relação para uma metodologia clássica, enquanto a linha contínua mostra como se espera que a relação melhore nas metodologias ágeis.

Figura 3 – Comparação das metodologias



Fonte: http://audir.wordpress.com/

O termo metodologias ágeis tornou-se popular em 2001, quando 17 especialistas em processos de desenvolvimento de *software*, representando os métodos *Estreme Programming* (*XP), Scrum, DSDM, Crystal*, entre outros, estabeleceram princípios comuns compartilhados por todos esses métodos. O resultado foi a criação da Aliança Ágil e o estabelecimento do Manifesto Ágil. Enquanto as metodologias ágeis variam em termos de práticas e ênfases, compartilham algumas características como desenvolvimento iterativo e incremental, comunicação e redução de produtos intermediários, como documentação extensiva.

Os conceitos-chave do Manifesto Ágil enfatizam:

a) Indivíduos e interações em vez de processos e ferramentas;
b) *Software* executável em vez de documentação;
c) Colaboração do cliente ao invés de negociação de contratos;
d) Respostas rápidas a mudanças em vez de seguir planos.

O Manifesto Ágil não rejeita processos e ferramentas, documentação, negociação de contratos nem planejamento, mas simplesmente mostra que estes têm importância secundária quando comparados com os indivíduos, com o software executável, com a colaboração dos clientes e as respostas rápidas às mudanças. Esses conceitos aproximam-se melhor da forma como as pequenas e médias empresas trabalham e respondem às mudanças.

3.1.1 *Scrum*

Ametodologia ágil*Scrum*é um framework para desenvolvimento e sustentação de produtos complexos. Criado no inicio dos anos 90 por Jeff Sutherland e Ken Schwaber o *Scrum* tinha como objetivo agilizar os processos de desenvolvimento, com um resultado que fosse realmente útil para o cliente.

Segundo Ferreira, Costa, Alonso, Alves e Nunes (2005) o *framework Scrum* se desenvolve da seguinte maneira:

***Product Backlog*:** Lista de todas as funcionalidades a serem desenvolvidas durante o projeto completo, sendo bem definido e detalhado no início do trabalho, deve ser listado e ordenado por prioridade de execução.

***Sprint*:** Período não superior a 30 dias, onde o projeto (ou apenas algumas funcionalidades) é desenvolvido.

***Sprint Backlog*:** Trabalho a ser desenvolvido num *Sprint* de modo a criar um produto a apresentar ao cliente. Deve ser desenvolvido de forma incremental, relativa ao *Backlog* anterior (se existir).

***Scrum*:** Reunião diária onde são avaliados os progressos do projeto e as barreiras encontradas durante o desenvolvimento.

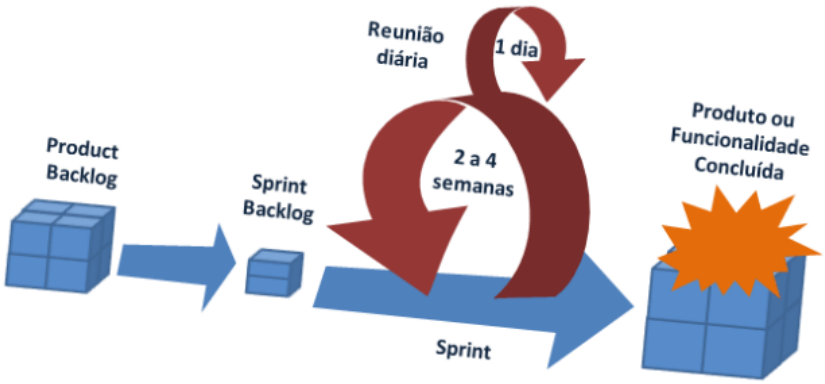
***ScrumMeetingRules*:**Protocolo a seguir de modo a realizar uma reunião *Scrum*.

***Scrum Team*:** A equipe de desenvolvimento de um *Sprint*.

***Scrum Master*:** Elemento da equipa responsável pela gestão do projeto e liderar as *ScrumMeetings*, são normalmente engenheiros de *software* ou da área de sistemas. Apesar de ser gestor não tem propriamente autoridade sobre os demais membros da equipa. É incentivada a autogestão.

A Figura 4, abaixo, ilustra todo o processo de *Scrum* detalhado acima.

Figura 4 – Descrição do processo *Scrum*



Fonte: http://www.mindmaster.com.br/scrum/

3.2 PADRÕES DE PROJETOS

A definição de padrão de projeto segundo Christopher Alexander: "defende a tese que cada padrão descreve um problema e o cerne da sua solução de tal forma que o mesmo possa ser utilizado quantas vezes necessário, sem utilizá-lo da mesma maneira" - (AIS-77).

