FACULDADE ANHANGUERA DE JUNDIAÍ
CIÊNCIA DA COMPUTAÇÃO

PROJETO DO TRABALHO DE CONCLUSÃO DE CURSO

INTEGRAÇAO DE SISTEMAS *MOBILES* COM *WEBSERVICES* E BANCO DE DADOS

Danilo Gropelo
Islan Coelho
Ronaldo Trudes
Ana Helena Barreta
José Rubens Mazzola

Jundiaí
2010

FACULDADE ANHANGUERA DE JUNDIAÍ
CIÊNCIA DA COMPUTAÇÃO

INTEGRAÇAO DE SISTEMAS *MOBILES* COM *WEBSERVICES* E BANCO DE DADOS

Projetos de pesquisa apresentados para aprovação na disciplina Trabalham de Conclusão de Curso II.

Orientador: Cláudio Luís Vieira Oliveira
Orientandos:

Danilo Gropelo	8900097
Islan Coelho	0803268
Ronaldo Trudes	0803965
Ana Helena Barreta	0895771
José Rubens Mazzola	1141345190

Jundiaí, 17 de junho de 2011.

FACULDADE ANHANGUERA DE JUNDIAÍ

CIÊNCIA DA COMPUTAÇÃO

INTEGRAÇAO DE SISTEMAS *MOBILES* COM *WEBSERVICES* E BANCO DE DADOS

Projetos de pesquisa apresentados para aprovação na disciplina Trabalham de Conclusão de Curso I.

Orientandos:

Danilo Gropelo	8900097
Islan Coelho	0803268
Ronaldo Trudes	0803965
Ana Helena Barreta	0895771
José Rubens Mazzola	1141345190

_____

Orientador: Prof. Msc. Cláudio Luís Vieira Oliveira

Sumário

1. CARACTERIZAÇÃO DO PROJETO

O projeto tem a finalidade de comparar todo o processo de desenvolvimento de softwares nos sistemas operacionais móveis iOS e Android OS, utilizando recursos de acesso a WebService e Banco de dados remoto.

1.1 Área / Objeto

Área: Comparação no desenvolvimento para plataformas move
Objeto: O Estudo das diferenças no desenvolvimento de softwares para a plataforma Android e iOS, que resultará em um aplicativo no formato de rede social, onde cada usuário poderá visualizar locais de interesse próximos (restaurantes, bares, etc.), baseando-se nas informações de geo-localização do aparelho, colocando comentários e uma avaliação sobre o local e compartilhar as informações com outros usuários. .

1.2 Tema

Tema: Comparação no desenvolvimento de aplicações nas plataformas móveis Android e iOS.

1.3 **Problema**

Por serem plataformas pouco conhecidas, o Android e iOS acabam ignorados por muitos desenvolvedores, que optam por plataformas mais comuns para desenvolvimento. O Propósito deste projeto é desmistificar este conceito, compartilhar experiências e fazer com que desenvolvedores possam, baseando-se em nossos comparativos, identificar os prós e contras de cada plataforma e optar por aquela que mais se adapta às suas necessidades e as dos seus clientes.

1.4 **Objetivo**

O Principal propósito do projeto é comparar as etapas no desenvolvimento de uma aplicação para os dispositivos moveis Android e iOS, analisando quais as facilidades e

dificuldades de cada plataforma durante a codificação da aplicação e avaliando qual plataforma é mais amigável e eficiente na utilização de web services e banco de dados. .

1.5 **Hipótese**

Partimos da ideia de que muitos desenvolvedores cogitam desenvolver sistemas para aplicativos móveis, mas, por receio quanto à linguagem de programação utilizada ou dificuldades ao se adaptar, acabem não se aventurando neste novo terreno. Ao levantarmos esses estudos, qualquer desenvolvedor interessado em escolher o desenvolvimento para dispositivos móveis poderá tirar suas conclusões a partir deste projeto e decidir a melhor plataforma de desenvolvimento.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Contexto da Mobilidade

"A maioria dos usuários provavelmente usa *smartphone* como instrumento social. Telefona, confere *e-mails*, tuita, envia fotos, comunica-se. Essa, afinal, sempre foi à função do telefone." (MARQUEZI, 2010).

Fundamentando a afirmação acima há uma pesquisa a qual comprova que "Para cada 100 brasileiros há 95 celulares em operação" (KURA, 2010), com isso o desenvolvimento de aplicativos para divulgação de serviços na *web* aumentou.

Hoje a maioria dos fabricantes de aparelhos celulares está investindo em tecnologia, viabilizando recursos de hardware e software para acesso a serviços e redes sociais.

Isso também despertou a atenção das operadoras de serviço móvel, que estão investindo em suas redes e serviços para que possam manter os usuários cada vez mais conectados à internet, através das tecnologias *Wireless Application Protocol* (WAP), *Global System Mobile* (GSM), *Enhanced Data-Rates for Global Evolution* (EDGE), 3G e futuramente o 4G, diminuindo assim a distância entre pessoas e tecnologia.

Atualmente os produtores de softwares estão investindo cada vez mais em aplicações para plataformas móveis.

A tecnologia *mobile* esta sendo cada vez mais difundida entre as pessoas pela facilidade de acesso a informação, justamente pela questão da mobilidade, sabendo-se que qualquer lugar com acesso à *internet* possibilita ao usuário a interação com qualquer serviço da rede mundial de computadores.

Cada vez mais os ramos de negócios estão adotando as tecnologias *mobile*, criando assim novos adeptos para seus serviços.

2.2. Smartphones

Desde o surgimento dos computadores vem se estudando maneiras de redução de tamanho dos componentes eletrônicos e aumento de suas capacidades, daí surgiram os microcomputadores, notebooks e smartphones entre outras tecnologias.

Os smartphones, que ficavam em desvantagem em relação aos modelos de microcomputadores e notebooks com recursos de processamento e armazenamento muito superiores, acabaram superando todas as expectativas, e passou de um equipamento cuja função principal era a de fazer ligações, para um aparelho que, apesar de sua capacidade de processamento bem menor em relação aos computadores, acabou se destacando no meio das tecnologias mais usadas no dia-a-dia no mundo inteiro, pois, propicia ao usuário uma quantidade significativa de aplicações e serviços.

Os smartphones hoje representam 10% dos celulares existentes no mundo, e deverão crescer 32 % ao ano até 2014 segundo relatório da Analysis Mason1.

Hoje os serviços oferecidos por um smartphone são inúmeros, desde fazer ligações até mesmo auxiliar em projetos e serviços como vendas, atendimento a clientes, envio e recebimento de e-mails, acesso à web, acesso às redes sociais, bastando para isso uma rede de acesso a dados.

Atualmente, alguns smartphones já utilizam a tecnologia NFC (near-field communication) que permite efetuar pagamentos via celular, com a mesma funcionalidade de um cartão de crédito.

Não há argumento que vá contra o uso de smartphones já que estão ganhando cada vez mais o mercado, principalmente o corporativo, até mesmo por sua agilidade no acesso a informação e comunicação em diversos meios.

2.2.1. Android

O Android é a proposta da Google de um sistema operacional para aparelhos móveis, baseado em Linux que possui um ambiente de desenvolvimento muito poderoso e versátil. O sistema operacional do Android foi baseado no kernel 2.6 do Linux que é responsável pelo gerenciamento de memória, processos, threads, drivers, redes e arquivos.

O Android é um “open source”, ou seja qualquer pessoa pode desenvolver aplicações (“apps”) e as disponibilizar a outros usuários.

Isso é feito através do “Android Market”, uma loja online dirigida pelo Google onde se baixa aplicações para os Smartphones Android.

Entre as principais funções da plataforma android estão o acesso a internet e as aplicações do Android Market.

Ele oferece ainda serviços e aplicações Google, como a Pesquisa Google, Google Earth, G-mail, Google Docs entre outros.

As versões do OS Android estão evoluindo rapidamente e está atualmente na versão 3.0. Uma curiosidade é que a cada atualização, a versão recebe um nome de sobremesa.A versão 1.5 recebeu nome Cupcake, a 1.6 de Donut, a 2.0/2.1 de Eclair, a 2.2 de Froyo (Frozen Yogurt), a 2.3 de Gingerbread e a mais recente 3.0, voltada para o crescente mercado de tablets, recebeu o nome de Honeycomb.

O design do Honeycomb é bem diferente das versões anteriores do Android, se aproveitando das telas maiores dos tablets e em sua tela inicial, permite que aplicativos exibam conteúdo, como Twitter e Google Calendar. O sistema também melhorou a visualização dos chats por vídeo e apresenta interface gráfica diferentes para programas do próprio Google, como Gmail e Youtube.

O Android SDK vai permitir que os desenvolvedores elaborem as aplicações para os aparelhos de celular. Algumas das tecnologias suportadas pelo sistema operacional são: Touchscreen, Threaded (mensagens de texto), telefonia GSM, Câmera, GPS, bússola, acelerômetro (sensor de movimento), Bluetooth, EDGE, 3G(internet alta velocidade), e WiFi (internet sem fio) (que vão depender de hardware).

A plataforma apresenta suporte para mídias de áudio, vídeo e imagem, nos formatos MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF, bem como acelerador gráfico 3D, baseados no OpenGL ES. Os dados podem ser armazenados em SQLite e a pltaforma traz um navegador integrado com base no código livre do motor WebKit(mecanismo de layout, projetado para permitir web browser processar na página).

Um dos maiores motivos da fabricação do Android foi que o Google pensou em uma plataforma de software flexível desenvolvida para permitir uma experiência de usuário customizavel e personalizada em aparelhos celulares e com isso pode ser uma plataforma dominante entre os smartphones, pois agrada a 4 públicos diferentes: os fabricantes de celulares, os desenvolvedores, os fabricantes de chips e tem tudo para agradar também os consumidores. Em relação aos fabricantes de chips, novos recursos adicionados aos celulares, aumentará a procura por processadores mais rápidos, impulsionando o desenvolvimento e a venda de novos produtos. O Google obviamente ganha ao investir no desenvolvimento de um sistema operacional open-source para celulares e o distribuindo de graça. Isso acontece porque os aparelhos móveis são uma área bastante estratégica para a empresa, já que serve de atalho entre o público e os seus produtos.

Quando o Android foi anunciado causou uma grande expectativa, não só por ser uma aposta da Google, mas também por contar com o apoio de um grupo chamado "Open Handset Alliance" (grupo formado por grades empresas do ramo de aparelhos móveis como a Motorola, LG, Samsung, Sony Ericsson e muitas outras) ou apenas "OHA". Esse grupo foi formado com a intenção de padronizar uma plataforma de código aberto e livre para celulares, a fim de deixar os consumidores mais satisfeitos com o produto final e também criar uma plataforma moderna e flexível para o desenvolvimento de aplicações coorporativas. Todos se beneficiam dessa união: fabricantes, usuários e os desenvolvedores.

Para os fabricantes a principal vantagem é o fato de que a plataforma é única e de código aberto, com isso cada fabricante pode realizar as alterações que julgar necessárias sem que posteriormente sejam obrigados a repassar essas alterações para o restante do grupo, visto que a licença do Android é bastante flexível.

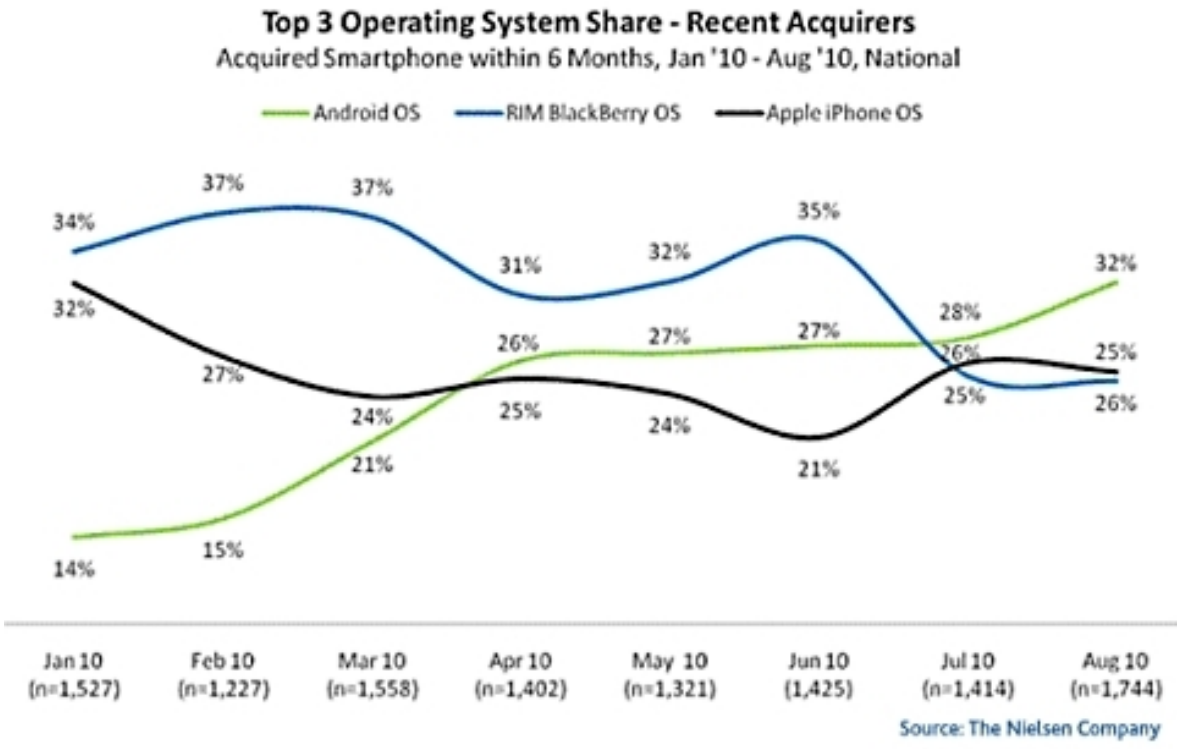
Os usuários sempre buscam um aparelho com muitos recursos, visual interessante e de fácil manuseio, e o Android proporciona tudo isso, pois por padrão é um sistema operacional feito para touch screen e em grande maioria os aparelhos que possuem o sistema instalado possuem GPS, acesso a internet, jogos, entre outros.

