DESENVOLVIMENTO DE APLICAÇÃO MÓVEL COM ESTUDO DE CASO PARA CLÍNICA ESTÉTICA

Caio César Leonhardt[1]
Wilian Bouviér[2]

RESUMO

Empresas que desejam criar um vínculo mais forte com os seus clientes necessitam oferecer acessibilidade à informação. O *mobile marketing* é essencial para atender as características da sociedade moderna, onde a escassez de tempo torna o usuário cada vez mais seletivo em suas fontes de conhecimento. Desta forma, esse estudo busca abranger as duas principais plataformas de desenvolvimento para dispositivos móveis e apresentar a que melhor atende às necessidades da empresa Delux Centro de Estética Avançada. A partir do estudo verificou-se que a plataforma iOS se destacou em um âmbito geral, sendo definida como a plataforma a ser desenvolvida a aplicação para exibir as *dicas de beleza* e *promoções* disponibilizadas pela empresa.

Palavras chave: Desenvolvimento *mobile*. Dispositivos móveis. Android. iOS.

1. INTRODUÇÃO

A Delux Centro de Estética Avançada é uma empresa fundada em maio de 2011 na cidade de Passo Fundo - Rio Grande do Sul, atua no ramo de tratamentos das disfunções estéticas faciais, corporais e o reequilíbrio corpo e mente. Prezando pela valorização dos clientes, a empresa busca oferecer conforto e resultado através de equipamentos modernos, técnicas inovadoras e atualizações na área da beleza e bem-estar. Ainda, a mesma busca ser líder no setor através da transparência, resultado e marketing responsável. Para alcançar os seus objetivos ela utiliza variados meios de comunicação, como exemplo, revistas, comerciais e *website.* Contudo, sabemos que no contexto de acessibilidade à informação existem tendências na sociedade, e para a empresa se aproximar ainda mais de seus clientes, superando os meios convencionais, ela deve ser adaptativa e oferecer o que o mercado exige.

[1] Aluno do curso Tecnologia de Sistemas para Internet pelo Instituto Federal de Educação , Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSUL). Email: caioleonhardt@outlook.com.

[2] Orientador, professor do Instituto Federal de Educação , Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSUL), graduado em Tecnologia em Sistemas para Internet (2012). Email: nailiwb@gmail.com.

Deste modo, o mobile marketing oportuna o consumidor a ter acesso a informação a qualquer hora e qualquer lugar através dos dispositivos móveis.

Quando abordado o tema dispositivos móveis, existem diversas plataformas disponíveis para desenvolvimento, cada qual com suas vantagens e desvantagens. Enquanto uma plataforma possui mais usuários, outra é mais rentável e menos segmentada, valendo do comportamento e a necessidade inicial do consumidor para que seja escolhida a mais adequada.

Neste trabalho é apresentado o desenvolvimento de um aplicativo destinado a apresentação de conteúdo, tomando como base as necessidades da Delux em um marketing moderno e eficaz. Assim, tem-se um referencial teórico apresentando as duas plataformas com mais destaque no mercado de dispositivos móveis, seguido dos métodos e abordagens utilizadas para o desenvolvimento, tal como ferramentas, tecnologias e a documentação. Além disso, o desenvolvimento apresenta detalhes do projeto e códigos utilizados na aplicação e, por fim, são apresentadas as considerações finais.

2. REFERENCIAL TEÓRICO

Segundo Rogers et. al. (2009), afirma que em novembro de 2007 foi anunciado o Sistema Operacional (SO) para dispositivos móveis chamado Android, uma plataforma *open-source* desenvolvida com base no *kernel* do SO *Linux* por um grupo de empresas chamado *Open Handset Alliance* (OHA), liderados pela Google Inc. Aplicações desenvolvidas para esta plataforma utilizam linguagens de programação Java ou C/C++. Para aplicações que necessitam de desempenho é disponibilizado o Kit de Desenvolvimento Nativo (abreviatura em inglês, NDK), no qual os códigos são escritos em C/C++. Contudo, a maioria das aplicações são desenvolvidas em Java, fazendo uso do *Software Development Kit* (SDK) (ANDROID NDK, 2014).