Os padrões de projetossão divididos em quatros elementos principais:

a) O Primeiro elemento (Nome do Padrão):é uma referência que pode ser utilizada para descrever um problema, sua solução e consequências em palavras curtas, ou muitas vezes uma única palavra. Atribuir nomes aos padrões torna mais fácil o raciocínio e comunicação em relação aos projetos além de possibilitar altos níveis de abstração.
b) Segundo elemento (O problema): elemento que descreve a melhor situação para aplicar o projeto. Explica o problema e seu contexto. Em alguns momentos e possível que esse elemento inclui-a uma lista de condições quedevem ser satisfeitas para que faça sentido aplicar o padrão.
c) Terceiro elemento (ASolução):expõe elementos que compõe o padrão de projeto, seus relacionamentos, responsabilidades e colaborações. Fornece uma descrição abstrata de um problema de projeto e de como um arranjo geral de elementos (classes, e vantagens da aplicação do padrão).
d) Quarto elemento (As Consequências): Responsável pelos resultados e analises e desvantagens (*trade-offs*) da aplicação do padrão embora as consequências sejam raramentemencionadas, são construtivas para a avaliação de alternativas de projetos e benefícios da aplicação do padrão. (GAMMA; HELM; JOHNSON; VLISSIDES)

3.2.1 Introdução a padrões de projetos

O foco dos padrões do projeto não está em entrar diretamente no código, mas sim em entrar, primeiramente, na mente e carregar o cérebro com conhecimento dos padrões para poder aplica-los a novos projetos e a retrabalhar seu código antigo quando achar que ele está se transformando em uma mistura inflexível de código.

3.2.2 Padrões compostos

O nome dado a um conjunto de padrões trabalhando de forma unificada num design que pode ser aplicado a muitos problemas diferentes que podem ser repetitivos ou genéricos é denominado Padrão Composto. (FREEMAN; FREEMAN; 2009)

3.2.3 Padrões de projetos e suas utilidades

Em suma padrões de projeto são maneiras ágeis e simplificadas de reutilizar código orientado a objetos entre projetos e programadores. A ideia é simples: catalogar interações comuns entre objetos que são utilizados frequentemente pelosprogramadores. (GAMMA; HELM; JOHNSON, VLISSIDES)

3.2.4 MVC

MVC (Modelo, Visualização e Controle)esse padrão de projeto é um conjunto de padrões que trabalham em conjunto em uma determinada estrutura. (FREEMAN; FREEMAN; 2009).

Esse padrão utiliza o modelo *Observer* para manter as visualizações e os controladores informados sobre seu estado atual. A visualização e o controlador são implementados no Padrão *Strategy*.

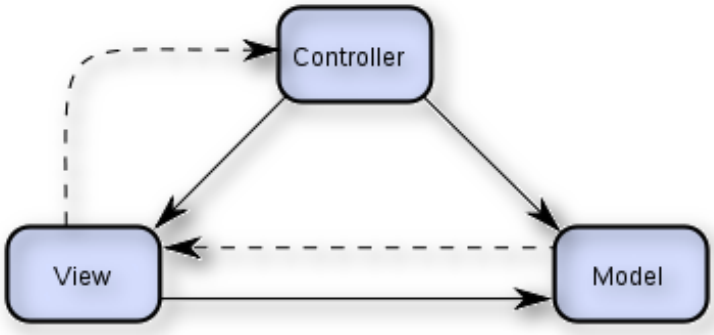
O controlador é o comportamento da visualização e pode ser trocadofacilmente. A visualização por sua vez utiliza um padrão para gerenciamento de janelas, botões e outros componentes da tabela chamado Padrão *Composite*.

A visualização fornece uma apresentação do modelo, a visualização obtém o estado e os dados que devem ser exibidos diretamente no modelo. (FREEMAN; FREEMAN; 2009)

O modelo é responsável por receber os dados e determinar o que esses dados iram significar para o sistema.

O controlador é responsável por conter todos os dados,estados e lógicas além de realizar a interação (*interface*) entre Visualização e controlador.

Figura 5 – Modelo MVC



Fonte: http://elias.praciano.com/2014/08/os-principios-do-mvc-para-desenvolvedores-php/

3.2.5 *MVC* no *Android*

O *MVC* basicamente separa a *interface* (*View*) do usuário das regras de negócio e dados(*Model*) usando um mediador (*Controller*) para obter comunicação entre o *Model* e a *View*.

O *Android*utiliza um padrão bem próximo ao *MVC*, com os arquivos *XML* agindo em conjunto com a *View*.

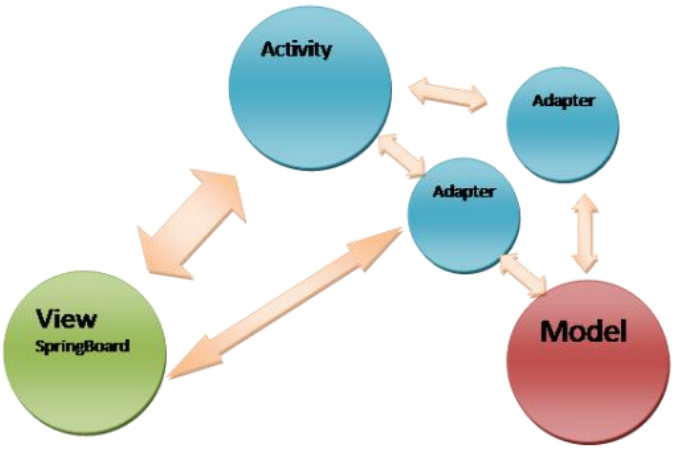
Existe uma integração do sistema *Android* com o *SQLite.* O *SQLite* possui um leve e poderoso banco de dados, o que possibilita utilizar o banco de dados normalmente na aplicação. Somente a aplicação que criou o banco pode visualizar o banco de dados.

No *SQLite* a *classe SQLiteHelper*éresponsável por encapsular toda a lógica de criação do banco, osmétodos*OnCreate* e *OnUpgrade* são chamados automaticamente pelo *Android* e o banco é criado pela primeira vez ounecessita ser atualizado devido a uma nova versão *Android*.