Para os desenvolvedores as vantagens são varias, em primeiro lugar o fato de existir um padrão para poder desenvolver para muitos aparelhos de diversas marcas. Outro fato importante é de que todo o SDK é baseado em Java, ou seja, diferentemente do iOS em que os desenvolvedores precisam aprender uma linguagem praticamente do zero, o desenvolvedor pode se utilizar de uma linguagem de programação já bastante conhecida e uma das principais no mercado hoje em dia. Além disso a plataforma de desenvolvimento é moderna e conta com

muitos recursos, seja para o acesso a banco de dados SQLite, para a utilização do hardware de GPS, integração com câmera fotográfica, envio de SMS, entre outros.

Uma das receitas de sucesso do Android até aqui é o Android Market, que é o canal de distribuição de aplicativos do sistema operacional Android, onde o desenvolvedor disponibiliza seu aplicativo para que seja baixado e utilizado pelos usuários finais, sem precisar publicar esse aplicativo em outros meios de comunicação como, por exemplo, sites de downloads. Para poder distribuir seus aplicativos através do Market, o desenvolvedor deve realizar um cadastro junto a Google e pagar uma taxa de USD 25.00, em seguida o desenvolvedor já estará apto para publicar aplicativos gratuitamente. Para o caso do desenvolvedor colocar um custo no aplicativo e vendê-lo, será necessário realizar um segundo cadastro, onde poderá ser aprovado ou não pelo Google, caso aprovado terá seu aplicativo publicado com um determinado custo definido por ele próprio. Segundo Cláudia Tozetto do site de ténologia do IG de São Paulo, o Android Market já conta com mais de 100 mil aplicativos disponíveis para download onde seu principal concorrente, a App Store do iOS, conta com mais de 280 mil.

Segundo uma pesquisa feita pela consultoria americana The Nielsen Company, hoje o Android é o sistema operacional mais popular entre os smartphones comprados nos Estados Unidos nos últimos seis meses, conforme o gráfico a seguir:



Em relação ao kernel do Android, cada aplicativo iniciado no sistema operacional cria um novo processo, disparando uma tela para o usuário ou simplesmente sendo executado como um serviço em segundo plano (por exemplo, um cliente de e-mail, que fica sempre buscando novos e-mails no servidor) enquanto o usuário pode continuar utilizando outros processos normalmente. Quanto à finalização dos processos, o próprio sistema operacional pode decidir finalizar processos que não estão sendo usados (tanto pelo usuário quanto pelo sistema operacional) a fim de liberar mais memória para os outros processos em execução.

2.2.2. iPhone

O iPhone é um smartphone desenvolvido pela Apple Computer Inc., lançado em 29 de junho de 2007, que une em um único aparelho recursos de celular, dispositivo de acesso a redes sem fio e iPod. Sua principal característica é a utilização de uma tela sensível ao toque para acesso e manipulação de diversos recursos do celular, como aplicativos, teclado virtual, fotos e mapas.

As versões do aparelho desde o seu lançamento são, em ordem cronológica, o iPhone, iPhone 3G, iPhone 3GS e iPhone 4, sendo que este último foi lançado em Junho de 2010 nos EUA.

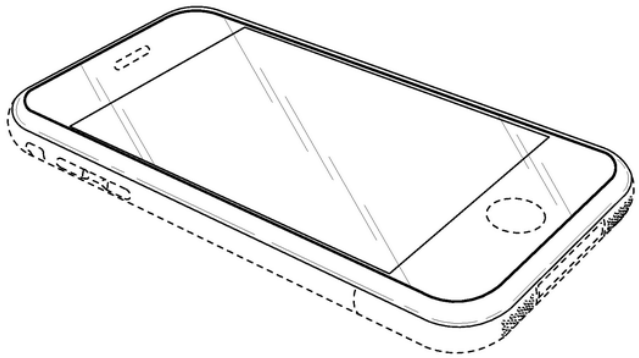


Figura 1: iPhone

"O iPhone 4 utiliza o sistema operacional iOS 4, possui processador Apple A4, tela retina display de 3,5" com resolução de 960x640 pixels, câmera, até 32GB de memória interna, GPS, Wi-Fi 802.11b/g/n, Bluetooth, acelerômetro, bússola, giroscópio, além de possuir player de áudio e vídeo.

Inicialmente chamado de iPhone OS, o iOS é o sistema operacional utilizado nos dispositivos móveis da Apple, que está atualmente na versão 4.3.3. Seu kernel é uma variação do kernel chamado Mach, encontrado no Sistema Operacional Mac OS dos desktops da empresa.

O iOS possui as seguintes camadas de implementação: Cocoa touch, Media, Core Services e Core OS. Nas camadas superiores ficam os serviços e tecnologias mais sofisticadas e, nas camadas mais baixas, os serviços fundamentais do SO que as aplicações podem utilizar.

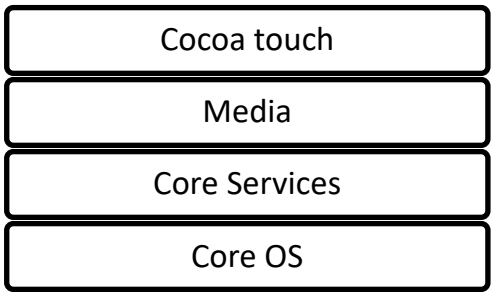


Figura 2: Camadas do iOS

Na camada Cocoa touch encontram-se os principais frameworks para desenvolvimento de aplicações, como suporte ao multi touch, notificações push e multitarefa. Nesta camada estão os frameworks UIKit e Foundation que oferecem a infraestrutura básica para o desenvolvimento de aplicações gráficas. A camada Media armazena as tecnologias para manipulação de gráficos, áudio, etc. A camada Core Services armazena os serviços fundamentais utilizados por todas as aplicações, como biblioteca SQLite, suporte a XML, acesso a Contatos e recursos do telefone. A camada Core OS possui todos os recursos de baixo nível usados pelos outros serviços e tecnologias, como segurança, threading, acesso a redes, informações de localização, etc. [Apple Computer 2011 (1)].

As aplicações para o iPhone e similares são desenvolvidas utilizando a linguagem Objective-C, derivada da linguagem C, com recursos de programação orientada a objetos, também usada no desenvolvimento de aplicações para o Mac OS. Em março de 2008 a Apple disponibilizou o primeiro SDK de desenvolvimento para iPhones, chamado de iPhone SDK (hoje, iOS SDK). O iPhone SDK era composto por uma IDE com compilador para Objective-C, com frameworks que permitem utilizar todos os recursos disponíveis no aparelho, um Simulador de iPhone, biblioteca de classes e ferramentas para a construção da interface para a aplicação, chamado Interface Builder [Dalrymple e Knaster 2008].

As aplicações desenvolvidas para iPhone são disponibilizadas em uma loja virtual chamada App Store. A loja, que é exclusiva para aplicativos dos dispositivos da Apple, conta atualmente com mais de 350.000 programas para iPhone, iPod Touch e iPad, que podem ser baixados diretamente pelo celular através de aplicativo próprio, bastando possuir um id Apple e uma senha [Apple Computer 2011 (2)].

O desenvolvimento de aplicações para os dispositivos é bastante restrito, só sendo possível se o desenvolvedor possuir um computador Apple (MacBook, iMac, Mac mini ou Mac Pro), uma vez que o SDK só roda nestes computadores. É necessário também criar um id de desenvolvedor e associar-se ao serviço pago anual do iOS Developer Program, que atualmente custa USD 99.00. O iOS Developer program foi criado para permitir que

desenvolvedores distribuam seus aplicativos e troquem conhecimento sobre a criação de softwares.

É possível também fazer somente o registro gratuito como desenvolvedor. O registro possibilita o download do SDK, porém só permite o desenvolvimento de aplicações para testes. Tornando-se membro do iOS Developer program o desenvolvedor poderá ter acesso ao SDK mais recente, testar o software no aparelho e publicar a aplicação na App Store para distribuição.

2.3. Redes Sociais

Redes sociais são grupos de pessoas que se unem e se comunicam de alguma forma. E através da internet possuem também a possibilidade de se expressar por meio de vários recursos como vídeo, foto, *posts*, mensagens instantâneas e aplicativos (como jogos e animações).

Através das redes sociais é possível criar um usuário, definindo suas características (podendo também modificar alguns aspectos), ou criar um personagem. Por se tratar de um ambiente virtual, as características não são necessariamente reais, o que torna mais fácil a comunicação para algumas pessoas do que o contato pessoal.

O principal objetivo é a comunicação e interação entre os usuários que se conhecem, e a amplitude da rede de amigos. As ferramentas permitem o compartilhamento de arquivos, fotos, vídeos, textos, envio de mensagens (*on line* ou *desligado line*) e links, possibilitando aos usuários a liberdade de opinião e expressão.

Surgiram no ano de 1970. O primeiro site de relacionamento foi o *SixDegress* que possibilitava a utilização de um perfil, listagem de contatos, comunicação dos usuários e visualização de quaisquer perfis da rede.

Novos sites de relacionamento foram criados, os principais foram *Friendster (2002), My Space (2003), LinkedIn (2003), Orkut (2004), Facebook (2004)* e *Twitter (2006).*

Diferente das demais redes citadas, o *LinkedIn* tem o objetivo de unir profissionais. Neste site é possível realizar indicações dos perfis conhecidos, exibição do histórico profissional, localização de usuários para troca de informações, compartilhamento de experiências, discussão de informações e tira dúvidas com os fóruns.

As redes são também um banco de conhecimento. Tanto para instituições que precisam de informações sobre o público, como para usuários que encontram disponível na rede dados de qualquer outro usuário associado. Muitas empresas estão participando das redes sociais com o objetivo de divulgar seus produtos, sua marca e possuir com essas novas tendências mais relacionamento com seus clientes.

Para os desenvolvedores das redes sociais o grande ganho está no patrocínio que recebem. Por terem milhões de adeptos seus produtos são cobiçados, já que a propaganda dos patrocinadores está disponível a todos os usuários.

Nas atuais redes sociais (como *Facebook*, *Twitter*, *Orkut*) é possível ficar conectado 24 horas por dia fazendo *login* uma única vez, pois há conectividade dos aparelhos com sistemas operacionais *Android* e *IOS* através de aplicativos, enquanto conectado qualquer ação recebida da rede é exibida no aparelho. Estes aplicativos, como por exemplo, do IOS estão

disponíveis na loja virtual *App Store* [Apple, http://www.apple.com/br/ipad/from-the-app-store/social.html].

2.4. PHP

PHP, que significa "PHP: Hypertext Pre-processor" é uma linguagem de programação voltada para desenvolvimento web, embora com uso de algumas bibliotecas também possibilite o desenvolvimento para desktop.

O PHP hoje é muito utilizado por sua facilidade no desenvolvimento e também pode ser facilmente mesclada com código HTML.

O PHP conta com características como:

- Velocidade e robustez;
- Pode ser utilizado como desenvolvimento estruturado ou orientado a objeto;
- Independência de plataforma;
- Sintaxe similar com a de outras linguagens como C/C++++, Perl e Java, tornando-se uma linguagem de fácil compreensão.
- Além de tudo é uma ferramenta *open-source*, ou seja, de código aberto e gratuito.

Para o projeto em questão, será utilizado o PHP na criação de um *middleware* para integrar o *webservice* diretamente a aplicação desenvolvida no Android e iPhone.

2.5. Web Services e SOAP

Web Services (Serviços *Web*) são métodos escritos em uma determinada linguagem de programação e armazenadas em um servidor web, que permitem a comunicação entre cliente e servidor, independente da linguagem de programação em que é utilizada no servidor ou no cliente e a plataforma em que será executado. Devido à utilização de protocolos como o SOAP (*Simple Object Access Protocol*), no qual a comunicação entre eles ocorre no padrão de sintaxe XML (*Extensible Markup Language*) para codificar os dados, torna-se possível a integração de sistemas e comunicação de aplicações em plataformas diferentes, onde o principal ponto está na padronização da troca de mensagens e no modo de estruturar a comunicação entre cliente e servidor. Assim, cada aplicação pode manter sua plataforma e linguagem especifica, mas ao estabelecer comunicação com o outro lado, esta troca de informações deve utilizar um mesmo padrão. Facilitando a construção das aplicações de cada um e, onde um mesmo *web service* pode ser utilizado por vários clientes, atendendo várias requisições.

Utilizando um *web service* as aplicações ficam mais simples, onde não é necessário que toda a implementação de funções fiquem no seu escopo. Estes métodos ficam no *web services* possibilitando que várias aplicações apenas façam chamadas aos métodos armazenados. O processo funciona da seguinte forma, o cliente cria sua requisição, envia ao servidor, uma aplicação no servidor localiza o *web service* disponível, é então enviada a mensagem (através do protocolo SOAP) com as requisições. O serviço (*web service*) ao receber esta requisição a processa e retorna uma mensagem para o servidor de aplicação, este servidor processa os dados necessários e retorna ao cliente. Todas as mensagens são trocadas pelo protocolo SOAP.

Essa comunicação pode ocorrer de várias formas, tanto utilizando um *browser* (navegador de páginas pela internet) onde o cliente acessa uma página, e, efetuando alguma ação, como preencher um formulário e executar uma consulta, é estabelecida uma comunicação com o *web service*. E, ocorre também diretamente, através de uma aplicação cliente, por exemplo, um ERP que precise verificar a restrição de crédito de CNPJs, esta solicitação é enviada ao *web service* que processa a informação e envia o retorno ao ERP (se tem restrição de crédito ou não).

SOAP (em português: protocolo simples de acesso a objetos) foi inicialmente proposto pela IBM, Ariba e Microsoft. Tem o objetivo de efetuar a troca de dados entre cliente e servidor. Sua vantagem em relação aos outros protocolos já existentes, como DCOM (*Distributed*

Component Object Model) e CORBA (*Common Object Request Broker*), é que utiliza a porta 80 para a troca de mensagens entre os aplicativos, não sendo bloqueada pelo *firewall*. Essas transferências de mensagens são feitas no protocolo de transporte pela internet HTTP (*Hypertext Transfer Protocol*). O SOAP é um protocolo baseado em XML para a troca de informações, junto a isso com a utilização do HTTP possibilita esta comunicação em um ambiente distribuído.