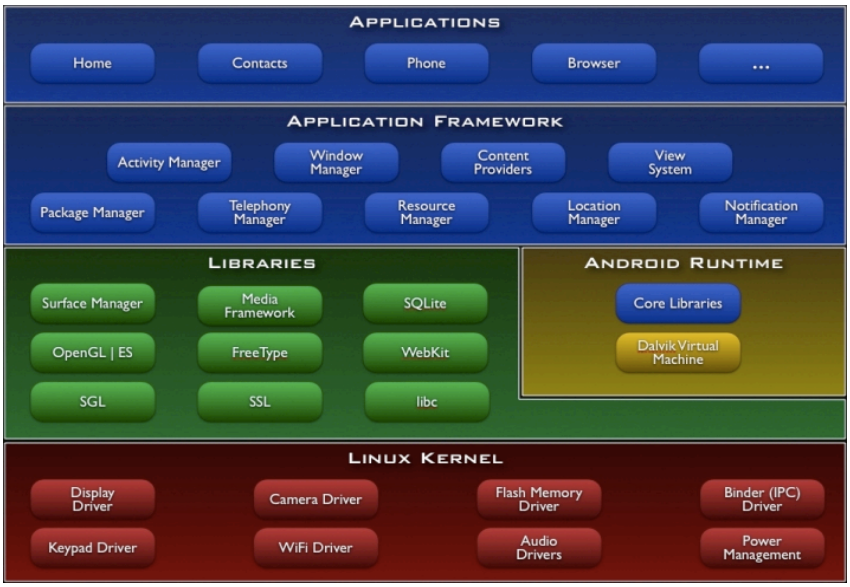
Niemeyer (2013) explana que a história do Java se iniciou em 1990 na *Sun Microsystems*, sob a orientação de James Gosling e Bill Joy. Seu projeto envolvia ser uma linguagem de programação independente de plataforma, segura o suficiente para percorrer nas redes e com desempenho o suficiente para substituir o código nativo. Da mesma forma, Santos (2003) colabora com a ideia de que as fronteiras que separavam eficácia e simplicidade de desenvolvimento foram ultrapassadas, compondo uma linguagem de programação robusta, na qual, os erros mais frequentes e limitações da linguagem C/C++ foram considerados desde o início (SANTOS, 2003). O código Java pode ser executado em

qualquer plataforma, ou seja, o programa que utiliza essa linguagem poderá ser executado em SOs Windows, Mac, Linux, etc. Primeiramente o código é compilado em um formato de *bytecodes* (formato universal de instruções para máquina virtual), após, poderá ser executado pelo interpretador Java em qualquer ambiente (NIEMEYER, 2013).

Para o Android, a Google optou por não seguir os padrões, adotando uma implementação alternativa para a máquina virtual (abreviatura em inglês, VM) Java, assim ela desenvolveu uma VM otimizada, capaz de lidar com velocidade de processamento limitado, memória RAM limitada, energia fornecida por bateria e grande diversidade de dispositivos, essa VM foi chamada de Dalvik (EHRINGER, 2010).

Na Figura 1, é ilustrada a arquitetura da plataforma Android. Estando na mais alta das camadas as aplicações que irão estar disponíveis ao usuário. Logo abaixo, em *Application Framework,* estão disponíveis as Interfaces de Programação de Aplicações (abreviatura em inglês, API) recursos utilizados pelos aplicativos, entre as principais está a *Activity Manager*, *Windows Manager e View System*. Na camada de bibliotecas são encontradas funções para acesso ao banco de dados (SQLite), funções para renderização 2D e 3D, bibliotecas das áreas de multimídia e também fontes bitmap e vetorizadas. Em *Android Runtime* (ambiente de execução), é carregada uma instância da máquina Dalvik para cada aplicativo em execução. Por fim, o Kernel Linux contém todo o gerenciamento de hardware por parte do sistema operacional, como a gestão de memória, gestão de processos, drivers de câmera e energia (PEREIRA, 2009).

Figura 1 Arquitetura da plataforma Android



Fonte: DEVMEDIA, 2014.

Para cada componente utilizado na aplicação, deve ser declarado o seu uso em um arquivo de *manifest*. Estes componentes possuem um ciclo de vida próprio e são responsáveis pela comunicação do *app* com o sistema (ANDROIND, 2014). Um componente muito utilizado é a *Activity*, ela representa uma tela da aplicação, resumidamente, ela é responsável pelo controle de estados da tela, passagens de parâmetros para outra *Activity* e o controle de eventos. No entanto, a *Activity* não possui a habilidade de renderizar elementos visuais, a responsável por desenhar na tela é a classe *View* (LECHETA, 2010). Frequentemente é necessário a utilização de serviços que rodem em segundo plano, para tal tarefa o Android possui um componente chamado *Services*, o ciclo de vida dele difere de uma Activity pois não sofre interrupções em seu estado como tal, como exemplo, uma aplicação pode tocar uma música MP3 sem afetar a interação do usuário com a sua execução. Ainda possuem dois componentes essenciais, o *Content Provider*, que permite o compartilhamento de dados entre aplicativos e o *Bradcast Receiver*, que é responsável por tratar eventos externos recebidos por uma *Intent* (GARGENTA; NAKAMURA, 2014).

Por fim, quando necessário, uma aplicação que requer salvar os dados no telefone, pode utilizar o banco de dados SQLite, que segundo Rogers et al. (2009), o banco de dados SQLite não é um projeto da Google, este é de domínio público, podendo ser utilizado para fins comerciais ou privados. O SQLite é uma biblioteca escrita na linguagem de programação C, que implementa um banco de dados transacional com propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade)[3], tornando-se adequado para situações que não exijam recursos avançados de um SGBD.