As *Actives* se encaixam como *Interface*(*View*) o pacote *controler* e suas classes e métodos(*Model*) são responsáveispelos processamentos dos dados recebidos pelas *actives*, e o pacote nomeadocomo *Controller*, realiza a interação e comunicação das *Actives*com os dados do *Model*. A figura 6 – *MVC* no *Android*,

exibe uma referência visual de como ocorre a interação entre o *Model, Viewe* o *Controller. (DIAS, J. L.F; 2014).*

Figura 6 – MVC no Android



Fonte:http://www.devmedia.com.br/android-mvc-criando-um-framework-model-view-controller-para-android/29924

4 DESENVOLVIMENTO DE UMA APLICAÇÃO *ANDROID* UTILIZANDO A TECNOLOGIA *MVC*

O estudo de caso foi especificamente baseado nas experiências e nos problemas da empresa Qui Panquecas.

Para se conseguir atingir o objetivo do presente projeto, foramdesenvolvidos dois aplicativos onde o cliente possa realizar o seu pedido e acompanhe todo o processo do seu pedido até a entrega ser efetuada. Com isso os funcionários ficam mais livres para se dedicarem a produção dos serviços e os clientes não ficam preocupados, pois estarão acompanhando passo a passo o andamento do seu pedido.

4.1 A EMPRESA

A empresa Qui Panquecas possui quatro anos de experiência no mercado de alimentação na Baixada Santista. Surgiu como uma nova proposta de sabor ao paladar das pessoas, com pratos deliciosos, feitos por profissionais experientes.Atualmente a empresa possui quatro filiais em toda a baixada santista. Sendo três em Santos e uma na Praia Grande. Além de suas tradicionais panquecas e beirutes, a empresa também possui em seu cardápio pizzas, strogonoff, refeições executivas, entre outros.

Segundo Georges, gerente na filial do bairro Campo Grande situado na cidade de Santos, as ligações de reclamação dos clientes é uma realidade muito preocupante. Segundo ele, o maior problema que a empresa enfrenta atualmente é quando o tempo se torna chuvoso, pois o número de ligações aumenta e o estabelecimento acaba se sobrecarregando. Consequentemente alguns pedidos atrasam, e com os pedidos atrasados os clientes resolvem ligar para reclamar, o que acaba atrapalhando mais ainda o andamento do serviço.

4.2 OS APLICATIVOS

O presente projeto desenvolve dois aplicativos – o do cliente e o do entregador. O aplicativo do cliente poderá realizar e acompanhar seu pedido e o

aplicativo do entregador fará a comunicação com o primeiro aplicativo para que o cliente possa acompanhar a entrega. Ambos os aplicativos estão diretamente ligados a um sistema web de acesso restrito aos funcionários do estabelecimento.

No primeiro aplicativo o cliente terá que, primeiramente, realizar o cadastro. Após estar autenticado no aplicativo, o cliente terá disponíveis as seguintes funcionalidades:

a) **Realizar o pedido:** Onde o cliente terá disponível uma lista de produtos, poderá selecionar os produtos desejados e realizar um pedido de compra, que deverá ser confirmado pelo estabelecimento.
b) Acompanhar o andamento do pedido: Após o pedido ser confirmado pelo estabelecimento, o cliente poderá verificar o status do seu pedido. Se ele já está sendo produzido, se ele já está disponível para entrega, se já saiu para entrega e se o entregador já está próximo de local de destino.
c) Atualização de Cadastro: Onde o cliente poderá realizar a atualizações de todos os seus dados cadastrais.
d) **Local do estabelecimento:**Onde o cliente poderá visualizar o endereço, através de ferramenta *Google Maps*, e todos os meios de contatos com o estabelecimento.