O SOAP possui duas definições para tratativas das requisições, um mecanismo baseado em XML pra a troca de mensagens, e outro com chamadas de procedimento remoto (RPCs) que são chamadas locais a métodos de objetos ou serviços remotos, que permite acessar os serviços de um objeto da mesma rede.

Para a elaboração de uma mensagem SOAP, será necessário utilizar pelo menos os elementos envelope, *header* e *body*. O envelope contém declarações dos *namespaces* e também atributos adicionais do estilo de codificação (que define como os dados serão apresentados no XML). O *header* é opcional, contem informações adicionais importantes para o tráfego na rede. E, o *body* pode conter um elemento chamado Fault, que é utilizado para passar as mensagens de erro que são geradas durante o processamento da mensagem.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Figura 3: Esqueleto de uma mensagem SOAP – Disponível em http://www.w3schools.com/soap/soap_syntax.asp

WSDL (*Web Services Description Language*) é um arquivo baseado em XML que contem informações descrevendo a sintaxe do serviço *web*. É uma espécie de biblioteca de serviços, nele está definido o que um serviço pode fazer, onde residem, quais parâmetros são utilizados, interfaces e como chamá-lo. Estas são as informações necessárias para que ocorra a

comunicação entre a aplicação e o *web service*, de modo que as informações enviadas pelo cliente sejam encontradas. O WSDL é responsável por definir o XML *Schema* ao descrever o *web service*, de modo que o arquivo gerado pelo cliente seguindo esta estrutura definida pelo WSDL contenha todos os dados necessários e organizados possibilitando a comunicação de cliente e *web service*. Como o serviço remoto já esta configurado para receber informações de um determinado modo, para que todos os clientes consigam construir o XML com os parâmetros necessários para envio das mensagens, utiliza-se o WSDL na tradução das informações que irão acopladas ao XML.

Apesar destas funcionalidades, o *web service* não possui definido qual o mecanismo de segurança padrão, comprometendo a autenticidade das transações entre cliente e servidor, a privacidade das mensagens e integridade de que as informações não sejam alteradas durante a comunicação, a única vantagem nesta questão é que torna mais simples a implementação. Mas é utilizado com grande aceitação pelo mercado, pelo fato de utilizar o XML como padrão na representação dos dados, já que o XML também está sendo utilizado como padrão universal, que é processado pelos sistemas e possui uma fácil interpretação visual das informações contidas em seu escopo. E também pelo fato de utilizar o HTTP como protocolo de transporte, que garante que o web Server atenda as requisições do web service.

2.6. Banco de Dados

Banco de dados nada mais é sistema de “armazenamento” que registra, reúne e organiza uma série de informações relacionadas a um determinado assunto em uma determinada ordem. Seu principal objetivo é possibilitar um ambiente que seja adequado e eficiente para o uso na recuperação e armazenamento de informações inter-relacionadas. Um exemplo seria uma ficha de um acervo de biblioteca ou até mesmo uma lista telefônica.
Um Banco de dados apresenta 4 modelos de dados que são: entidade, atributo, relacionamento e cardinalidade.

Outro fator importante é o SGBD - uma coleção de programas que permite ao usuário definir, construir e manipular Bases de Dados para as mais diversas finalidades.

2.6.1. MySQL

O MySQL nasceu na Suécia e foi criado por David Axmark, Allan Larsson e o finlandês Michael “Monty” Widenius que trabalham em conjunto desde 1980. No desenvolvimento do MySQL, trabalhavam mais de quatrocentas pessoas que atuavam em localidades diversas ao redor do mundo. Esse número não considera os mais de mil responsáveis pelos testes do software e análise da sua integração com as várias versões das aplicações e produtos ou que criam conteúdo sobre ele.

O banco de dados MySQL é um sistema de gerenciamento SGBD que usa a Linguagem de Consulta Estruturada como interface. A flexibilidade, versatilidade e a facilidade encontrada em seu uso fizeram do MySQL um dos bancos de dados preferidos. Usado por algo em torno de 10 milhões de instalações no mundo todo. O MySQL é usado pela NASA, Friendster, Bradesco, Nokia, HP, Sony, Lufthansa e outras.

Facilmente integrado com linguagens de programação, como o “PHP”, no requisito hospedagem de site é o primeiro da lista dos mais procurados. O banco de dados MySQL pode suportar Unicode, Full Text Indexes, replicação, Hot Backup, GIS, OLAP, etc.

É aceito por quase todas as plataformas usadas atualmente, compatível com drivers ODBC, JDBC e .NET e tem módulos de interface para as linguagens de programação como o Java, C/C++, Delphi, Phyton, Perl, ASP, PHP e Ruby. Isso assegura uma fantástica performance e garante grande estabilidade na operação dos servidores. Além de simples de

usar, é gratuito e tem licenciamento do tipo GPL. Permite também utilizar MyISAM, InnoDB, Falcon, BDB, Archive Federated, CSV, Solif, etc. Suportando controle transacional, triggers, Stored Procedures e Functions, possui replicação configurada de forma descomplicada, extremamente facilitada e interfaces gráficas MySQL Toolkit. Isso garante alta usabilidade.

Possui extrema robustez e realiza o trabalho de forma perfeita em qualquer ambientação, MySQL roda em mais de 20 plataformas incluindo (Windows, linux, solaris, Mac OS, Sun OS, SGI, IBM, AIX, etc.) tornando-o extremamente flexível. Essas características levaram o MySQL a se tornar muito popular e ser uma grande sacada para aqueles que precisam gerenciar bancos de dados de qualquer tamanho e de qualquer volume de requisições. Desde as aplicações simples e destinadas apenas a um único usuário, até as mais complexas rotinas destinadas às aplicações das grandes corporações multinacionais.

Não apenas multinacionais, mas empresas como um todo buscam o MySQL quando se fala em economia, pois com ele economiza-se em time e dinheiro.

GPS

Antes de falarmos sobre GPS devemos saber o seu surgimento e qual motivo que o GPS chegou a atualidade.
Antes de 1957 na URSS (Rússia) foram feito alguns testes sobre geodésica espacial e na URSS mesmo em outubro de 1957 foi feito lançamento do primeiro artificial da Terra por satélite, o Sputnik I.
Mais tarde foi concluído que existia determinadas mudanças por sinais transmitido pelo satélite Sputnik(usado para transmissão Droppler), a partir de estações conhecidas como posições. Foi possivel estabelecer orbita do satélite em volta da Terra e isso permitiria obter posições de um receptor em qualquer lugar.

Durante a década seguinte as investigações se concentraram em desenvolver e aperfeiçoar os métodos básicos de observações de satélite sistemas de cálculo de órbitas e implementar a fim de posicionar e determinar o campo de gravidade da Terra, que ajudou a criar o primeiro sistema de posicionamento geodésico.

Esse uso era exclusivo de militares (observações Droppler) e torno-se operacional em 1964 e 3 anos depois foi usado para trabalhos geodesicos como medições de grandes redes geodésicas, a determinação parâmetros entre sistemas geodésicos, e outras aplicações científicas e tecnologia da época.Esse sistema funcionou até 1996 porque existiam alguns tipos de falhas a serem corrigidos.
Agora falaremos sobre GPS (Global Positioning System) em português Sistema Global de Posicionamento .

Ainda em uso militar foram feitos os programa de implementação NAVSTAR, GPS (navegação
Temporização do sistema And Ranging Global Positioning System) , a implementação foi iniciada em dezembro de 1973 e alguns anos a seguir em 1978 (5 anos depois) foram lançados os primeiros satélite em uma série de quatro.

O GPS pode se dizer que é um sistema que visa a determinação de as coordenadas espaciais de pontos em um sistema de referência o mundo. Os pontos são localizados em qualquer parte do mundo sem exceções, pode permanecer estático ou dinâmico e observações podem ser feitos a qualquer hora do dia, momentos, lugares e etc.

Para obter o sistema de coordenadas com base na determinação distâncias simultânea para quatro satélites . Essas distâncias são obtidos a partir dos sinais dos satélites, que são previamentes recebidos por receptores especialmente concebidos. As coordenadas de satélite são feitas através de um sistema receptor.

No GPS existem dois requisitos que são o Direto e Implementação que fazem fixar pontos exatos no terreno e determinar as suas coordenadas em um sistema de referencia fixo ou se dada as coordenadas de um ponto fixo, materializar o terreno (feito em latitude e longitude).
Sistema GPS é constituido por três fundamentos que são de Controle , Usuario e Espacial.

JUSTIFICATIVA

O projeto servirá de auxílio para desenvolvedores com interesse na criação de aplicações para dispositivos móveis, baseando-se em um estudo comparativo do desenvolvimento de software de acesso a Banco de dados remoto através de webservice nas plataformas Android e iOS.

3. PLANO DE TRABALHO

Pretende-se efetuar uma comparação entre as plataformas de desenvolvimento para aplicações utilizando os sistemas operacionais para alguns dos dispositivos moveis mais populares do mercado: Android e iOS.

3.1. Metodologia

Será realizado um estudo bibliográfico, prático e comparativo sobre acesso a *webservices* e banco de dados via dispositivos móveis, resultando em um aplicativo que utiliza estes recursos para seu funcionamento.

3.2. Cronograma

ATIVIDADE / MÊS		MAR			ABR				MAIO					JUN				JUL				AGO					SET		
Pesquisa Tema	P																											
	R																											
Desenvolvimento do Projeto	P																											
	R																											
Planejamento	P																											
	R																											
Codificação	P																											
	R																											
Teste	P																											
	R																											
Documentação da codificação	P																											
	R																											
Implantação	P																											
	R																											
Suporte	P																											
	R																											

P - Previsto; R – Realizado

4. VIABILIDADE

O desenvolvimento deste software é viável, principalmente financeiramente. O custo do projeto é baixo, já que as ferramentas utilizadas são de uso gratuitos ou possuem licenças *express* (licença gratuita para estudante), exceto o SDKs do iOS. Custos extras serão a hospedagem do banco de dados e *webservice*, registro de domínio e computadores.

4.1. **Equipamentos, Materiais e Ferramentas**

Computadores PC e Mac, celulares Android e iPhone, *software* de desenvolvimento xCode e Eclipse, hospedagem e registro do domínio.

4.2. Orçamento

Hospedagem no valor de R$ 10,00 (mês) , registro de domínio por R$ 30,00 (anual), Os gasto serão rateados entre os integrantes do grupo.

Serviço de Membro para utilização do SDK, incluindo IDE xCode para o iOs no valor de USD 99.00 (anual) e no valor de USD 25.00 o registro.

Computadores e celulares serão fornecidos pelos integrantes do grupo, sem a necessidade de comprá-los.

O *softwares* utilizados para o estudo e desenvolvimento de partes do aplicativo, como Eclipse, não terá custo, pois, serão usadas versões *express*, *freeware* ou *open source.*

Comparação

Ambiente de desenvolvimento IDE e SDK

Android

O desenvolvimento no Android é feito através do modelo de programação orientada a objetos, modelo ao qual é muito utilizado pelos desenvolvedores das principais linguagens de programação utilizadas atualmente.
Para iniciar o desenvolvimento de aplicações para Android é necessários alguns itens que darão suporte ao desenvolvedor, pois, são eles de suma importância para que seja feita uma aplicação.
Os aplicativos desenvolvidos para o Android tem cada um seu processo separado dos demais, pois, cada aplicativo tem sua userid que é gerada automaticamente pelo Android durante a implantação, por esse motivo a aplicação fica separa da execução de outros aplicativos, logo, uma aplicação que tenha um mal funcionamento não interfere nas demais aplicações que estão sendo executada.

IDE (*Integrated Development Environment*):
É o ambiente onde o desenvolvedor desenvolve toda rotina de codificação da aplicação a IDE faz a compilação do código e executa de forma com que se obtém o resultado final, a proposta feita pelo pela Google para o desenvolvimento de aplicações Android é o uso da IDE Eclipse.
O Eclipse é uma IDE bastante utilizada por desenvolvedores de diversas linguagens, logo, ele também é utilizado pelo Android por ser uma IDE já bastante madura, com suporte as principais funções de refatoração e organização de código.
O Eclipse é uma IDE *open source,* ou seja, ele é gratuito, além disso é um ambiente de desenvolvimento multiplataforma funciona nos principais sistema operacionais e é utilizado para desenvolvimento em várias linguagens de programação além do Android.
Para Android é possível desenvolver o layout do aplicativo em dois modos no modo texto e modo design ambos funcionam muito bem, porem, há quem diga que o desenvolvimento em modo texto é mais eficiente, isso é uma preferencia do desenvolvedor. Para o modo design a Google disponibiliza um *plugin* para que tudo possa ser efetuada simplesmente clicando e arrastando os componentes para tela, e graças a esse *plugin* tudo é feito dentro do eclipse.

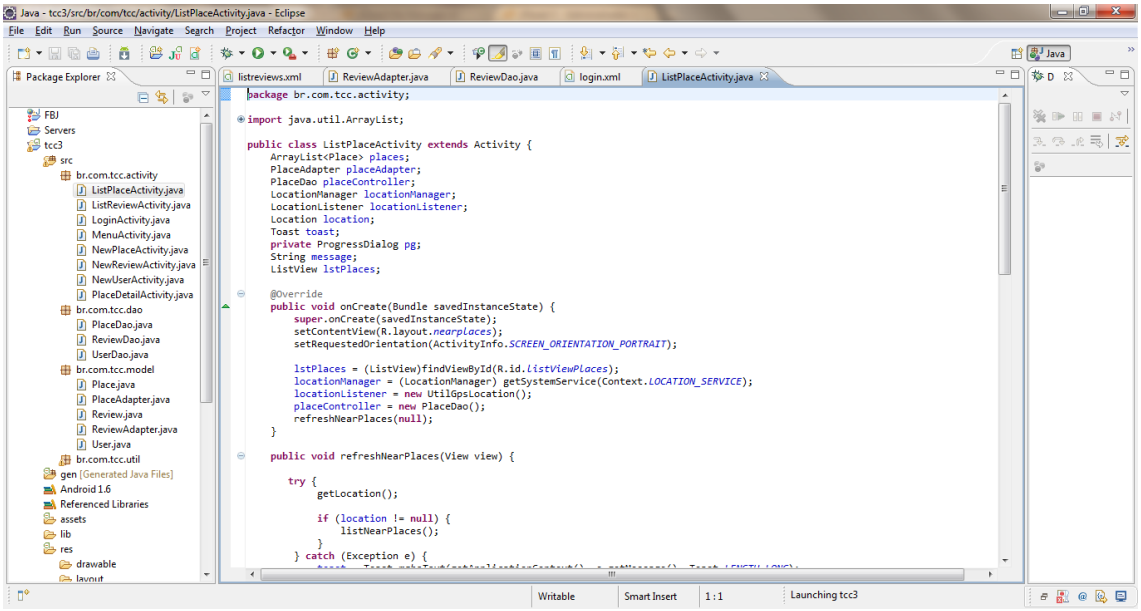


Figura 4 - Exemplo de desenvolvimento em modo texto.