Segundo Kochan (2012), a empresa multinacional Apple Inc. anunciou o iPhone no ano de 2007. Porém, o desenvolvimento de aplicações nativas era restrito a empresa, sendo que terceiros não possuíam esta capacidade. A forma que a Apple tentou atrair desenvolvedores para sua plataforma, foi disponibilizar a produção de aplicativos baseados na web. Assim, as aplicações seriam executadas no web browser Safari, fato que não deixou os desenvolvedores satisfeitos, pois haviam muitas limitações. O SDK para o desenvolvimento de aplicações nativas foi publicamente anunciado em março de 2008. Isso permitiu que as aplicações fossem instaladas, executadas e testadas diretamente no sistema operacional do iPhone, oferecendo os recursos de hardware, como, câmera, acelerômetros (ALLAN, 2010). O sistema iOS utiliza a linguagem Objective-C, a mesma foi desenvolvida por Brad J. Cox

[3] ACID é um conceito utilizado para definir as propriedades que uma transação em banco de dados deve possuir para garantir a integridade dos dados.

no início dos anos 1980, sendo uma extensão da linguagem C com suporte a programação orientada a objetos baseada na linguagem chamada *Smalltalk-80* (KOCHAN, 2012).

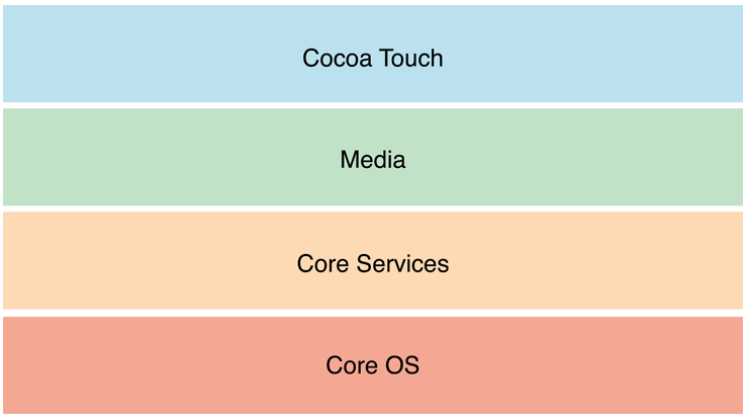
Allan (2010) afirma que o modelo de orientação a objeto de Objective-C baseia-se no envio de mensagens para instâncias de objetos, isso caracteriza a linguagem como sendo mais dinâmica em tempo de execução em comparação a linguagens de programação baseadas em Simula (considerada primeira linguagem de programação orientada a objetos e influência no desenvolvimento de outras linguagens, por exemplo, C++ e Java), pois o objeto que recebe a mensagem não precisa conhecer o tipo de dado, ele irá tentar interpretar a mensagem e caso não consiga entendê-la, retornará nulo.

Fairbairn, Fahrenkrug e Ruffenach (2012), explicam que uma importante diferença da plataforma é o gerenciamento de memória. Enquanto Java utiliza *Garbage Collection*, Objective-C utiliza *Reference counting* (em português, contagem de referência). Este conta o número de proprietários (referências para memória) que um objeto possui, assim, quando um elemento necessita o objeto, a contagem é incrementada. Da mesma forma, quando um elemento libera o objeto, a contagem é decrementada. Uma vantagem de implementar esse conceito de posse, são os custos mínimos de recursos computacionais.

Para desenvolver aplicativos para iOS, é necessário um computador com o sistema operacional OS X, requisito para instalação do Ambiente de Desenvolvimento Integrado (abreviatura em inglês, IDE), chamada Xcode. A mesma possui um conjunto de ferramentas e frameworks que permitem criar, instalar, depurar e testar aplicações nativas (APPLE, 2014).

Como ilustrado na Figura 2, a arquitetura do iOS apresenta níveis de abstrações. Nas camadas mais altas estão presentes serviços já encapsulados e frameworks visuais e mídia, e nas camadas mais baixas estão os serviços fundamentais e o core do sistema (APPLE, 2014).

Figura 2 - Arquitetura da plataforma iOS



Fonte: APPLE, 2014.

Finalmente, a persistência de dados no iOS se deve a uma biblioteca compacta, rápida e confiável escrita na linguagem C, chamada SQLite. Mesmo possuindo algumas limitações para processamento de grandes dimensões de dados, ele se torna efetiva em aplicações em dispositivos com poucos recursos computacionais e processamentos menores (BENETT; FISHER & LESS, 2010). No entanto, a utilização do SQLite é complexa e exige muita codificação. Assim, a Apple Inc. desenvolveu o *framework* Core Data, o qual fornece as principais funcionalidades de um banco de dados, implementando por completo a camada de modelo. Toda a sua utilização requer orientação objeto, e a criação de tabelas e entidades pode ser feita através de interface gráfica (MARZULLO, 2012).

3. METODOLOGIA

A Delux Centro de Estética Avançada, buscando aumentar seu mercado, sentiu a necessidade da criação de um marketing moderno, rápido, com bom custo benefício e que atendesse as características da sociedade atual. Ehrenberg (2011), explica a importância do uso de estratégias digitais, de forma a tornar acessível todos os produtos e serviços prestados aos consumidores. No Brasil, o *mobile marketing* corresponde a um meio de comunicação eficaz e assertivo, de modo que atende positivamente as principais necessidades dos consumidores atuais.

O desenvolvimento do aplicativo será conduzido com base no referencial bibliográfico extraído de livros técnicos nacionais e internacionais. Outras referências importantes foram sites oficiais das plataformas estudadas e sites organizacionais das tecnologias *open source* utilizadas. Ainda, visando as melhores práticas e padrões de desenvolvimento, a busca bibliográfica abrange publicações do período de 2009 a 2014.