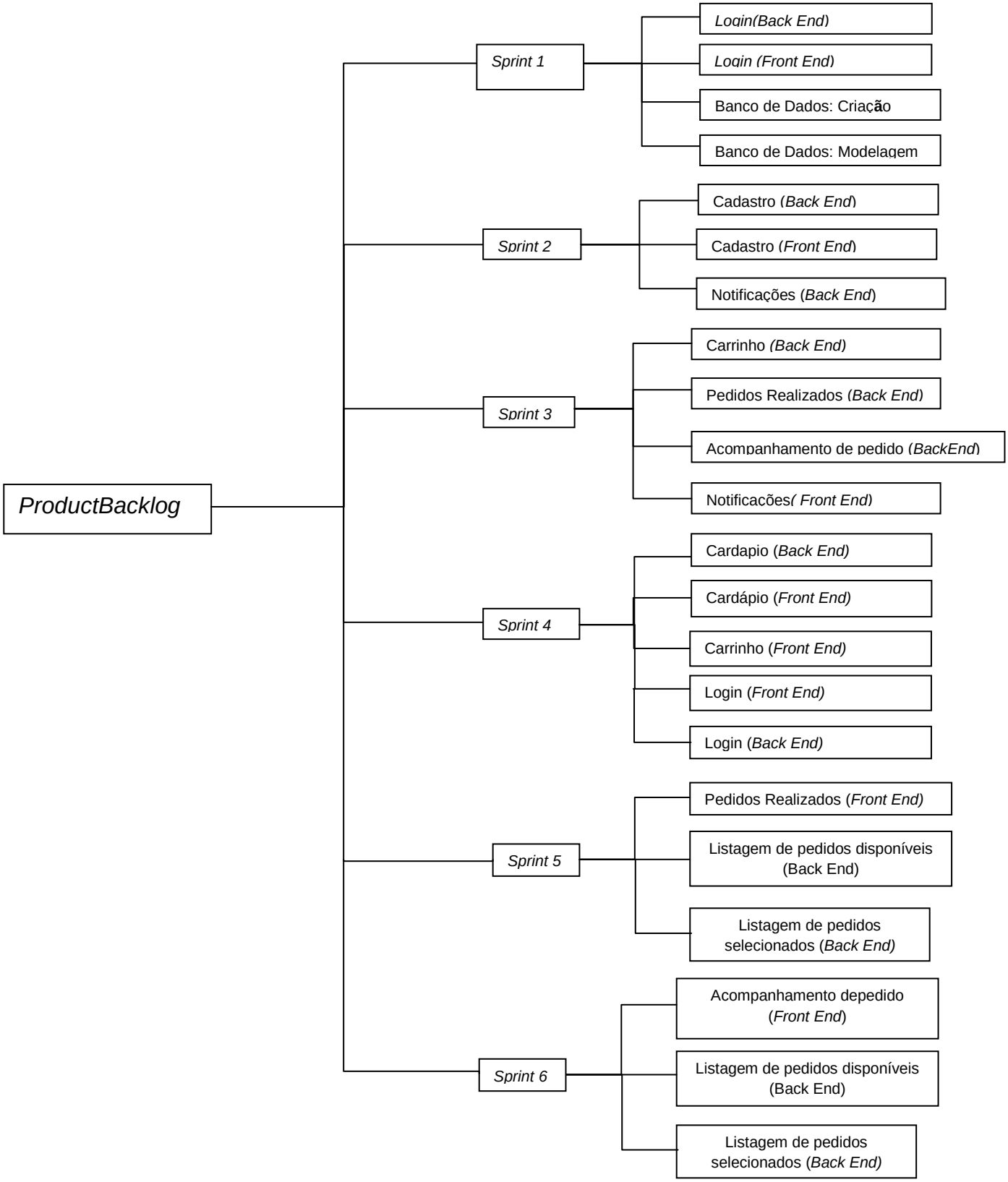
No segundo aplicativo, o entregador terá seu *Login* e Senha para a autenticação.Autenticado no aplicativo, o entregador terá disponível uma listagem de pedidos que estão prontos para serem entregues. O entregador então seleciona quais pedidos irá realizar a entrega naquele momento, e após todos os pedidos selecionados, poderá iniciar seu processo de entrega. Após iniciar sua entrega, o entregador poderá visualizar os pedidos que foram selecionados, podendo verificar o endereço do pedido, caso haja algum problema. E também podendo alterar o *status* do pedido, com as opções “Cancelar pedido”, “Devolver pedido” e “Finalizar Pedido”.

4.3 *SCRUM*

Para o desenvolvimento do presente projeto, foi utilizada a metodologia ágil *Scrum.* O primeiro passo é a definição de todas as tarefas que serão executadas no projeto, e em seguida dividi-las nas *Sprints*por ordemde desenvolvimento. Cada

*sprint*possuiu a duração de 15 dias, e ao final de cada *sprint* foi feita reuniões de equipe para a avaliação do andamento da *sprint*e planejamento para o início da próxima *sprint.* A figura 7, a seguir, mostra o *ProductBacklog*com todas as *Sprints*e suas respectivas tarefas.

Figura7 –*Scrum: ProductBacklog*



Fonte: ALVES, LIMERES, NDIAYE (2015)

4.4 *MVC*

O padrão de projeto utilizado no presente projeto, para o desenvolvimento do aplicativo, é o padrão *MVC*. Como foi explicado no capitulo 3, este padrão é dividido em 3 camadas: *View*, onde é desenvolvido os códigos que serão exibidos na tela, *Model*, onde é feita a conexão com o banco e onde é executada todas as regras do projeto, e *Controller*, onde faz o relacionamento da *View* com a *Model*. Segue, abaixo, um exemplo de cada camada que foi desenvolvida no projeto.

4.4.1 *View*

A figura 8, abaixo, mostra parte do código Java responsável pela exibição das informações na tela e receber todas as informações digitadas pelo usuário.Neste caso a tela se refere a tela de *login* do aplicativo, conforme mostra a figura 9.

Figura 8 – Código *View*

```
public void entrar(View v)
{
    String login = edlogin.getText().toString();
    String senha = edsenha.getText().toString();
    boolean autenticaLogin = loginController.autenticaLogin(login, senha);
    if (autenticaLogin){
        String nome = loginController.getNomeEmailUsario(login, "nome");
        String email = loginController.getNomeEmailUsario(login, "email");

        session.createLoginSession(nome, email);
        mensagemExibir("Login", "Bem Vindo");
        Intent it = new Intent(this, CardapioActivity.class);
        startActivity(it);
    }else{
        mensagemExibir("Login", "Usuário Inválido");
    }

}
```

Fonte: ALVES, LIMERES, NDIAYE (2015)

Figura 9 – Tela de *Login* (Aplicativo)

Fonte: ALVES, LIMERES, NDIAYE (2015)