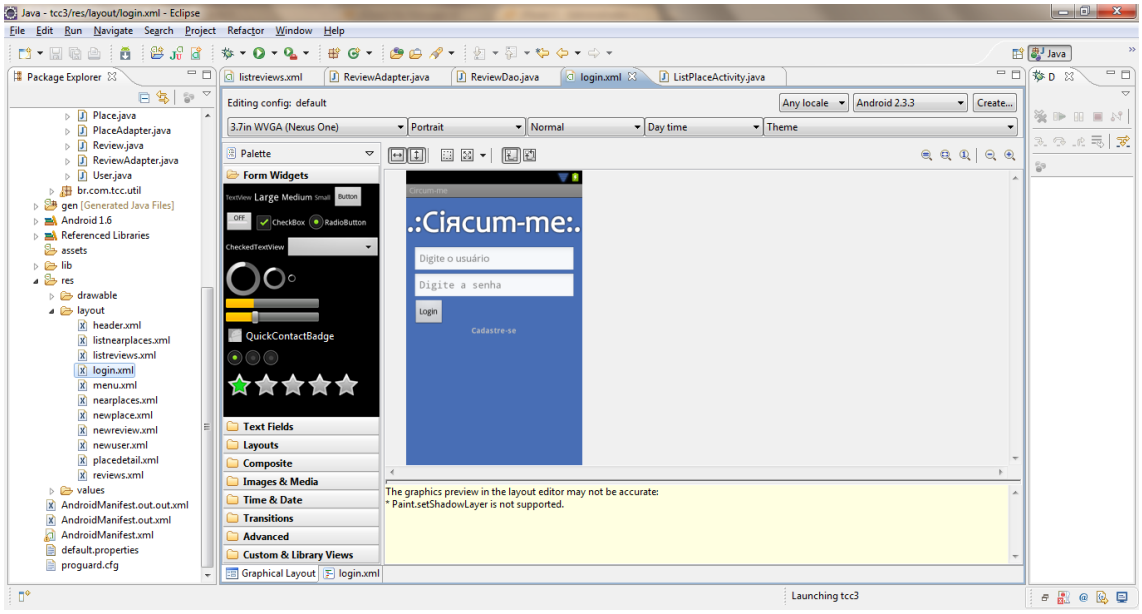


Figura 5 - Exemplo de desenvolvimento em modo layout

Maquina Virtual

O desenvolvimento de aplicações para o Android requer o uso da maquina virtual Dalvik desenvolvida pela Google utilizando a linguagem JAVA da Oracle. A Dalvik é baseada em registradores, na qual em ambientes com menor capacidade ou um tanto restritos o desenvolvimento baseado em registradores tem uma melhor performance.

Processo Oracle contra Google

Diante desse assunto está tramitando na justiça um processo da Oracle contra a Google a qual esta sendo acusada de violar patentes da tecnologia Java da Sun que foi adquirida pela Oracle no início de 2010. Segundo a Oracle a Google utilizou os recursos Java para o desenvolvimento da Dalvik e contratou engenheiros da Sun, já a Google diz que o código da Dalvik baseada em Java foi feita do zero ou seja todo desenvolvimento foi realizado por eles. Porém, até então não se houve acordos entre ambas as partes.

O Desenvolvimento de aplicações para o Android não pode ser realizado diretamente pelo Eclipse, pois, o código gerado pelo Eclipse, precisa ser analisado e interpretado por uma ferramenta desenvolvida pela Google chamada DX, a qual converte os arquivos gerados de classes Java para arquivos "DEX" que são executáveis da maquina virtual Dalvik. Toda aplicação feita para o Android é embalada em uma extensão .apk a qual é gerada pelo AAPT (*Android Asset Packaging Tool*). Para que o desenvolvimento se torne mais amigável e simples. A Google também disponibiliza o ADT (Android Development Tools) para o Eclipse. A atribuição do ADT para o Eclipse é feita a partir do momento em que o Eclipse gera o arquivo .dex e o ADT converte o .dex em arquivos .apk que é o tipo de arquivos lidos pela plataforma Android. Além da função citada acima o ADT é o *plug-in* que faz a integração entre o Eclipse e o SDK do Android, função essa imprescindível para o desenvolvimento no Eclipse.

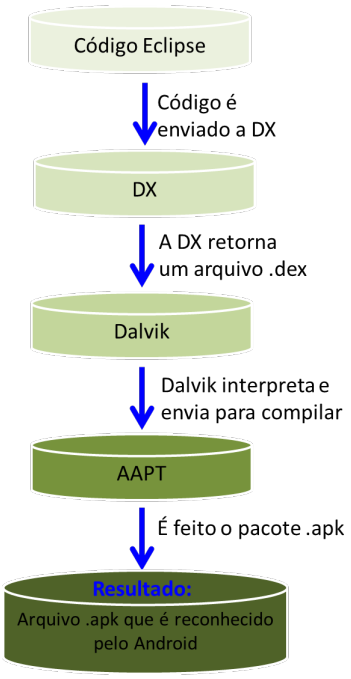


Figura 6 - Passos do desenvolvimento

SDK (*Software Development Kit*)

O SDK Android é um pacote de ferramentas desenvolvido pela Google a fim de disponibilizar tudo o que é necessário para desenvolvimento em Android, essa ferramenta é essencialmente funcional e torna o desenvolvimento mais simples e interativo, proporcionando um ambiente de desenvolvimento mais favorável e com uma interface bem amigável ao desenvolvedor, porém, há também a possibilidade de criação das interfaces via texto, ou seja, pode-se tambem programar utilizando códigos xml, onde é feita as requisições de recursos utilizados na aplicação.

É nesse *kit* de ferramentas que esta embutido recursos como caixas de texto, *radio buttons*, *combo box* entre outros componentes de tela.

Outro aspecto muito importante do SDK esta na possibilidade de facilitar a integração do aplicativo á recursos de hardware e software do dispositivo como conexão de dados, GPS, *bluetooth*, contatos, entre outros recursos disponíveis, para isso é necessário solicitar permissão através do arquivo AndroidManifest.xml.

Exemplo de chamada de recursos:

Utilização de rede de dados:

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

Utilização de GPS:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
```

Simulação

No SDK do Android há a possibilidade de criar diversas maquinas virtuais para que possa ser visto o resultado do trabalho podendo por ela executar o aplicativo em desenvolvimento como se fosse um dispositivo móvel real, existe também a possibilidade de conectar o dispositivo no computador e efetuar todos os testes diretamente nele, sem precisar passar por nenhuma maquina virtual (além da Dalvik). A maquina virtual e as ferramentas utilizadas no desenvolvimento de aplicações Android são bem flexíveis, pois, elas possuem todas as versões de diversos tipos de dispositivos, logo, com o mesmo SDK é possível desenvolver para dispositivos como celulares e *tablets*.

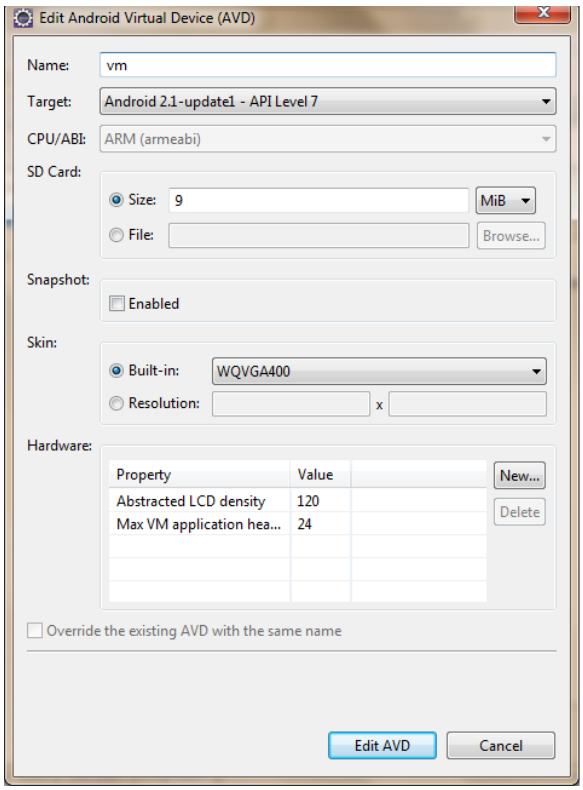


Figura 7 - Tela de Criação do simulador

Figura 8 - Simulador Android no PC

Comparação

Ambiente de desenvolvimento IDE e SDK

iOS

Para a criação de nosso aplicativo no *IOS*, utilizamos o ambiente de desenvolvimento *Xcode*, que é um conjunto de ferramentas disponibilizado pela *Apple*. Através dele é possível a criação de telas utilizando recursos gráficos, digitação do código fonte, depuração e execução da aplicação.

Na figura 1 é possível visualizar o ambiente *Xcode*. Do lado esquerdo estão os navegadores, e em suas estruturas os arquivos de desenvolvimento que compõe o projeto e os recursos que são utilizados.
Extensão dos arquivos do projeto:

- .xib: Cada arquivo é referente a uma tela (sua representação gráfica), também chamada de view;
- .h: Arquivo com código fonte. Cabeçalho da classe que contém declarações e definições de variáveis e campos;
- .m: Arquivo com código fonte. É o complemento da classe, contém as instruções e execuções dos métodos;

Para a elaboração da interface gráfica, são disponíveis recursos que facilitam a interação. Basta selecionar o controle e arrastá-lo para a tela, posicionando de modo livre em qualquer área da janela.
Caso o objeto da tela receba, exiba uma informação ou execute algum procedimento, é obrigatório para o funcionamento que este objeto seja vinculado à programação. A vantagem é que basta selecioná-lo e arrastá-lo para seu código fonte, representado pelo *Placeholder* '*File'sOwner'*(figura 2), que serão exibidos os atributos do código para o relacionamento.

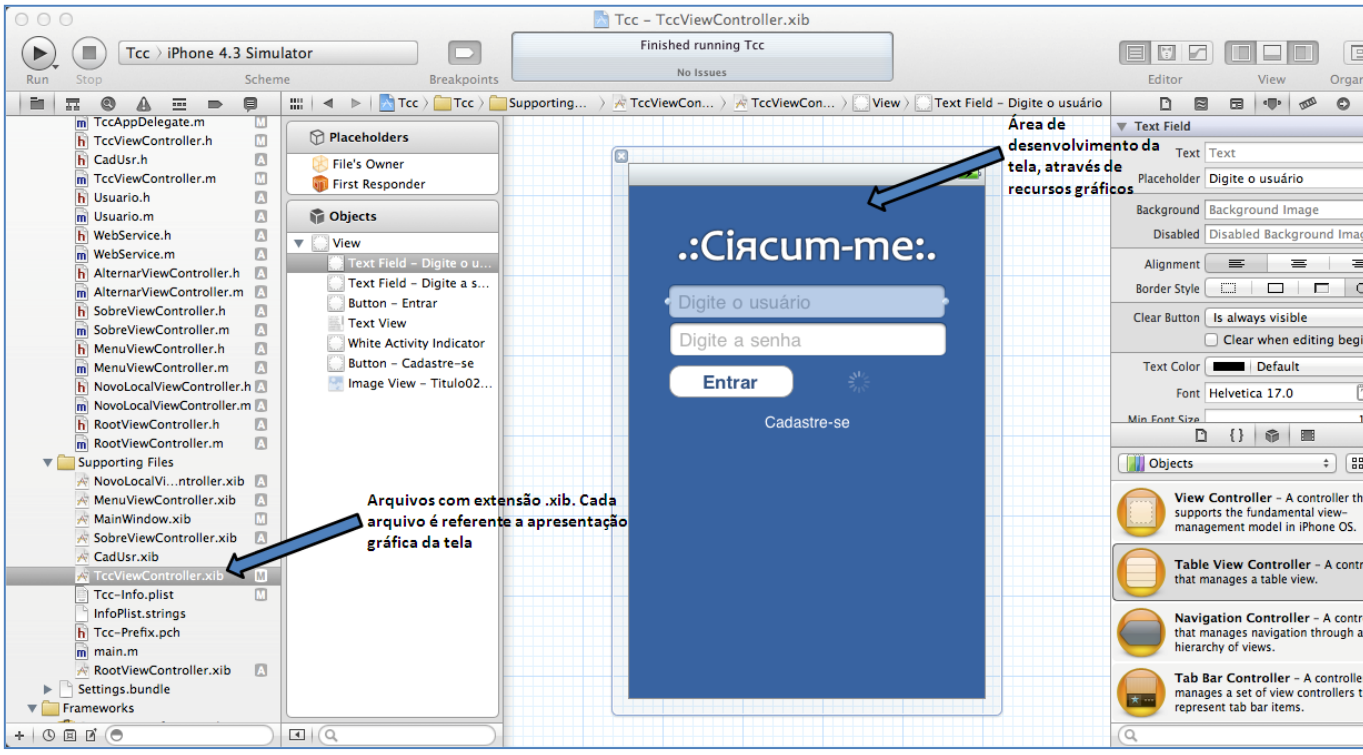


Figura 9: Ambiente de desenvolvimento xCode

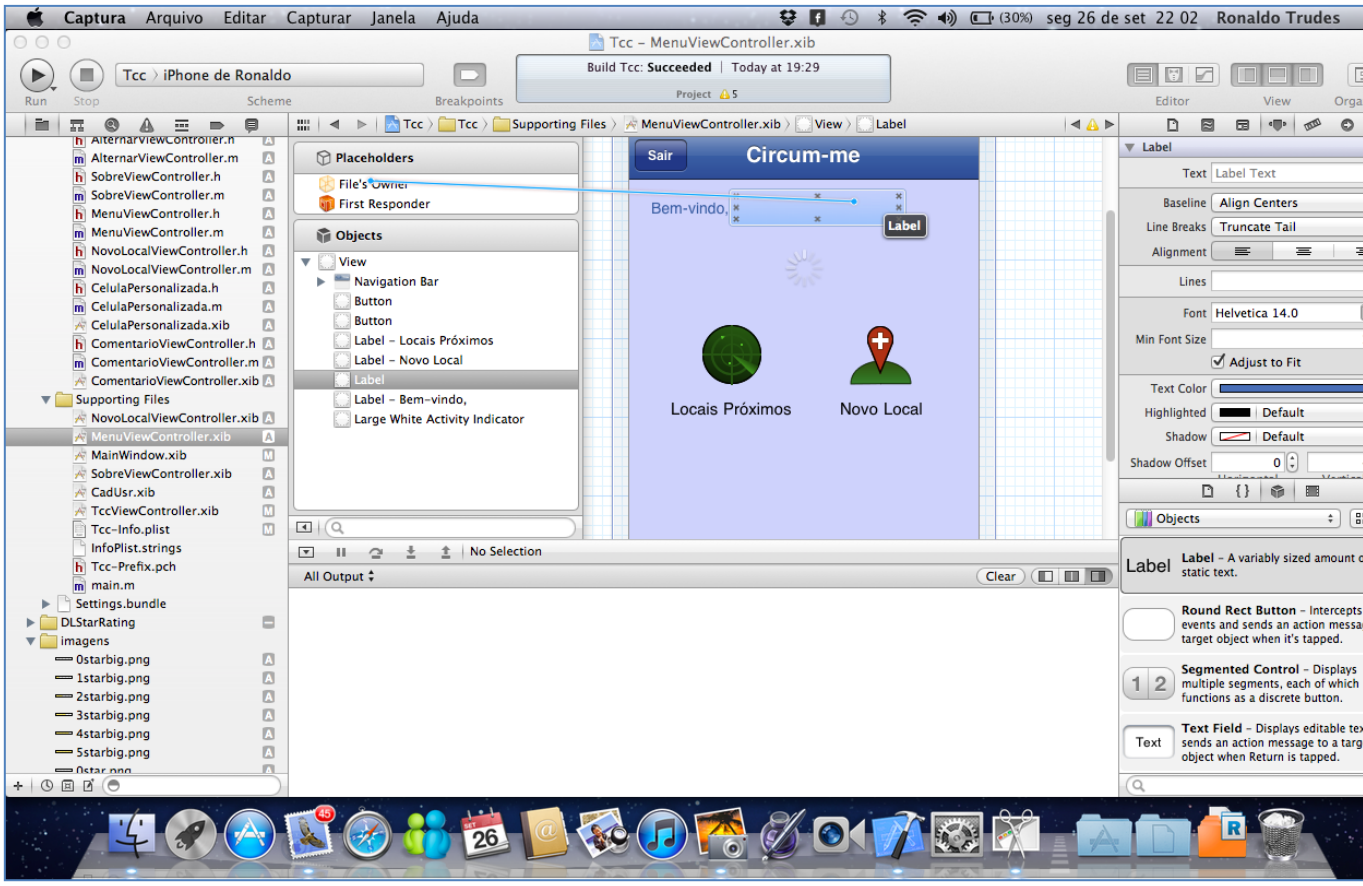


Figura 10: Utilização do recurso de vincular o objeto gráfico ao código fonte

Ao selecionar o arquivo com extensão “.m” ou “.h” é exibido seu código fonte, conforme a figura abaixo.

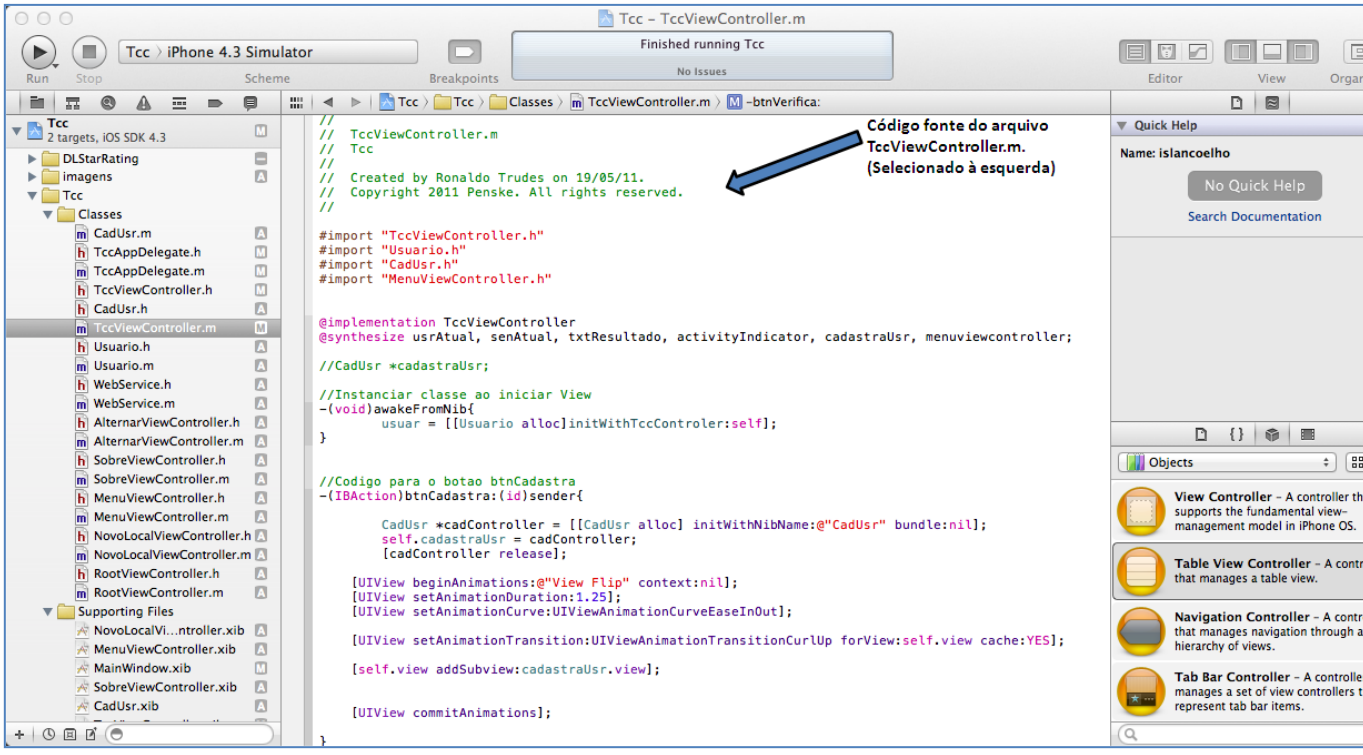


Figura 11: Exemplo de um dos códigos fonte da aplicação

Durante o desenvolvimento é possível visualizar o resultado da aplicação através do simulador. Existe a opção de rodar no aparelho conectado ao computador, ou no simulador do próprio *Xcode*. Basta selecionar a opção conforme indicado na figura 4. Isto facilita e agiliza o processo de testes do sistema, pois em paralelo com o desenvolvimento torna-se prático a validação da rotina. O IPhone Simulator faz a simulação do aparelho de acordo com a versão do software selecionada, permitindo utilizar grande parte dos recursos existentes no celular.

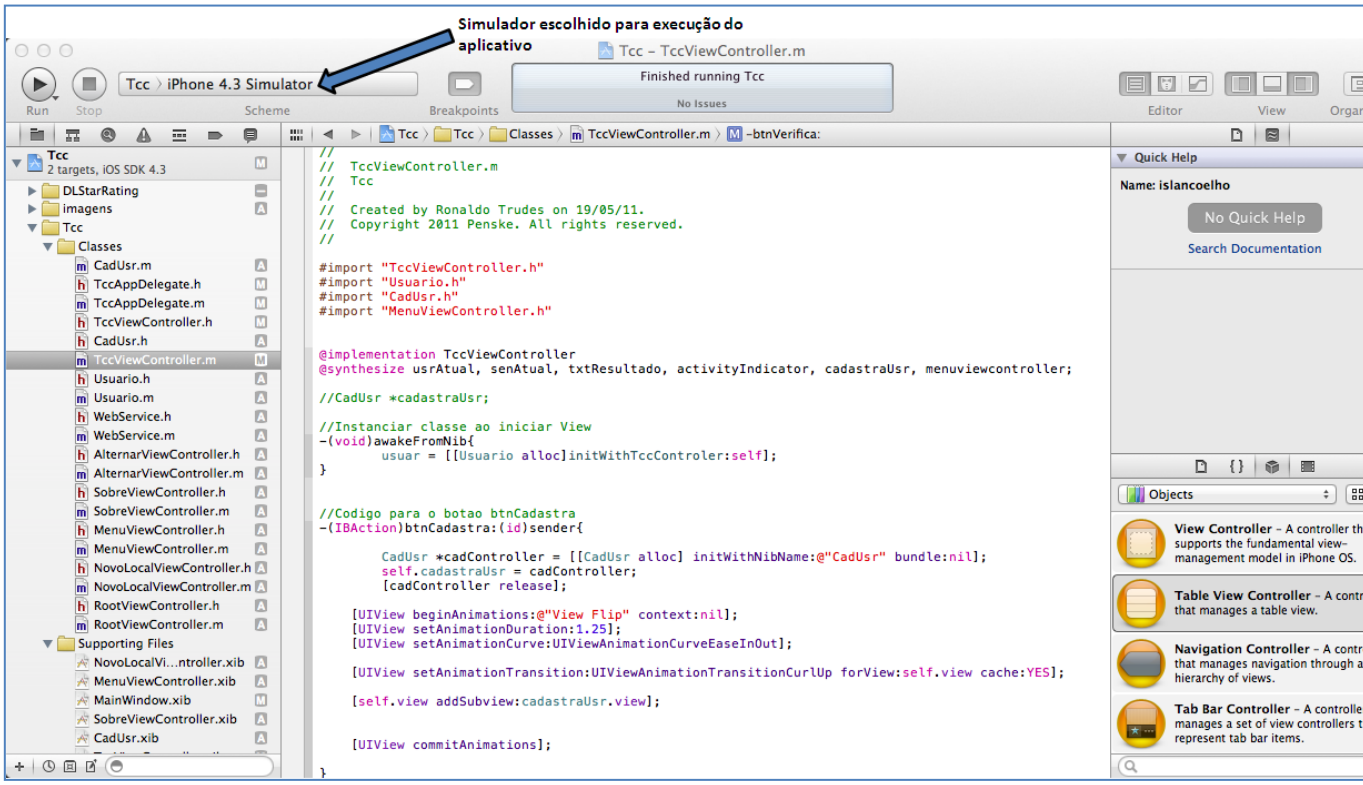


Figura 12: Seletor de execução

Figura 13: Simulador em execução

Este *IDE* da *Apple* possui uma funcionalidade que auxilia os desenvolvedores, pois identifica alguns erros de sintaxe, semântica e erros de digitação. Antes da compilação já exibe o alerta de erro na instrução, e também uma solução para o problema. Esta lista com a sugestão de soluções é chamada de *Fix-it*.

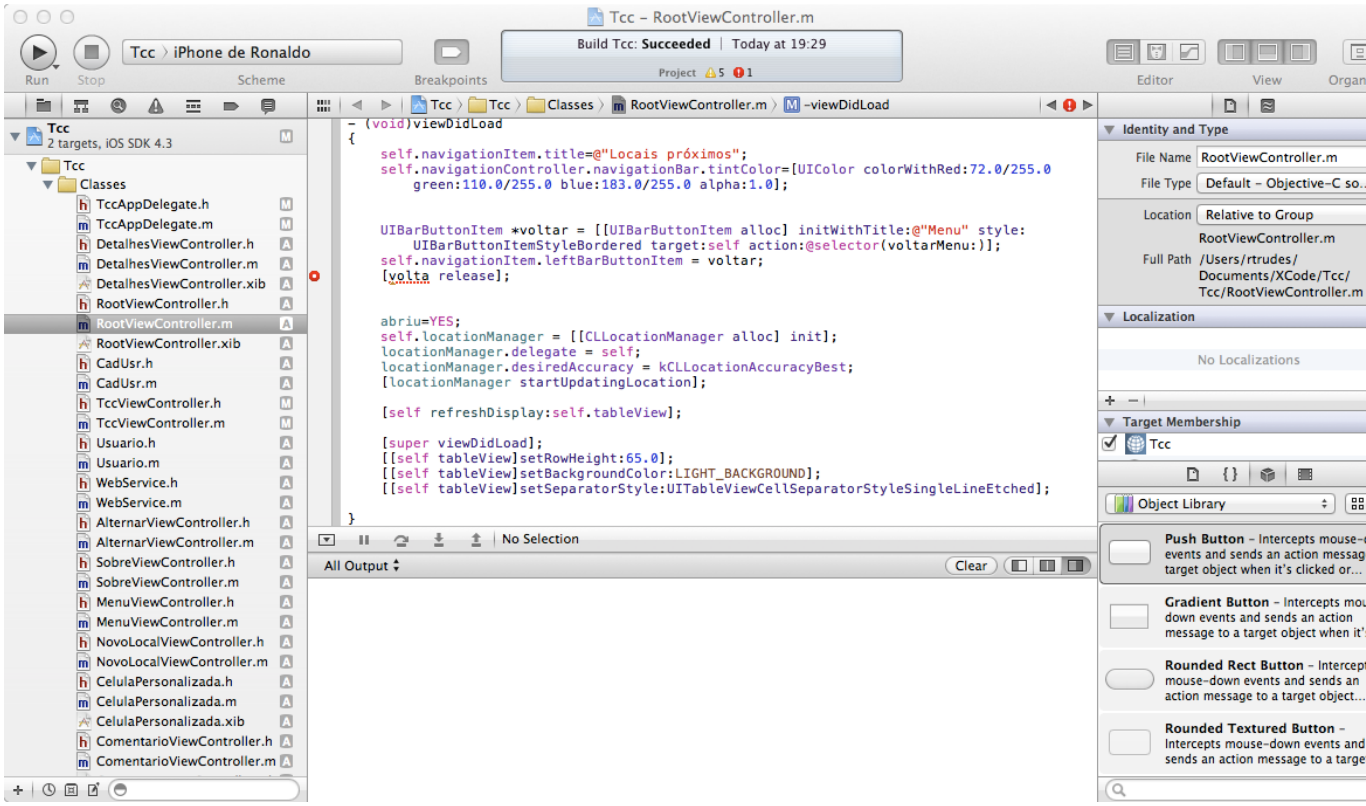


Figura 14: Erro no código fonte sendo apontado pelo xCode

Para a comunicação entre a aplicação e o hardware, não é possível um acesso direto, é necessário a implementação de *frameworks*, como o *UIKit, CoreGraphics* e o *CoreLocation*, que estão em camadas mais baixas, por isso podem utilizar componentes do hardware.
Os frameworks permitem a utilização de gráficos 3D, áudio, controle de câmera e GPS, bastando referencia-los no início do código.

Comparação

GUIs – Bibliotecas e Interfaces Gráficas

Atualmente se tratando de aplicações para dispositivos móveis um dos pontos mais importantes é o ambiente gráfico o qual chamamos de GUI (*Graphical User Interface),* ou seja, interface gráfica do usuario. As GUIs são extremamente importantes, pois, é atraves dela que o aplicativo torna-se atrativo ao usuario e simples se utilizar.
É Atraves de rucursos de GUIs que são feitas as tela, definidas as cores, papeis de parede, barra de menus personalisados entre outros itens que deixam a aparencia do aplicativo amigavel com elementos graficos e não apenas em modo texto.

Android
O Funcionamento das GUIs no Android é feito em pacotes de itens que são chamados diretamente na codificação da aplicação.

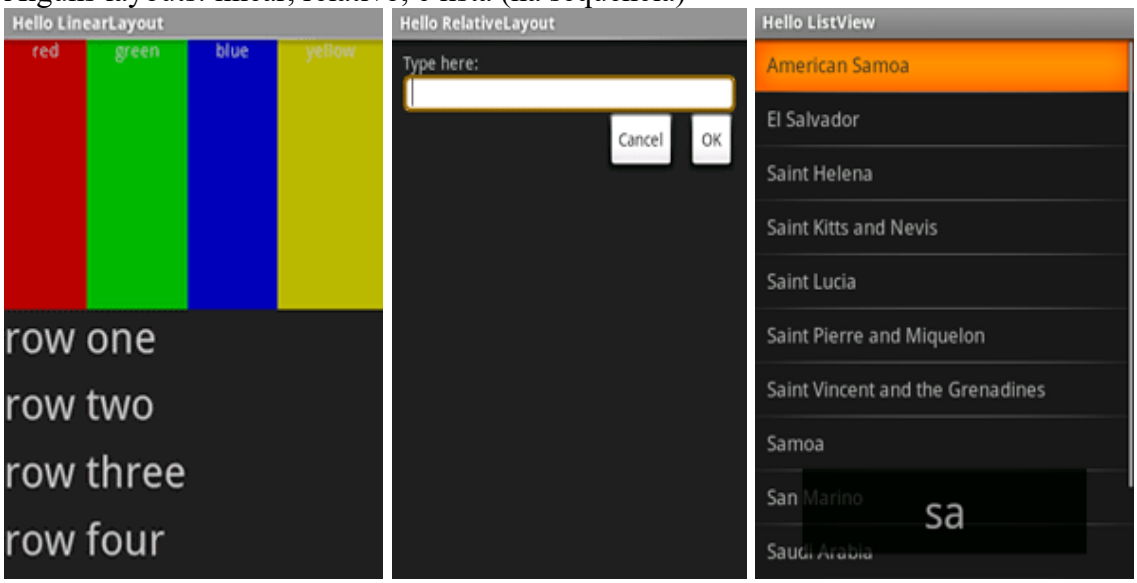
- **Icon Design**
  É esse pacote que possibilita o trabalho com os ícones que estarão sendo utilizado, a partir desse pacote é possível definir, tamanho, cor, sombreamento do ícone.
- **Widget Design**
  Esse pacote possibilita a interface de inclusão de links para arquivos gráficos e modelos que vão tornar a vida o designer é mais fácil.
- **Activity and Task Design**
  Esse pacote permite o gerenciamento das aplicações diretamente quando usado o recurso de multitarefa, é ele que cobre toda o gerenciamento multitarefas.
- **Menu Design**
  Esse pacote possibilita as aplicações para Android fazer uso de menus de opções e menus de contexto que permitem aos usuários realizar operações e navegar para outras partes do seu aplicativo ou para outras aplicações. Essas orientações descrevem a diferença entre opções e menus de contexto, como organizar os itens do menu, quando colocar os comandos na tela, e outros detalhes sobre o projeto do menu.

Gerenciadores de Layout

Para facilitar a organização da tela são disponibilizados tipos de gerenciadores de layout diferentes, assim como ocorre no framework *Swing* muito conhecido pelos programadores java, alguns dos gerenciadores principais são:

- *Linear Layout* – Nesse caso fica basicamente um componente embaixo do outro, de forma como o próprio nome diz, linear, sem a possibilidade de posiciona-los de forma livre.
- *Relative Layout* – Esse é o gerenciador que melhor permite o posicionamento manual dos componentes de tela, pois é possível posiciona-los praticamente onde quiser.
- *ListView* – Basicamente utilizado para mostrar listas de seleção.

Alguns layouts: linear, relativo, e lista (na sequencia)



Como mencionado anteriormente, é possível criar essas telas de duas formas, a primeira utilizando a própria IDE Eclipse com o componente ADT, apenas clicando e arrastando os componentes para a tela, e a outra que é codificando arquivos XML manualmente, abaixo demonstramos como exemplo a codificação de um *Linear Layout*.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

  <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1">
      <TextView
          android:text="red"
          android:gravity="center_horizontal"
          android:background="#aa0000"
          android:layout_width="wrap_content"
          android:layout_height="fill_parent"
          android:layout_weight="1"/>
      <TextView
          android:text="green"
          android:gravity="center_horizontal"
          android:background="#00aa00"
```

```
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <TextView
            android:text="blue"
            android:gravity="center_horizontal"
            android:background="#0000aa"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <TextView
            android:text="yellow"
            android:gravity="center_horizontal"
            android:background="#aaaa00"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
    </LinearLayout>

    <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1">
      <TextView
          android:text="row one"
          android:textSize="15pt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
      <TextView
          android:text="row two"
          android:textSize="15pt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
      <TextView
          android:text="row three"
          android:textSize="15pt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
      <TextView
          android:text="row four"
          android:textSize="15pt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
    </LinearLayout>

</LinearLayout>
```