Possuindo o embasamento necessário sobre as principais plataformas *mobile*, se esclareceu aos diretores da Delux os recursos que cada plataforma dispõe conforme a Tabela 1. Com isso, foi definida a plataforma que melhor se ajusta às necessidades atuais da empresa, levando em consideração aspectos como, quais dispositivos seus clientes utilizam (através de pesquisa informal entre os colaboradores da empresa) e facilidade no desenvolvimento da aplicação, sendo por parte de codificação ou criação de layouts (ponto previsto pois tem implicação direta na métrica de tempo). Assim, optando por fornecer uma experiência personalizada aos seus clientes fidelizados, chamados de Clientes Ouro, a empresa deteve seu interesse na plataforma iOS, tendo como base a pesquisa informal obtida através dos colaboradores para saber o dispositivo usado pelos clientes.

Tabela 1 - iOS e Andoid

Tópico	IOS	Android
Linguagem de Programação	Objective - C	Java
Notificações	Local/Remota	Local/Remota
Criação do Layout	Storyboard	XML
Interface de desenvolvimento	Xcode	Eclipse
Serviços em background	Requer autorização do usuário	Sim

Fonte: Do Autor

As ferramentas, softwares e técnicas utilizadas serão esclarecidos no decorrer do desenvolvimento. O estudo de caso, ou seja, o desenvolvimento de um aplicativo para dispositivo móvel, que motivou este trabalho foi obtido através de pesquisa com a diretoria da Delux, resultando na elaboração dos requisitos funcionais e não-funcionais. Segundo Guedes (2011), o primeiro corresponde com as funcionalidades que o software deve realizar e o outro com as condições de negócio que o mesmo possui.

De modo geral, foi relatada a necessidade de uma aplicação que facilite a visualização das *dicas de beleza* e *promoções* que a clínica oferece. Conjuntamente, se estabeleceu que o layout deve seguir o padrão de cores utilizados na logo, ou seja, preto e durado.

Requisitos Funcionais:

- Exibir as últimas cinco dicas cadastradas;
- Exibir promoções disponíveis do mês;

Requisitos Não-Funcionais:

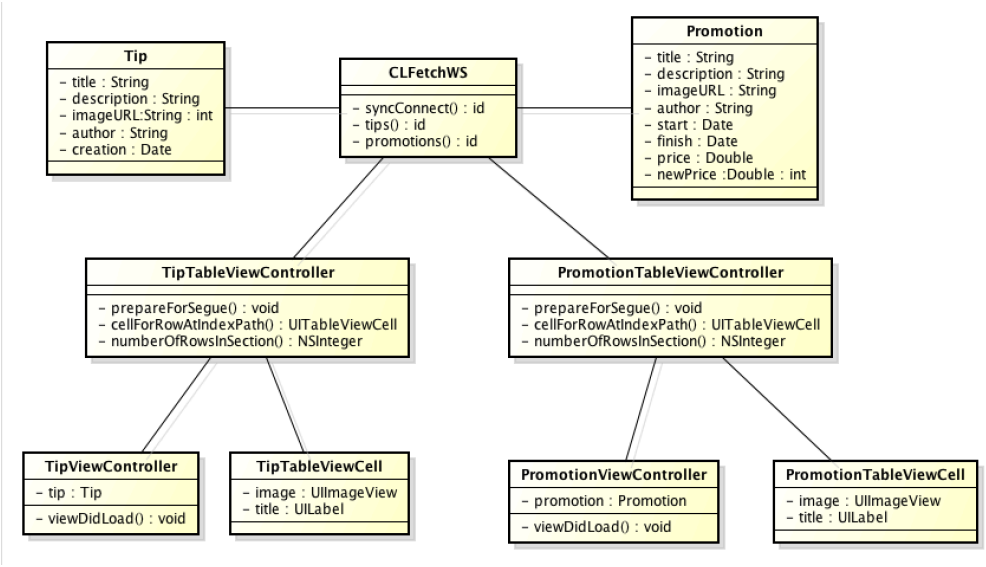
- Seguir o padrão de cores preto e dourado para o layout;

Para documentação do sistema, foi utilizada a UML (Unified Modeling Language), uma linguagem de modelagem de softwares adotada em 1997, que permite aos engenheiros de software definirem os requisitos do sistema, suas características e comportamentos de modo visual. A documentação do sistema é requerido pois sistemas de informação estão em constante crescimento, seja pelo desejo dos clientes por melhorias ou adaptações. Assim, um sistema bem documentado facilita a manutenção do software (GUEDES, 2011).

Guedes (2011), afirma que o diagrama de classes permite a visualização das classes que estarão presentes no sistema, detalhando quais os atributos e métodos que a compõe, assim como suas respectivas relações com as demais classes. O fato deste diagrama definir a estrutura lógica do sistema, torna este um dos mais importantes diagramas da UML.

A Figura 3 mostra o diagrama de classes utilizado para este projeto, apresentando as classes utilizadas para manipular dados de entidades (objetos CoreData), a classe responsável por conectar com o *webservice* e as classes controladoras da interface gráfica.

Figura 3 - Diagrama de classes



Fonte: Do Autor, 2014.