4.4.2 *Model*

A figura 10, abaixo, mostra parte do código Java pertencente a *loginModel,* onde recebe as informações da *controller* e executa a função após a aprovação do *login.*

Figura 10 – Código*Model*

```
public boolean carregalogin(String login,String senha){
    boolean verificadoLogin = false;
    String urlPost="http://www.sistemaentrega.16mb.com/androidLogin.php";   //Servidor Web HOSTINGER
    //String urlPost="http://sistemadelivery.site90.com/androidLogin.php"; //   Servidor Web 000webhots
    ArrayList<NameValuePair> parametrosPost = new ArrayList<NameValuePair>();
    parametrosPost.add(new BasicNameValuePair("login",login));
    parametrosPost.add(new BasicNameValuePair("senha",senha));
    String retornaResposta = null;
    try{
        retornaResposta = ConexaoHttpClient.executaHttpPost(urlPost, parametrosPost);
        String resposta = retornaResposta.toString();
        resposta = resposta.replaceAll("\\s+","");
        if (resposta.equals("1")){
            verificadoLogin = true;
        }
    }
    catch(Exception erro)
    {
        Log.i("Erro", "erro = "+erro);

    }
    return verificadoLogin;
}
}
```

Fonte: ALVES, LIMERES, NDIAYE (2015)

4.4.3 *Controller*

A figura 11, abaixo, mostra parte do código Java responsável por receber as informações da *loginActivity (View)* e enviar para a *loginModel.*

Figura 11 – Código *Controller*

```
package com.controller.delivery;

import android.content.Context;
import android.content.Intent;
import com.model.delivery.LoginModel;
import com.view.dellivery.LoginActivity;

public class LoginController {
    LoginModel loginModel;

    public LoginController() {
        loginModel = new LoginModel();
    }

    public boolean autenticaLogin(String login, String senha) {
        // TODO Auto-generated method stub
        return loginModel.carregalogin(login, senha);
    }

    public String getNomeEmailUsario(String login, String parametro)
    {
        return loginModel.getNomeEmailUsario(login, parametro);
    }


}
```

Fonte: ALVES, LIMERES, NDIAYE (2015)

4.5 *GOOGLE MAPS*

Para possibilitar a interação entre o aplicativo do cliente e o aplicativo do entregador, utilizamos o *Framework System Location*onde recebe a latitude e longitude do entregador e mostra na tela, através de um mapa a localização exata do entregador. O código abaixo demonstra como o procedimento foi realizado.

Figura 12 – Código Mapa

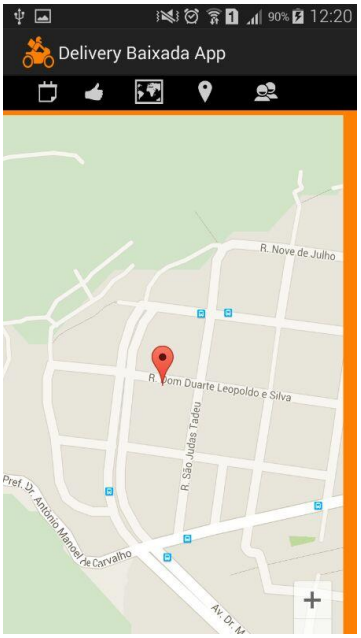
```
public void carregaMapa(){
    final LatLng frameworkSystemLocation = new LatLng(Double.parseDouble(latitude), Double.parseDouble(longitude));
    map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();
    Marker frameworkSystem = map.addMarker(new MarkerOptions().position(frameworkSystemLocation).title("Framework System"));

     //Move a Câmera para FrameWork System com zoom 15
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(frameworkSystemLocation , 30));
    map.animateCamera(CameraUpdateFactory.zoomTo(20), 2000, null);

}
```

Fonte: ALVES, LIMERES, NDIAYE (2015)

Figura 13 – Tela Mapa (Aplicativo)



Fonte: ALVES, LIMERES, NDIAYE (2015)

4.6 NOTIFICAÇÕES

Para possibilitar o envio de notificação ao cliente conforme seu pedido vai alterando de status, foi utilizado a classe *NotificationManager.* O código abaixo demonstra como o procedimento foi realizado.

Figura 14 – Código Notificação

```
@SuppressWarnings("deprecation")
public void notificacao(String titulo, String codPedido){
    verificaNotificacao = true;
    if(verificaNotificacao){
    final String mensagemBarraStatus = "Notificação Delivery";
    //Classe que voce Gostaria de Chamar Assim que clickado na notificacao.
    //Voce Pode colocar outra acitivity para outra tela aqui se desejar.
    final Class<?> activity = NotificacaoActivity.class;
    // Servico de Notificacao
    NotificationManager nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    //Escolhendo o Icone da Notificacao, Mensagem da Barra de Status e o Tempo que sera executado o horario.
    Notification notificacao = new Notification(R.drawable.ic_action_accept, mensagemBarraStatus, System.currentTimeMillis());
    // PendingIntent para executar a Activity se o usuario selecionar a notificação
    Intent intentMensagem = new Intent(this, activity);
    intentMensagem.putExtra("statusNotificacao", titulo);
    intentMensagem.putExtra("codigoPedido", codPedido);
    PendingIntent p = PendingIntent.getActivity(this, 0, intentMensagem, 0);
    // Reunindo todas as informacoes e Montando a mensagem e a notificacao
    notificacao.setLatestEventInfo(this, titulo, mensagem, p);
    // espera 100ms e vibra por 250ms.
    notificacao.vibrate = new long[] { 100, 250, 100, 500 };
    notificacao.defaults |= Notification.DEFAULT_LIGHTS;
    notificacao.defaults |= Notification.DEFAULT_SOUND;
    notificacao.flags    |= Notification.FLAG_AUTO_CANCEL;
    // id (número único) que identifica esta notificação. Mesmo id utilizado para cancelar
    nm.notify(R.string.app_name, notificacao);
    }
}
```