Nesse exemplo podemos observar que cada *tag* que criamos é um novo componente, e essas *tags* possuem atributos onde definimos o posicionamento e a aparência de cada componente.

Comparação

GUIs – Bibliotecas e Interfaces Gráficas

iOS

GUI (*Graphical user interface*) é a interface gráfica que está entre o hardware e o usuário. Tem o objetivo de permitir a criação de um aplicativo mais simples e associativo para a utilização. Pois permite o uso de janelas, botões, caixas de textos, menus, etc. e a interação com estes objetos pode ser através de mouse, teclado e toque na tela.
Para o IOS esta interface é realizada no ambiente de desenvolvimento XCode.

O IOS é dividido em quatro camadas, na camada CocoaTouch existem vários frameworks para a construção da aplicação, fornecem os recursos necessários para que a aplicação possa gerenciar os dispositivos de hardware. Para implementação de aplicações com recursos gráficos (como animação, áudio, vídeo), recursos de localização, conexão e acesso a banco de dados.

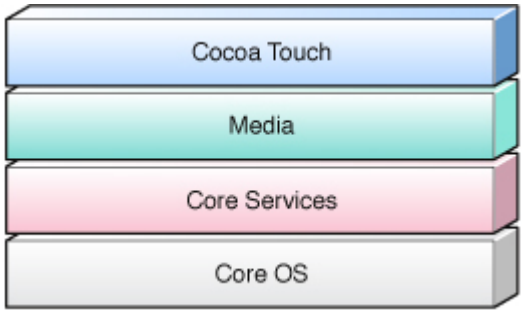


Figura 15: Camadas do IOS

http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_iPhone_OS_Overview/_index.html#//apple_ref/doc/uid/TP40007592

Framework é um diretório que contém uma biblioteca compartilhada (e os recursos associados da biblioteca). Para se utilizar destes recursos que estão na biblioteca, é necessário referenciá-lo no projeto, através de uma API que é inserida no código do framework.
Com a utilização de bibliotecas compartilhadas dinamicamente, vários aplicativos podem chamá-las em concorrência, pois a aplicação carrega o código delas e os recursos na memória, então cada execução tem seu acesso de forma exclusiva.
Os frameworks são importados para o projeto, e estão disponíveis na área de navegadores do Xcode. Os principais frameworks são adicionados durante a criação do projeto, os demais precisam ser copiados para o projeto, de acordo com a necessidade da aplicação.

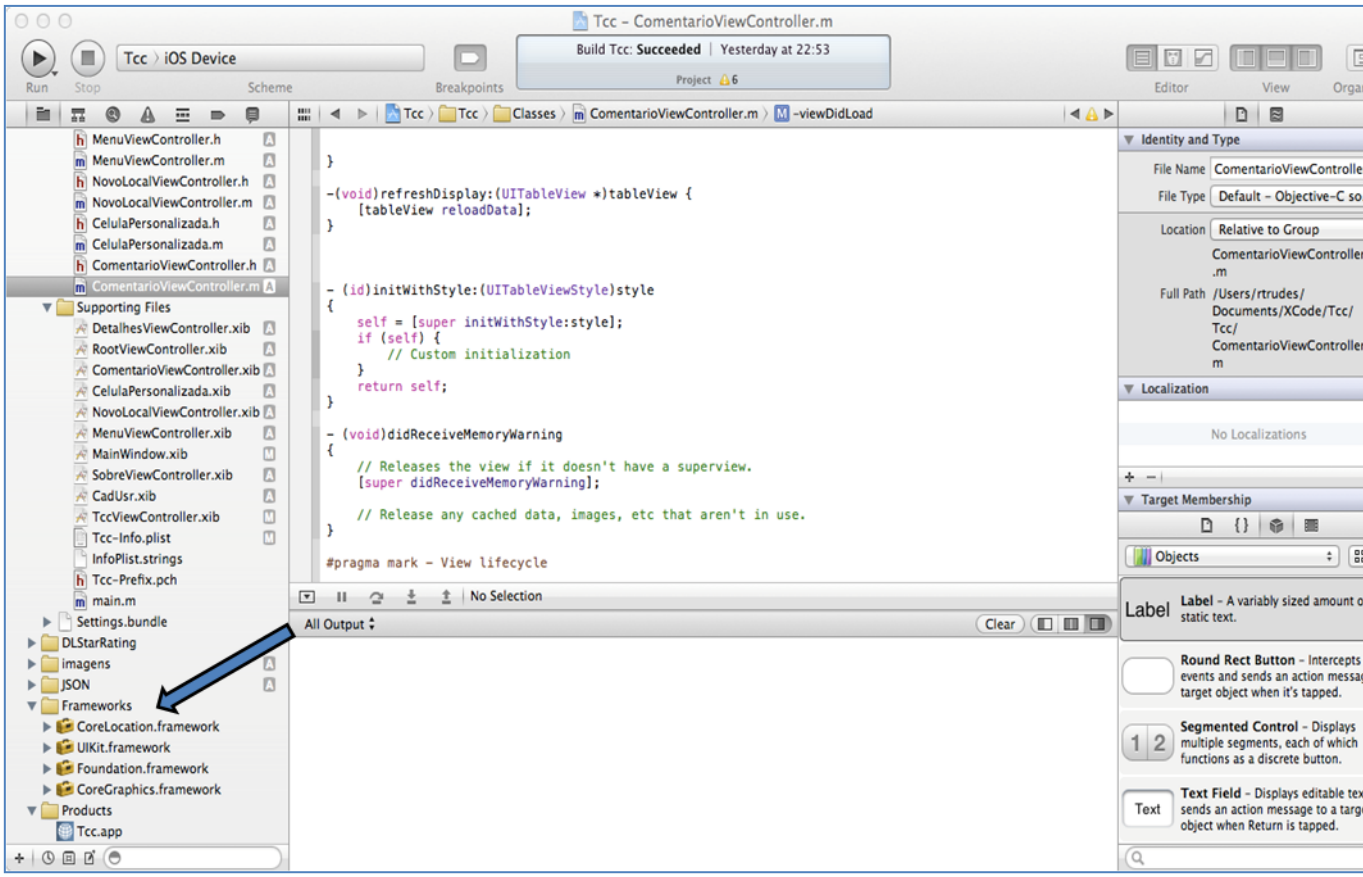


Figura 16: Frameworks utilizados no projeto

O Core Graphics permite a criação de objetos com característica 2D, 3D, gerenciamento de desenhose animação (criação e modificação), com efeitos avançados. E em relação a outras classes, possui algumas vantagens como facilitar a criação de interface sem a utilização de APIs.