Para manipular os dados recebidos pelo *webservice* são utilizadas as classes Tip e Promotion, as mesmas são uteis para mapear a estrutura JSON e assim utilizá-las para exibir os dados na interface gráfica. Para que essa conexão se realize, a classe *CLFetchWS* possui métodos responsáveis para tal atividade, sendo que o método *syncConnect* permite a aplicação receber uma URL qualquer e se conectar de forma síncrona a tal recurso, retornando para a aplicação um ponteiro *id*, ainda na classe *CLFetchWS*, os métodos restantes possuem acesso específico para os recursos da aplicação, ou seja, para se conectar ao *webservice* das dicas e promoções. As classes com sufixo *Controller* e *Cell* possuem o papel de controlar a interface gráfica da aplicação, deste modo, possuem os atributos com elementos especificados no *Storyboard* criando a conexão entre os mesmos, e contém os métodos que atuam na passagem de parâmetros e delegação de eventos.

Neste projeto será usado o Git como sistema de controle de versão (sigla em inglês, VCS), de acordo com Chacon (2009), um VCS é responsável por armazenar mudanças que ocorrem em um arquivo, permitindo que diferentes versões do código sejam gerenciados ao mesmo tempo. Em outras palavras, através de uma versão estável do código, criam-se ramificações com novas implementações necessárias. Caso haja alguma questão que futuramente ocasionará em defeito, as modificações são facilmente desfeitas. Ainda, o Git foi desenvolvido pela comunidade de desenvolvimento Linux em 2005 com os objetivos de fornecer aos seus usuários:

- Velocidade;
- Interface simples;
- Forte suporte para desenvolvimentos paralelos;
- Sistema de controle de versão totalmente distribuído;
- Habilidade de manipular grandes projetos.

4. DESENVOLVIMENTO

Obtidos os requisitos e a documentação necessária para o desenvolvimento, a fonte de dados que será utilizada para o aplicativo popular seus objetos será obtida através de um *Web Service RESTful.* Segundo Richardson e Ruby (2007), *Web Services* são serviços de softwares acessados remotamente, afim de obter dados ou processar determinado algoritmo, com um protocolo definido através de uma rede. O termo *RESTful* é um paradigma para a construção de sistemas distribuídos, seus princípios são simples e bem definidos, utiliza o protocolo HTTP (*Hypertext Transfer Protocol*) como meio de transporte para suas requisições. Este protocolo faz a comunicação através do modelo cliente-servidor, enviando mensagens que indicam a ação a ser realizada e o recurso especificado na URI (*Universal Resource Identifier*), e o servidor, por sua vez, retorna uma mensagem de resposta ao cliente.

Conforme Figura 4, segue um exemplo de implementação de um serviço web na linguagem de programação Java utilizado para o *app*. A classe *PromotionWS* é responsável por receber uma requisição e listar todas as promoções disponíveis. Na linha 12, é especificado qual o caminho que esta classe responderá às requisições. As linhas 15 e 16 definem alguns comportamentos que o serviço deve apresentar, sendo que o primeiro indica que o serviço só responderá a requisições HTTP GET, e o segundo que o retorno do serviço será no formato JSON (*JavaScript Object Notation*). O site JSON ORG (2014), explica que JSON é uma estrutura de dados utilizada para o armazenamento e transmissão de informações

em formato de texto. Ainda, sua sintaxe leve torna a interpretação rápida e eficaz. As linhas 19 e 21 são responsáveis por acessar um repositório, ou seja, recuperar do banco de dados as promoções disponíveis e retornar ao cliente.

Figura 4 - Exemplo de código para Webservice REST

```
1   package com.delux.webservice;
2
3   import ...7 linhas
10
11
12  @Path("/promotions")
13  public class PromotionWS {
14
15      @GET
16      @Produces(MediaType.APPLICATION_JSON)
17      public List<Promotion> getAvailablePromotion() {
18
19          PromotionRepository repository = new PromotionRepository();
20
21          return repository.findAvailable();
22      }
23  }
```

Fonte: Do Autor, 2014.

Com os webservices prontos para responder a solicitações, o aplicativo ou qualquer outro software capaz de criar conexões HTTP já estaria apto a utilizar os dados referentes as *promoções* e *dicas de beleza*.

A criação do projeto iOS no Xcode requer algumas configurações iniciais, as mesmas são listadas com seus respectivos valores.