Fonte: ALVES, LIMERES, NDIAYE (2015)

5 CONSIDERAÇÕES FINAIS

Este projeto possibilitou, através de entrevistas e pesquisas de campo, observar e identificar diversos problemas que prejudicam estabelecimentos que utilizam sistemas *delivery*. O presente projeto visou solucionar estes problemas e frustrações e, para isto,foi desenvolvido um aplicativo que permite interatividade e transparênciana relação empresa-clientelevando conforto e satisfação para o cliente na hora de realizar seu pedido. O uso da metodologia ágil *Scrum* permitiu que o projeto fosse desenvolvido de uma forma organizada, se precavendo a qualquer imprevisto, e atingir o objetivo programado. Autilização da tecnologia *MVC*permitiuque o desenvolvimento dos aplicativos fosse realizado com códigos limpos e organizados para futuras manutenções. Para trabalho futuros propõe-se melhorias com o acréscimo de um gerenciamento de relacionamento com o cliente, permitindo, através de uma banco de dados, a criação de perfil dos clientes e possibilidade de efetuar o pagamento através do aplicativo.

REFERÊNCIAS

ALEXANDER, C. **A Pattern Language, Towns Buildings, Construction.** Nova York: Oxford University Press, 1977.

BECK, K. M. (2001). **Manifesto for Agile Software Development**. Disponível em:http://agilemanifesto.org/. Acesso em: 10 out. 2014.

DIAS, J. L.F.**Android MVC: Criando um Framework Model-View-Controller para vAndroid**. Disponível em: <http://www.devmedia.com.br/android-mvc-criando-um-framework-model-view-controller-para-android/29924#ixzz3Jdg1bnrG>. Acesso em: 10 out. 2014.

GAMMA, E; HELM, R; JOHNSON, R; VLISSIDES, J.**Padrões de Projeto: Soluções Reutilizáveis de Software orientado a objetos.**EbookKindle.1995.

FERREIRA, D; COSTA, F; ALONSO, F; ALVES, P; NUNES, T. **SCRUM – Um modelo ágil para gestão de projetos de software.**Faculdade de Engenharia da Universidade do Porto, 2005.

FILHO, C. Q.**ANDROID, Desenvolvendo seu Primeiro Aplicativo – "Entre de cabeça no mundo dos aplicativos móveis, criando e publicando seu próprio programa para o sistema líder do mercado!".** Novatec. 2014.

FREEMAN, E; FREEMAN, E.**Use a Cabeça! Padrões de Projetos (Design Patterns) – 2ª Edição Ed. Revisada.** Alta Books. 2009.

GARTNER. **Pesquisa de campo sobre o sistema operacional Android em comparação com outros sistemas para dispositivos Móveis. Disponível em:**<https://www.gartner.com/doc/2727718?ref=SiteSearch&sthkw=Android%20mobile&fnl=search&srcId=1-3478922254>

KOSCIANSKI, A; SOARES, M. D. S.**Qualidade de Software – 2ª Edição "Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software".**Novatec. 2007.

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

PEREIRA, L. C. O; SILVA, M. L. **Android paraDesenvolvedores**.Brasport. 2009.

PFALZER, S.**HELLO, ANDROID Introducing Google's Mobile Development Platform.**EdBurnette.2010.

SOMMERVILLE, I. **Software Engineering International Computer Science Series**.Edição 9, ilustrada.Pearson, 2011

APÊNDICEA - Pesquisa de campo(Empresa)

Nome: George Costa

Empresa: Qui Panquecas e Beirutes

Função: Gerente

Como é feito o sistema de entrega da sua empresa?

Simples através de um programa no computador. Atendemosao telefone registramos no computador e mandamos a comanda para a cozinha. Entãoo pedido é embalado e encaminhado para entrega.

Quais problemas e dificuldades são encontrados nesse sistema?

O maior problema é quando chove.Os clientes resolvem solicitar as entregaras todos ao mesmo tempo.Devidoà quantidade de pedidos retirados a cozinha acaba não suportando a demanda e o pedido acaba saindo já com atraso.

Os clientes ligam para cobrar a demora da entrega? Se sim, o quanto isso atrapalha?

Sim. Atrapalha por que acaba ocupando as linhas e tirando a concentração da equipe, pois o funcionário tem que verificar com quem a entrega esta, onde ele esta e quanto tempo o motoboy que estiver com a entrega atrasada vai demorar pra chegar aocliente. Tudo isso temos que responder para o cliente quando atrasa, ou seja, perdemos tempo.

A Empresa possui algum tipo de acesso mobile para os clientes? Se não, estaria interessado em possuir?

Não, acho interessante a ideia, pode ser viável para empresa.

Um aplicativo onde o cliente possa realizar o pedido e acompanhar a entrega ajudaria? Por quê?

Sim, por que ganharíamos tempo para preparar o pedido e o aplicativo deve informa-lo quando sair para entrega.