O Core Location possibilita ter a posição exata do aparelho (sua localização geográfica).

```
#import <CoreLocation/CoreLocation.h>

@interface NovoLocalViewController : UIViewController <CLLocationManagerDelegate, NSXMLParserDelegate,
UITextFieldDelegate, DLStarRatingDelegate>{
   IBOutlet UITextField *novolocal;
   IBOutlet UITextField *novocoment;
   IBOutlet UILabel *resultado;
   IBOutlet UIActivityIndicatorView *indicadorAtiv;
   IBOutlet UILabel *stars;

   NSMutableData *webData;
   NSMutableString *results;
   NSURLConnection *conn;
   NSXMLParser *xmlParser;

   BOOL elementFound;
   BOOL insereReview;
   CLLocationManager *locationManager;
   CLLocation *startingPoint;
   IBOutlet UILabel *strlatitude;
   IBOutlet UILabel *strlongitude;
}
@property (nonatomic, retain) UITextField *novolocal;
@property (nonatomic, retain) UITextField *novocoment;
@property (nonatomic, retain) UILabel *resultado;
@property (nonatomic, retain) UIActivityIndicatorView *indicadorAtiv;
@property (nonatomic, retain) UILabel *stars;
@property (nonatomic, retain) CLLocationManager *locationManager;
@property (nonatomic, retain) CLLocation *startingPoint;
@property (nonatomic, retain) UILabel *strLatitude;
@property (nonatomic, retain) UILabel *strLongitude;

-(IBAction)btnSalvar:(id)sender;
-(IBAction)btnCancelar:(id)sender;
-(void) InserirLocal:(NSString *)local InserirLat:(NSString *)latitude InserirLong:(NSString *)longitude;
-(void)ReviewIdUsr:(NSString *)idUsr ReviewPlace:(NSString *)idLocal ReviewStar:(NSString *)estrela
ReviewComm:(NSString *)coment;
-(IBAction)BackGroundTouch:(id)sender;
@end
```

Figura 17: Utilização do Core Location

O Core Foundation apresenta paradigmas, possibilita otimização, auxilia na organização com a criação de estruturas hierárquicas e fornece funcionalidades não abrangidas pelas classes baseadas em Objective-C.

O UIKit é uma biblioteca essencial de componentes gráficos. Possibilita utilização de algumas ferramentas básicas, como por exemplo, janelas, botões e gerenciamento de tela. O UIKit fornece classes que são responsáveis pelo gerenciamento da interface do aplicativo e o sistema IOS, controlando eventos e interação com a tela.
Com o UIKit é possível a criação de janelas, manipulação do tamanho e da localização dela e dos objetos inseridos, também pelo posicionamento e criação de desenhos. Todo o código deve ser escrito utilizando os métodos desta biblioteca para executar essas funções.
O Framework do UIKit possui diversos componentes, para o nosso projeto utilizamos as classesUINavigationBar.h, UITableView.h, UIImageView.h, UITextField.h, UILabel.h, UIActivityIndicatorView.h.

NavigationBar
É uma barra de navegação que normalmente é posicionada na parte superior da tela. Contém botões como voltar e avançar.

```
self.navigationController.navigationBar.tintColor=[UIColor colorWithRed:72.0/255.0 green:110.0/255.0
blue:183.0/255.0 alpha:1.0];


UIBarButtonItem *voltar = [[UIBarButtonItem alloc] initWithTitle:@"Menu"
style:UIBarButtonItemStyleBordered target:self action:@selector(voltarMenu:)];
self.navigationItem.leftBarButtonItem = voltar;
[voltar release];

[super viewDidLoad];
[[self tableView]setRowHeight:65.0];
[[self tableView]setBackgroundColor:LIGHT_BACKGROUND];
[[self tableView]setSeparatorStyle:UITableViewCellSeparatorStyleSingleLineEtched];
indicadorAtividade.center=CGPointMake(160, 240);
```

Figura 18: Trecho do código utilizando o NavigationBar

UItableView

É uma área para exibição e edição de tabelas (uma coluna e várias linhas). A exibição é de apenas uma coluna devido ao tamanho de tela do aparelho. É possível a rolagem vertical para acesso as células da tabela.

```
#pragma mark - Table view data source

// Número de seções da tabela.
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

//Número de linhas na tabela
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return [listComent count];
}

// Personalizar o que será exibido na célula.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath
{
    static NSString *CellIdentifier = @"Cell";

    CelulaPersonalizada *cell = (CelulaPersonalizada *)[tableView
dequeueReusableCellWithIdentifier:CellIdentifier];

    if (cell == nil) {

        NSArray *topLevelItems = [[NSBundle mainBundle] loadNibNamed:@"CelulaPersonalizada"
owner:nil options:nil];
        cell = [topLevelItems objectAtIndex:0];

        for(id currentObject in topLevelItems){
            if([currentObject isKindOfClass:[UITableViewCell class]]){
                cell = (CelulaPersonalizada *) currentObject;
                break;
            }
        }
```

Figura 19: Trecho do código utilizando o UITableView

UIImageView
Área para exibição de imagens, com a utilização de alguns controles é possível animá-las.

```
#import <UIKit/UIKit.h>

@interface CelulaPersonalizada : UITableViewCell{
   IBOutlet UILabel *comentario;
   IBOutlet UIImageView *imagem;
   IBOutlet UILabel *idUsuario;

}

@property (nonatomic, retain) IBOutlet UILabel *comentario;
@property (nonatomic, retain) UIImageView *imagem;
@property (nonatomic, retain) IBOutlet UILabel *idUsuario;


@end
```

Figura 20: Trecho do código utilizando o UlImageView

Textfield
É um objeto que exibe e permite edição de um texto, é usado para entrada de pequenas quantidades de textos.

```
#import <UIKit/UIKit.h>

@class CadUsr;
@class MenuViewController;

@interface TccViewController : UIViewController <NSXMLParserDelegate, UITextFieldDelegate> {
   IBOutlet UITextField *usrAtual;
   IBOutlet UITextField *senAtual;
   IBOutlet UITextView *txtResultado;
   IBOutlet UIActivityIndicatorView *activityIndicator;
   CadUsr *cadastraUsr;
   MenuViewController *menuviewcontroller;

   NSMutableData *webData;
   NSMutableString *results;
   NSURLConnection *conn;

   NSXMLParser *xmlParser;
   BOOL elementFound;
}

@property (nonatomic,retain) UITextField *usrAtual;
@property (nonatomic,retain) UITextField *senAtual;
@property (nonatomic,retain) UITextView *txtResultado;
@property (nonatomic, retain) UIActivityIndicatorView *activityIndicator;
@property (nonatomic, retain) CadUsr *cadastraUsr;
@property (nonatomic, retain) MenuViewController *menuviewcontroller;

-(IBAction)btnVerifica:(id)sender;
-(IBAction)btnCadastra:(id)sender;
-(void)connect:(NSMutableURLRequest *)theRequest;

@end
```

Figura 21: Trecho do código utilizando o UITextField

```
#pragma mark UITextField delegate methods

-(BOOL)textFieldShouldBeginEditing:(UITextField *)textField{
   //Definição do tipo de teclado para os campos de texto
   textField.keyboardType = UIKeyboardTypeAlphabet;
   return YES;
}

-(void)textFieldDidBeginEditing:(UITextField *)textField{
   if(textField==usrAtual && textField==senAtual){
      txtResultado.text=@"";
   }
}
-(BOOL)textFieldShouldReturn:(UITextField *)textField{
   //Ocultar teclado após sair da caixa de texto
   [textField resignFirstResponder];
   return YES;
}
```

Figura 22: Trecho do código executando métodos do UITextField

Label

Essa classe permite desenhar uma caixa de exibição de texto (apenas para visualização).

```
#import <UIKit/UIKit.h>

@interface CelulaPersonalizada : UITableViewCell{
   IBOutlet UILabel *comentario;
   IBOutlet UIImageView *imagem;
   IBOutlet UILabel *idUsuario;

}

@property (nonatomic, retain) IBOutlet UILabel *comentario;
@property (nonatomic, retain) UIImageView *imagem;
@property (nonatomic, retain) IBOutlet UILabel *idUsuario;


@end
```

Figura 23: Trecho do código utilizando a classe UILabel

ActivityIndicator

Essa classe cria e gerencia um indicador de progresso de determinada tarefa.

```
[activityIndicator stopAnimating];

if (xmlParser) {
   [xmlParser release];
}

xmlParser = [[NSXMLParser alloc] initWithData:webData];
[xmlParser setDelegate:self];
[xmlParser setShouldResolveExternalEntities:YES];
[xmlParser parse];

[connection release];
[webData release];
```

Figura 24: Trecho do código utilizando o ActivityIndicator

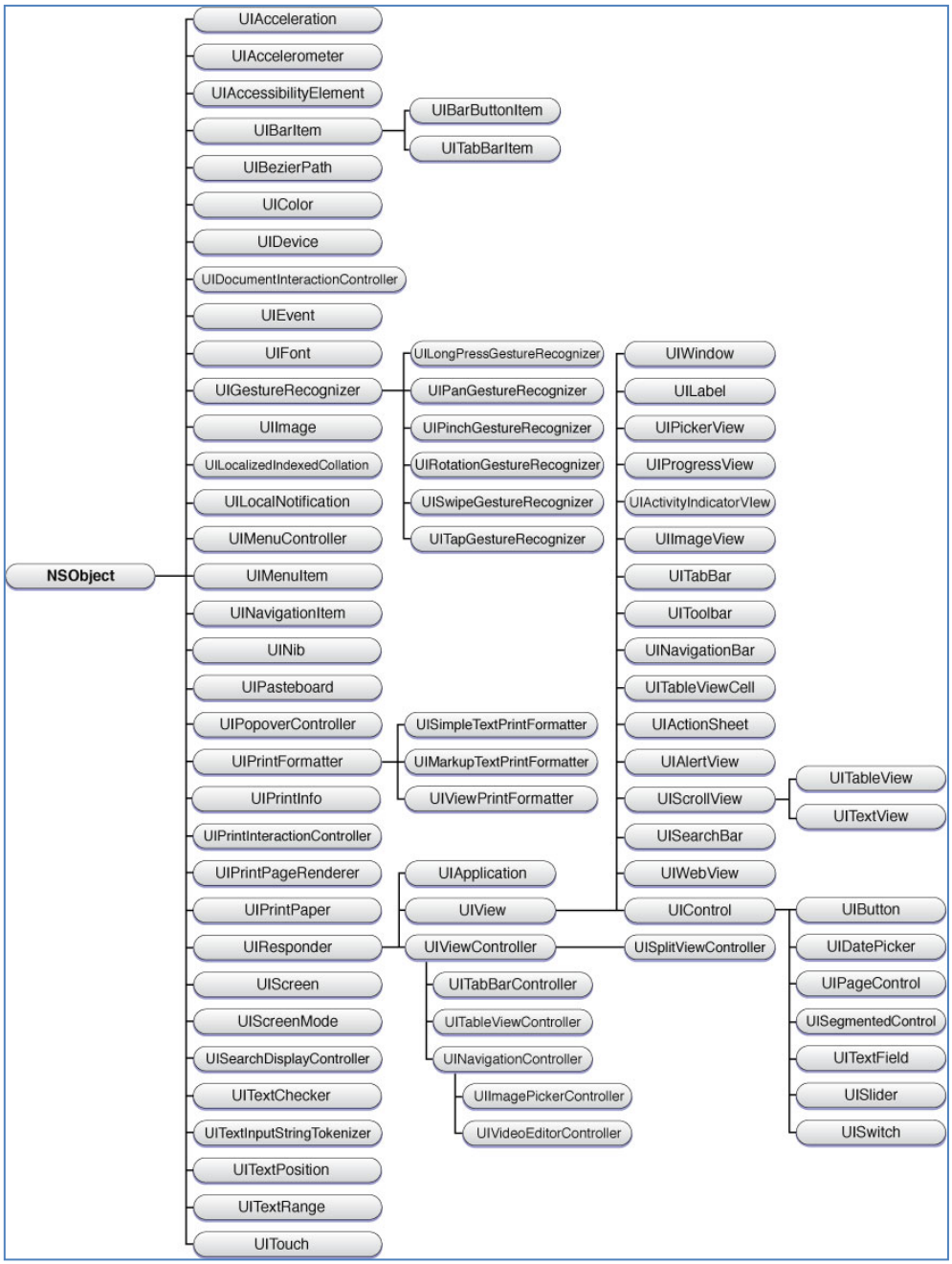


Figura 25: Componentes do UIKit.

http://developer.apple.com/library/ios/#documentation/UIKit/Reference/UIKit_Framework/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006955-CH1-SW1

Comparação

Webservice

O recurso de *WebService* é muito utilizado no desenvolvimento de aplicativos para dispositivos móveis, pois somente através dele é possível realizar consultas e manutenção a um banco de dados externo.

Android

O acesso ao Webservice é realizado através de uma biblioteca adicional chamada KSOAP2, que utiliza o XML associado ao protocolo SOAP.

O KSOAP2 permite a utilização de webservice sem a necessidade de codificação ou uso de proxies dinâmicos, levando em consideração que o a aplicação irá funcionar em um dispositivo móvel, e o proxy dinâmico é um recurso computacionalmente caro.