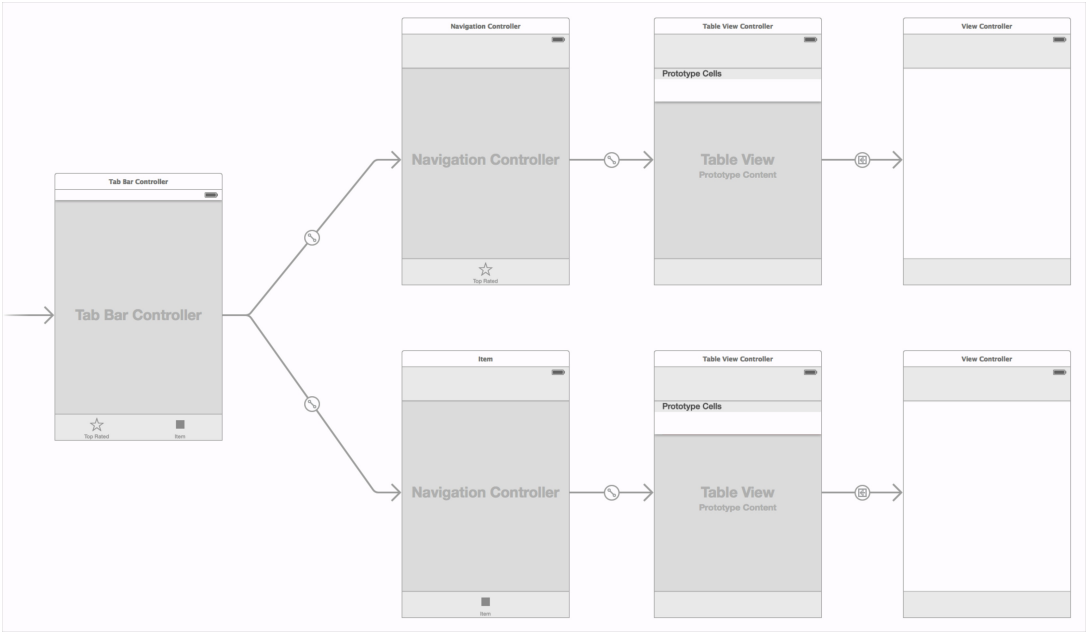
- Template : Single View Application
- Nome : Delux
- Nome da organização : Caio Leonhardt
- Identificador da companhia : caioleonhardt
- Prefixo de classes : CL
- Dispositivos : Iphone
- Use CoreData : Yes

Nas configurações do projeto se nota a presença do prefixo de classes (não encontrado em linguagens como Java), este é importante na criação de classes, protocolos e constantes, pois a linguagem Objective-C não possui *namespaces*, assim, para proteger a aplicação de conflitos entre frameworks, são utilizadas essas distinções nos nomes (APPLE, 2014).

Na Figura 5, a interface da aplicação é modelada com o uso do *Storyboard*. O *Storyboard* é a representação visual da interface do usuário, apresentando todas as telas,

componentes visuais e a conexão entre elas. Também permite identificar qual o controlador responsável por cada tela, analisar o fluxo e aparência da interface (APPLE, 2014).

Figura 5 - Modelo visual em Storyboard



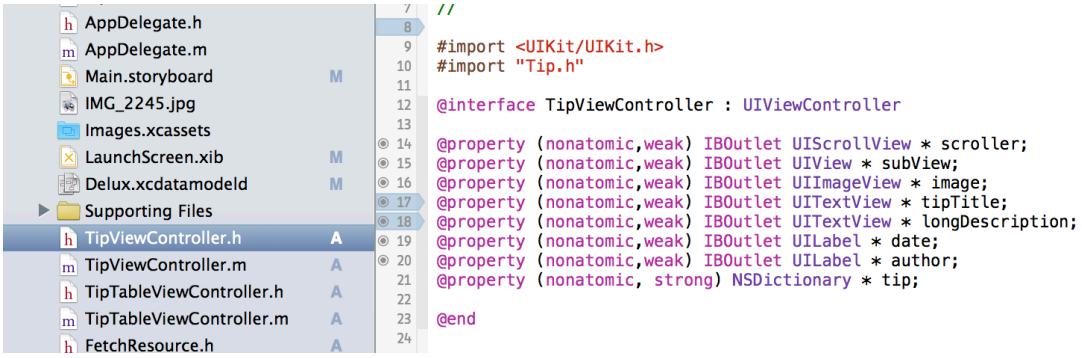
Fonte: Do Autor, 2014.

A *Tab Bar Controller* é utilizado para organizar as diferentes perspectivas do *app*, ou seja, permite a navegação entre as telas através dos botões de rodapé, como um menu rápido. Cada botão no *Tab Bar Controller* está associado a uma *Table View Controller* (que exibirá o conteúdo carregado via webservice), que por sua vez, possui um *Navigation Controller* para gerenciamento de telas e hierarquia de componentes (APPLE , 2014). Para visualizar com detalhes os dados inseridos na célula, existe uma conexão entre a mesma e uma *View Controller*, esta conexão é chamada de *segue.* Conforme Apple (2014), é disparada uma *segue* a cada evento do usuário relacionada a transições de tela, podendo passar parâmetros ou configurar o tipo da transição. Ainda, uma *View Controller* representa uma tela na aplicação, nesta são adicionados textos, imagens, botões, formulários ou outros componentes que estendem a classe *UIView.*

Após a definição do layout, é necessário especificar um controlador que responderá aos eventos da tela e também populará o conteúdo na mesma. Um exemplo de controlador é a classe *TipViewController*, especificamente, ela receberá um parâmetro da célula selecionada na tabela (através de uma *segue*), e ao iniciar, colocará os dados recebidos devidamente nos componentes definidos pelo desenvolvedor.

A Figura 6 apresenta a definição de uma classe e suas propriedades responsáveis por controlar os componentes gráficos de uma tela, com ela podemos demonstrar a maior parte dos conceitos da linguagem Objective-C.

Figura 6 - Interface TipViewController



Fonte: Do Autor, 2014.