Qual outra funcionalidade também ajudaria no aplicativo?

Informa ao cliente o tempo percorrido a partir do momento que o pedido sai para entrega até residência do cliente.

APÊNDICEB - Pesquisa de campo(Usuários)

Pesquisa de campo - Sistema de acompanhamento de entrega

Instituição: Fatec "Rubens Lara"
Alunos:
Ely Assumpção Ndiaye - Matricula: 1210093-1
Lucas Domingues Limeres - Matricula: 1110103-9
Renan Augusto Muniz Alvez - Matricula: 1110110-1

* Required

Qual sua idade? *

Sexo *
- Feminino
- Masculino

Você utiliza o serviço de Delivery? (Pizzarias, Restaurantes, Etc.) *
- Sim
- Não

Com que frequência o pedido atrasa mais do que o previsto? *
- Nunca
- Aproximadamente 25%
- Aproximadamente 50%
- Aproximadamente 75%
- Sempre

Quando o pedido atrasa, você liga para cobrar? *
- Nunca
- Poucas vezes
- Muitas vezes
- Sempre

Você já cancelou o seu pedido por motivo de atraso? *
- Nunca
- Poucas vezes
- Várias vezes

Você acha que se pudesse monitorar seu pedido, ficaria mais tranquilo? *
- Sim
- Não

Caso sim, porque?

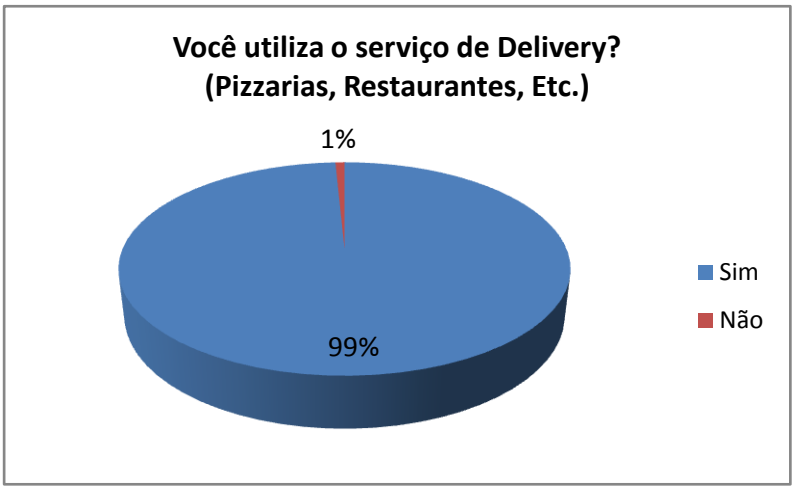
Você gosta de saber o andamento do seu pedido? (Se já está pronto, que horas saiu do estabelecimento, se já está próximo do destino, etc.) *
- Sempre
- As vezes
- Nunca

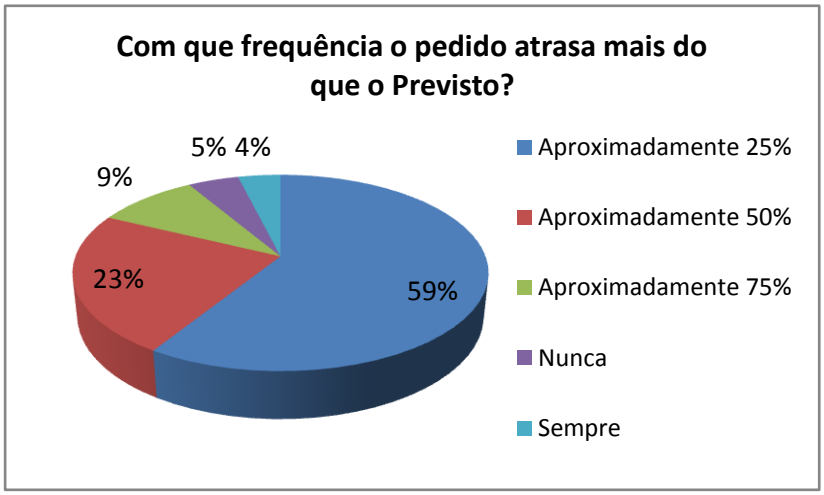
Caso houvesse opção, você realizaria seu pedido pelo Smartphone? *
- Sim
- Talvez
- Não