O código abaixo é um pré-requisito para utilizar o KSoap2, a classe HttpTransportSE é abstrata:
Quando utilizamos o KSOAP2 nós precisamos de uma classe do tipo HttpTransportSE, porém não é possível utiliza-la diretamente por ser abstrata com isso precisamos criar uma classe que herda de HttpTransportSE.

```
package br.com.tcc.util;

import java.io.IOException;

import org.ksoap2.SoapEnvelope;
import org.ksoap2.transport.HttpTransportSE;
import org.xmlpull.v1.XmlPullParserException;

public class UtilHttpTransport extends HttpTransportSE{
	public UtilHttpTransport(String s) {
		super(s);
	}

	public void call(String s, SoapEnvelope soapenvelope) throws IOException,XmlPullParserException {
		//byte bytes[] = createRequestData(soapenvelope);
		super.call(s, soapenvelope);
	}
}
```

Figura 26 - Herdadando da Classe Abstrata HttpTransport da biblioteca KSOAP2

O código abaixo foi feito para facilitar a comunicação com o webservice. Qualquer lugar do sistema que precise fazer uma chamada no webservice irá executar o método executeMethode, que por sua vez faz a requisição ao webservice:

```
import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;

public class UtilWebservices {
        private static final String URL = "http://www.islancoelho.com.br/webservice/webservice.php";
        public static final String URI = "http://www.islancoelho.com.br/webservice/";
        public Object executeMethode(SoapObject soap) throws Exception {
    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
    envelope.setOutputSoapObject(soap);
    try {
                HttpTransportSE httpTransport = new UtilHttpTransport(URL);
                httpTransport.call("", envelope);
                Object hello = envelope.getResponse();
                return hello;
                } catch (Exception e){

                        throw e;
                }
        }
}
```

Para que possamos finalizar e definitivamente chamar o webservice, abaixo segue um código que criamos para fazer a autenticação de usuário:

```
public boolean isLoginValid(String login, String password) throws Exception {
                if (login.equals("") && password.equals("")) {
                        return false;
                } else {
      //Instanciamos a classe de requisição ao webservice
                        UtilWebservices webservice = new UtilWebservices();
      /*Criamos um objeto SoapObject, com todos os dados referente ao método do
       webservice que queremos utilizar, nesse caso queremos usar o "userisValid" e passamos como
       parâmetro o login e senha informados*/
                        SoapObject soap = new SoapObject(UtilWebservices.URI, "userIsValid");
                        soap.addProperty("login", login.toUpperCase());
                        soap.addProperty("password", password.toUpperCase());
                        try {
        //Pegamos a resposta e convertemos para boolean.
                                Object result = webservice.executeMethode(soap);
                                return Boolean.valueOf(result.toString());
                        } catch (Exception e) {
                                throw e;
                        }

                }
        }
```

Comparação

Webservice

iOS

Nos iOs, para a comunicação com o web service é utilizado o protocolo SOAP, que efetua a troca de mensagens entre a aplicação e o web service, as mensagens são escritas em XML e transportadas através do protocolo HTTP.
Para acesso ao web service é utilizado um framework Core específico de baixo nível, em conjunto com o CFNetwork e CFXMLParser, que estão acessíveis no Core Services, permitindo a utilização de plugins e ferramentas.
As chamadas ao servidor são efetuadas pelo framework CoreFoundation (CF), criando um dicionário com a URL do servidor, o nome da operação e uma constante para especificação do protocolo, através do método NSMutableURLRequest. Para enviar dados da mensagem ao servidor é utilizado o método NSURLConnection, e para o tratamento da mensagem recebida o XMLParser.

```
//Definir Corpo da mensagem SOAP
   NSString *soapMsg = [NSString stringWithFormat:
                  @"<?xml version=\"1.0\" encoding=\"utf-8\"?>"
                  "<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">"
                  "<soap:Body>"
                  "<userIsValid xmlns=\"http://www.islancoelho.com.br/webservice/\">"
                  "<login>%@</login>"
                  "<password>%@</password>"
                  "</userIsValid>"
                  "</soap:Body>"
                  "</soap:Envelope>",usrAtual.text,senAtual.text];

   //Definição de cabeçalhos e valores da mensagem SOAP
       NSURL *url = [NSURL
URLWithString:@"http://www.islancoelho.com.br/webservice/webservice.php"];
       NSMutableURLRequest *req = [NSMutableURLRequest requestWithURL:url];

       NSString *msgLength = [NSString stringWithFormat:@"%d",[soapMsg length]];
       [req addValue:@"text/xml; charset=utf-8" forHTTPHeaderField:@"Content-Type"];
       [req addValue:@"http://www.islancoelho.com.br/webservice/userIsValid"
forHTTPHeaderField:@"SOAPAction"];
       [req addValue:msgLength forHTTPHeaderField:@"Content-Length"];

       [req setHTTPMethod:@"POST"];
       [req setHTTPBody:[soapMsg dataUsingEncoding:NSUTF8StringEncoding]];
```

Figura 27: Trecho do código montando a mensagem para envio ao webservice

Comparação

GPS

O sistema de localização GPS se popularizou bastante nos últimos tempos, tornando muito comum e eficaz sua utilização em aplicações móveis.
No Android é possível estabelecer interfaces com sistemas de localização, para que possa ser desenvolvida uma aplicação utilizando o GPS, é necessário a solicitação de uso de tal recurso, isso também é feito através do AndroidManifest.xml através do código abaixo:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
```

No Android não é necessário nenhum tipo de framework ou classe externa, pois, o SDK tem todas as classes que trata o uso desse recurso, como as classes Location, LocationListner e LocationManager.
Quando fazemos uma requisição complexa no android, como é o caso da requisição a dados de GPS o sistema pode ficar lento durante o procedimento, por isso é necessário a criação de uma Thread que roda uma caixa de progresso para que isso não fique visível ao usuário, e o processo de requisição do GPS ocorra por traz do aplicativo.
Podemos verificar a latitude e longitude atual através da classe LocationManager, que possui um método getLastKnownLocation o qual retorna uma instancia da classe Location, que armazena a localização atual.

```
location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

location.getLatitude()

location.getLongitude()
```

Comparação

GPS

iOS

Utilizamos a função GPS para identificar a localização do usuário pelo aplicativo no IOS. É necessário adicionar o framework CoreLocation no projeto. Este framework permite determinar a posição atual do aparelho, através de informações como longitude e latitude. Sendo possível também monitorar se o usuário muda de localização.
No desenvolvimento do código é necessário o CLLocationManagerDelegate que, como em outras rotinas, é responsável por gerenciar a troca de mensagens entre a classe atual e a CoreLocation.

```
#import <CoreLocation/CoreLocation.h>
#import "DetalhesViewController.h"

@interface RootViewController : UITableViewController <CLLocationManagerDelegate,
NSXMLParserDelegate, UITextFieldDelegate>{

    DetalhesViewController *detViewController;
    IBOutlet UIActivityIndicatorView *indicadorAtividade;
    CLLocationManager *locationManager;
    CLLocation *startingPoint;
```

Figura 28: Trecho do código com os comandos inicias de criação das váriaveis para o GPS

Para que a localização seja atualizada toda vez que a tela for carregada, colocamos o código que verifica a posição do usuário dentro do evento DidAppear ou DidLoad (evento executado após a abertura e carregamento da tela, respectivamente).

```
- (void)viewDidAppear:(BOOL)animated
{
    cont=0;
    self.locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyBest;
    [locationManager startUpdatingLocation];

    [self refreshDisplay:self.tableView];

    [super viewDidAppear:animated];
}
```

Figura 29: Trecho do código com instanciamento e inicialização da atualização de GPS

Comparação ao desenvolvimento usando recursos de GPS

1. Modelagem de Banco

1.1 Modelo Entidade e Relacionamento

1.2 Diagrama Entidade e Relacionamento

1.3 Esquema

2. Modelagem de Classes

2.1 Caso de Uso

2.2 Diagrama de Classes, *Unified Modeling Language* (UML)

Conclusão

Instalação e configuração do ambiente de desenvolvimento

No iOs Ide e SDK é integrado tudo em uma única instalação que fica disponível na App Store ou através do site developer.apple.com. A instalação e configuração do ambiente de desenvolvimento é totalmente automatizada.

No Android, é necessário seguir alguns passos:
1) efetuar a instalação do SDK, disponibilizada através do site developer.android.com
2) Efetuar a instalação do Eclipse, disponibilizada através do site eclipse.org
3) efetuar a instalação do plug-in ADT, é feito a partir da IDE adicionando o repositório https://dl-ssl.google.com/android/eclipse/

Neste aspecto o desenvolvimento para plataforma Apple leva uma grande vantagem em comparação com o Android, já que em apenas um passo todo ambiente de desenvolvimento estará pronto para utilização.

Comparação IDEs

A IDE xCode utilizada no iOS é um ambiente de desenvolvimento fechado, ou seja, somente computadores Apple podem executar a aplicação, além disso é necessária uma licença anual para a utilização de todos os recursos da IDE.
O Eclipse é uma IDE opensource, consequentemente não é necessário adquirir nenhum tipo de licença e é multiplataforma.
Neste aspecto o Android leva vantagem por ser gratuito e multiplataforma.

Utilização de recursos de hardware e software

No iOS são necessários frameworks que para utilização de recursos de hardware e software no projeto, os principais frameworks são adicionados ao projeto durante sua criação, porem para acessar recursos específicos como GPS e Internet é necessário a instalação de frameworks que devem ser adicionados ao projeto de forma manual.
No Android não são necessários frameworks externos para utilização desses tipos de recursos, apenas é necessário solicitar permissões de acesso no arquivo de configuração AndroidManifest.xml.
Neste aspecto o Android o leva uma pequena vantagem já que não é necessário adicionar recursos externos ao projeto.

Simulador
No iOS não é necessário nenhuma configuração para utilização do simulador, apenas escolher o tipo de dispositivo (iPhone/Ipod ou iPad)
no Android é necessário criar uma maquina virtual com as configurações desejadas selecionando quantidade de memoria, versão do Android, tamanho de tela, entre outros.
Com isso o iOS leva vantagem já que não é necessário nenhum tipo de configuração adicional.

Webservice

Graças a uma biblioteca opensource chamada KSoap2 o acesso a um webservice Soap é bastante facilitado, utilizando essa biblioteca não é necessária a criação de métodos para requisições SOAP, basta utilizar métodos pré-definidos passando o nome do método do webservice e os parâmetros em forma de objetos.

Para utilizar a KSoap foi necessário apenas adicionar a biblioteca ao projeto e solicitar permissão de acesso no arquivo AndroidManifest.xml. Deixando o processo todo bastante simplificado.

O acesso a WebService através do iOS exigiu a criação dos métodos de requisição SOAP, conexão e leitura dos dados diretamente no código, aumentando consideravelmente o código, porém não é necessário configurar permissões de acesso à Internet.

Consideramos que, com a ajuda das bibliotecas prontas, o acesso a Webservice no Android é de fácil implementação se comparado ao iOS, que, apesar de não ser complexo, exige codificação em excesso.

GUI

No desenvolvimento para Android o desenho das telas pode ser facilitado pelo plugin ADT para eclipse, permitindo que o procedimento seja feito de forma visual, apenas clicando e arrastando os componentes para tela. Porém, muitas vezes, a dificuldade em posicionar os componentes na tela obriga a edição direta do arquivo XML, que facilita a alteração das propriedades do componente, como por exemplo, alteração das margens para posicioná-lo no local desejado.
Outro ponto é a comunicação entre a view e o resto da aplicação onde é obrigatória a utilização das activitys. Sem elas é impossível fazer o "binding" entre a tela e a aplicação.
No iOS, o desenho das telas é feito de forma bastante intuitiva, já que não existem as limitações encontradas no Android. Os componentes podem ser arrastados e livremente posicionados na tela de design, sem a necessidade da utilização de codificação XML.
A comunicação entre a view e o código também é feita diretamente através da tela de design, bastando arrastar, segurando a tecla Option, os componentes da tela para um ícone que representa o código, chamado de placeholder, ou vice-versa.
Com isso em GUI concluímos que o iOS leva vantagem pela facilidade na edição dos layouts e a praticidade na comunicação dos componentes com o código.

GPS

A utilização dos recursos de localização no Android é bem simples, basta fazer uma requisição de permissão de acesso ao recurso no arquivo AndroidManifest (assim como é feito para o acesso a Internet), e utilizar as classes Location, LocationListner e LocationManager (todas pertencem à SDK do Android) sem a necessidade de recursos externos. A única dificuldade encontrada com relação ao recurso foi referente a linguagem, uma vez que a aplicação trava durante a execução do processo.
Foi necessária a utilização de uma nova thread para que um aviso de "Aguarde"(Progress dialog) pudesse ser exibido neste intervalo.

No iOS, é necessário importar para o projeto o framework CoreLocation e utilizar as classes CLLocationManager e CLLocation. Toda a operação é bem simples e executada automaticamente em um processo separado, não interferindo no funcionamento do aplicativo.

Concluímos que ambas as plataformas são bem amigáveis para a utilização do recurso GPS, porém o iOS leva uma pequena vantagem no que se refere à concorrência na execução do código.

REFERÊNCIAS

Contexto da Mobilidade

MORAES, Marcos Roberto de. Sistema de Pedidos Para Restaurantes. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=10326>. Acesso em 31 Out. 2010.

FOLHA.COM. Saiba o significado de algumas siglas ligadas à telecomunicação. Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u6096.shtml>. Acesso em 14 Nov. 2010.

KURA, Marcelo. Telefonia Móvel, São Paulo, VOCÊ S/A, edição 147, p. 92, set. 2010.

MARQUEZI, Dagomir. Meu *IPhone* multiuso, São Paulo, Info, ed. 295, p.26, set. 2010.

Smartphones

[1]1,7 bilhões de Smartphones no mundo até 2014 <http://www.maniadecelular.com.br/82235/17-bilhoes-de-smartphones-no-mundo-ate-2014.html> Acessado em 05/04/2011

Android

TOZETTO, Claudia Android Market ultrapassa 100 mil aplicativos <http://tecnologia.ig.com.br/noticia/2010/10/25/android+market+ultrapassa+100+mil+aplicativos+9839101.html> Acessado em 30/04/2011

Ferrer, RAFAEL, Android é a plataforma mais usada em novos smartphones <http://tecnologia.ig.com.br/noticia/2010/10/05/android+e+a+plataforma+mais+usada+em+novos+smartphones+diz+pesquisa+9610160.html> Acessado em 30/04/2011

Livro:

Google Android “Aprenda a criar aplicações para dispositivos móveis com o Android SDK” - Autor: Ricardo R. Lecheta - Editora: Novatec

http://developer.android.com/guide/basics/what-is-android.html or same site http://developer.android.com

iPhone

Apple Computer (2011). iOS Programming Guide. Apple Computer Inc, Disponível em: http://developer.apple.com.

Apple Computer (2011). Disponível em: http://www.apple.com

Dalrymple, M. and Knaster, S. (2008). Learn ObjectiveC on the Mac. Apress, Berkely, CA, USA.

Figura 1: iPhone Disponível em <http://www.uspto.gov/web/patents/patog/week11/OG/html/1364-3/USD0634319-20110315.html> Acesso em: 12/04/2011

Redes Sociais:

Filme:

The Social Network, 2010. Ben Mezrich, Aaron Sorkin.

Livro:

Castells, Manuel. A sociedade em rede Volume I.

Webservices e SOAP

Livro

Jorgensen, David. Desenvolvendo Serviços Web .NET com XML

Site

CUNHA, Davi Web Services, SOAP e Aplicações Web <http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html> Acesso em 10/05/2011

PHP

Site

Sobre PHP <http://www.php.net/manual/pt_BR/intro-whatis.php> Acesso em 02/06/2011

MySQL

First head PHP & MySQL, 1º edição O'Reilly, autores Michael Morrison e Lynn Beighley

Sobre o MySQL <www.mysql.com> Acesso em 26/05/2011