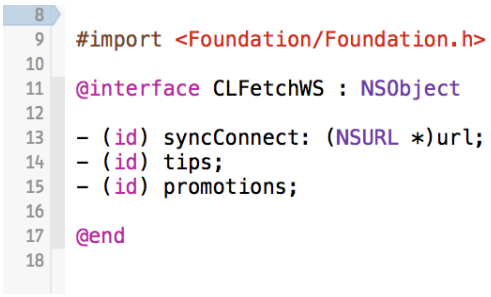
De acordo com Apple (2014), o *UIKit* fornece a arquitetura necessária para gerenciar a interface de usuário, manipular os eventos disparados pela interface e também a comunicação com o sistema. Concordando com Fairbairn, Fahrenkrug e Ruffenach (2012), a diretriz *@interface* sinaliza a declaração de uma interface, esta é a face visível para outros usuários, todas as propriedades e métodos devem estar declaradas entre esta diretiva de compilador e *@end*. Ainda, a classe deve estar em um arquivo com extensão "*.h"*. Exemplo de uma interface em Objective-C: @interface TipViewController : UIViewController @end.

Segundo Apple (2014), a definição de propriedades utiliza a diretiva *@property*, a mesma fornece um atalho na sintaxe do código, detalhadamente, com essa diretiva o compilador irá criar os métodos *getters* e *setters*, também definirá qual será a configuração de memória desse componente. O termo *nonatomic* significa que a variável não será sincronizada. A palavra *weak* sinaliza que o componente não possuirá uma referência forte ao objeto, isso é útil para os componentes visuais que podem requerer ser desalocados quando há transições de tela e o objeto pai deve ser desalocado. Já o *IBOutlet* cria uma referência de componentes criados no *storyboard* para serem usados no código fonte. Exemplo de código de propriedade: @property (nonatomic,weak) IBOutlet UIScrollView * scroller;

A implementação dos métodos da classe se deve ao arquivo TipViewController.m, este conterá todo o código fonte proposto pela interface. O código deve obedecer a sintaxe *@implementation* e *@end* (MARZULLO, 2012).

O próximo passo foi a realização da conexão do aplicativo com o *webservice*. Nesta etapa, uma classe deve possuir a capacidade de realizar uma conexão HTTP para uma URL específica, receber os dados que estão no formato JSON, preparar os dados para que a classe de controle manipule-os, e assim, enviá-lo para o controlador. Assim, com este propósito foi criada a classe *CLFetchWS,* conforme apresenta a Figura 7.

Figura 7 - Classe responsável por conexão com webservice



```
8
9   #import <Foundation/Foundation.h>
10
11  @interface CLFetchWS : NSObject
12
13  - (id) syncConnect: (NSURL *)url;
14  - (id) tips;
15  - (id) promotions;
16
17  @end
18
```

Fonte: Do Autor, 2014.

A declaração de métodos em Objective-C também possui uma sintaxe diferente de linguagens como Java. O sinal de menos que precede indica que o método será de instância, ou seja, sua atuação envolverá apenas o objeto referenciado. Entre parênteses é declarado o tipo de retorno. Em seguida, o nome do método e os parâmetros (caso possua) que receberá. Objective-C também possui um tipo de dados especial que pode referenciar qualquer objeto, sendo útil quando é necessário retornar tipos genéricos de dados, chamado de *id* (ALLAN, 2010).

Com a estrutura completa, a aplicação já permite a apresentação de dados e a interação do usuário com a aplicação. Ao iniciar a aplicação será exibida uma tela inicial para apresentar a Delux e sua logomarca. Em seguida já são apresentadas dicas recebidas do webservice, de mesmo modo o usuário pode escolher visualizar as promoções no menu inferior. A Definição destas telas pode ser observada através de algumas figuras que são demonstradas a seguir.

A Figura 8 representa a tela inicial da aplicação, no momento que o usuário clicar no ícone da aplicação no menu do sistema, será exibida esta tela. Na página de configuração do projeto pode ser definida uma tela padrão ao iniciar, utilizando deste meio, foi desenvolvida uma tela apresentando a logomarca da empresa. Para enfatizar a visualização desta, a tela foi configurada com uma duração de cerca de dois segundos até que o restante da aplicação carregue por completo.

Figura 8 - Tela Inicial do Sistema

Fonte: Do Autor, 2014.

No momento que a aplicação inicia, os dados são carregados na tabela e assim apresentados para o usuário como pode ser observado na Figura 9.

Figura 9 - Tela de exibição das dicas

Fonte: Do Autor, 2014.

Na tabela que lista as informações para o usuário, estão presentes a imagem e o título da mesma. A aplicação também permite que o usuário selecione com o toque estes componentes (imagem ou título), sendo direcionado para a tela que exibe em detalhes toda a informação. A tela de detalhe é representada na Figura 10.

Figura 10 - Tela de detalhe

Fonte: Do Autor, 2014.

A tela de detalhamento apresenta além do título e imagem, o autor que criou o conteúdo, a data de publicação e a descrição da dica. As imagens carregas na aplicação utilizam um componente nativo da linguagem que através da URL da imagem automaticamente faz o download para o objeto. Ainda, o texto apresentado como descrição está em formato puro, necessitando tratamento para remoção de tags html. Conforme o tamanho do texto e das informações dispostas, o tamanho da *View* aumenta verticalmente, permitindo o usuário fazer a rolagem da tela.