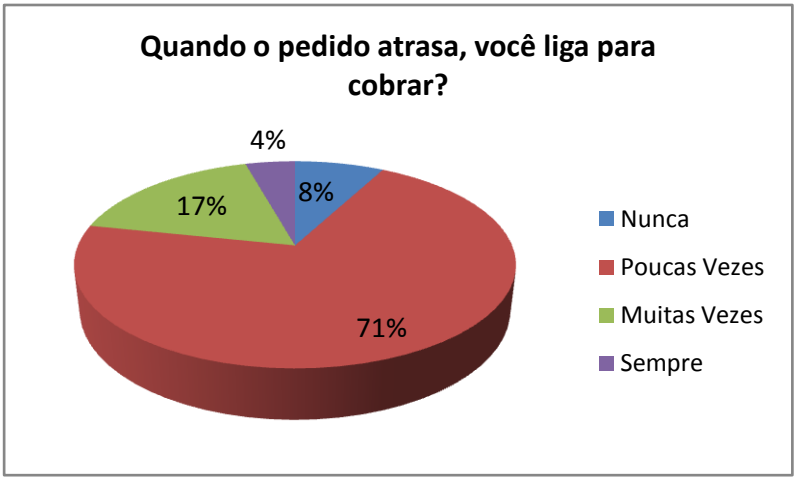
Submit

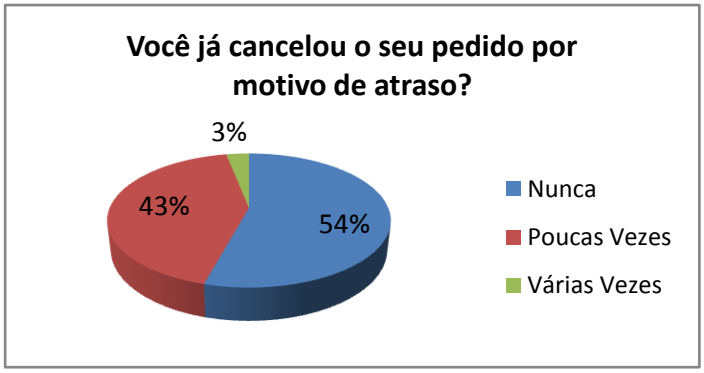
Never submit passwords through Google Forms.

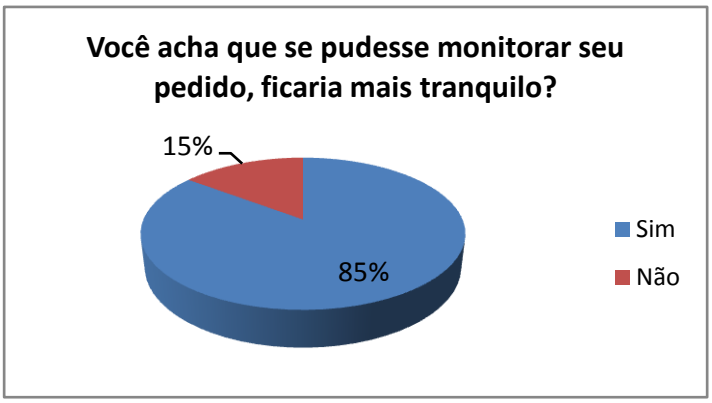
Resultados:

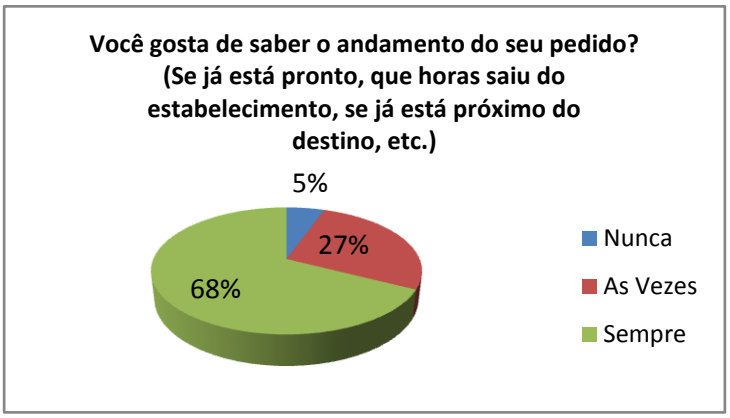

Você utiliza o serviço de Delivery?
(Pizzarias, Restaurantes, Etc.)
1%
99%
Sim
Não

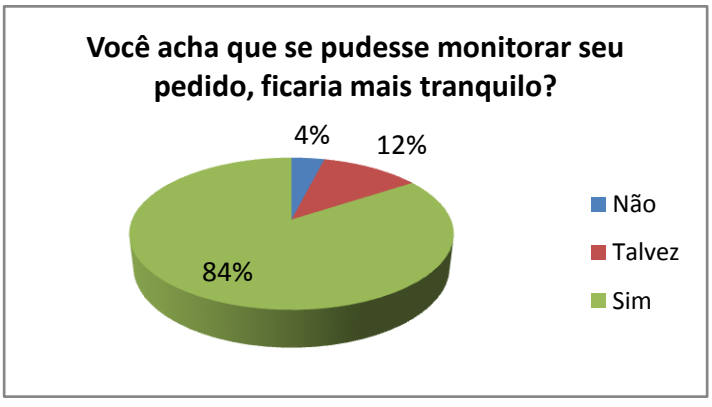

Com que frequência o pedido atrasa mais do que o Previsto?
5%
4%
9%
23%
59%
Aproximadamente 25%
Aproximadamente 50%
Aproximadamente 75%
Nunca
Sempre


Quando o pedido atrasa, você liga para cobrar?
4%
8%
17%
71%
Nunca
Poucas Vezes
Muitas Vezes
Sempre


Você já cancelou o seu pedido por motivo de atraso?
3%
43%
54%
Nunca
Poucas Vezes
Várias Vezes


Você acha que se pudesse monitorar seu pedido, ficaria mais tranquilo?
15%
85%
Sim
Não


Você gosta de saber o andamento do seu pedido? (Se já está pronto, que horas saiu do estabelecimento, se já está próximo do destino, etc.)
5%
27%
68%
Nunca
As Vezes
Sempre


Você acha que se pudesse monitorar seu pedido, ficaria mais tranquilo?
4%
12%
84%
Não
Talvez
Sim