5. CONCLUSÃO

Por fim, é notável a facilidade com que se pode criar aplicativos para plataforma iOS. Os recursos da linguagem Objective-C e a IDE fornecem uma grande vantagem ao desenvolvedor quando se trata de produtividade, afirmativa que vai de encontro com os achados de Fairbairn, Fahrenkrug e Ruffenach (2012).

Assim como Bouviér (2012) relatou em seus estudos, é importante ter um projeto bem documentado, permitindo a codificação com qualidade, e a escalabilidade para possíveis projetos futuros. Ainda, notou-se o fato da gama de recursos que estão disponíveis para o desenvolvedor através dos *frameworks*, requerindo o aprendizado contínuo, o que permite uma aplicação com funcionalidades atrativas ao usuário.

Esse projeto é fruto de uma necessidade real que vem de encontro a uma sociedade altamente conectada e globalizada, justificando a necessidade de novos estudos e aprimoramentos no desenvolvimento. Com base no projeto desenvolvido, vertentes de trabalhos futuros podem ser identificados, sendo estes, trabalhos individuais dedicados a adição de novos recursos e funcionalidades a aplicação existente ou projetos que tenham foco no desenvolvimento em outras plataformas mobiles utilizando o mesma documentação. Caso seja optado pela ampliação de recursos e funcionalidades, novas especificações funcionais podem ser obtidas com a Delux afim de proporcionar maior usabilidade e conforto aos seus clientes, como exemplo, notificações remotas quando uma nova promoção ou dica estiver disponível, visualização de tratamentos, visualização de horários disponíveis e agendamento de tratamentos. Em outra perspectiva, se novos projetos adotarem o desenvolvimento para outras plataformas móveis, a experiência obtida neste poderá auxiliar na criação do mesmo e a Delux terá uma abrangência maior no que se trata de divulgação de seus serviços.

Conjuntamente, os conceitos e metodologias utilizados neste projeto foram abordados no curso Tecnologia de Sistemas para Internet e consolidados em bibliografias de tecnologia.

REFERÊNCIAS

ALLAN, Alasdair. **Learning Iphone Programming**. Sebastopol : O'Reilly Media, Inc. , 2010.

ANDROID, Developers. Application Fundamentals. Disponível em: <https://developer.android.com/guide/components/fundamentals.html >Acesso em: 11 nov. 2014.

ANDROID NDK, Developers. Get Started. Disponível em: <http://developer.android.com/tools/sdk/ndk/index.html >Acesso em: 23 out. 2014.

APPLE INC. iOS Developer Library. Disponível em: <https://developer.apple.com/library/ios/navigation/ > Acesso em: 14 nov. 2014.

BENETT, Gary; MITCH, Fisher; LEES, Brad. **Objective-C for Absolut Beginners**: Iphone, Ipad, and Mac Programming Made Easy. New York: Apress, 2010.

BOUVIÉR, Wilian. **Sistema web mobile para controle de eventos**. Instituto Federal de Educação , Ciência e Tecnologia Sul-rio-grandense – IFSUL, Passo Fundo, 2012.

CHACON, Scott. **Pro Git : Everyting you need to know about the Git distributed source control tool**. New York : Apress, 2009.

DEVMEDIA. Android, a nova plataforma móvel. Disponível em: <http://www.devmedia.com.br/android-a-nova-plataforma-movel-parte-ii/8432>. Acesso em: 10 de nov. de 2014.

EHRINGER, David. THE DALVIK VIRTUAL MACHINE ARCHITETURE, 2010.

EHRENBERG, Karla Caldas. **Comunicação mercadológica em celulares: um panorama do mobile marketing brasileiro**. Universidade Metodista de São Paulo, 2011.

FAIRBAIRN, Christopher K.; FAHRENKRUG, Johannes; RUFFENACH, Collin. **Objective-C Fundamentals**. London, NY : Manning Publications, 2012.

GARGENTA, Marko; NAKAMURA, Masumi. **Learning Androi, Second Edition**. Sebastopol: O'Reilly Media, Inc., 2014.

GUEDES, Gilleanes T. A. **UML 2 : Uma abordagem prática**. São Paulo : Novatec Editora, 2011.

JSON ORG. Introducing JSON. Disponível em: < http://www.json.org >. Acesso em: 15 nov. 2014.

KOCHAN, Stephen G. **Programming in Objective-C, Fourth Edition**. 4. ed. Indianapolis : Developer's Library , 2012.

LECHETA, Ricardo R. **Google Android: aprenda a criar aplicações para dispositivos móveis com Android SDK / Ricardo R. Lecheta**. 2. ed. São Paulo : Novatec Editora, 2010.

MARZULLO, Fabio. **Iphone na prática : aprenda passo a passo a desenvolver soluções para iOS**. São Paulo : Novatec Editora, 2012.

NIEMEYER, Patrick; LEUCK, Daniel. **Learning Java, Fourth Edition**. Sebastopol : O'Reilly Media, Inc., 2013.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço da. **Android para desenvolvedores**. Rio de Janeiro : Brasport, 2009.

ROGERS, Rick et. al. **Android Application Development**. Sebastopol : O'Reilly Media, Inc., 2009.

RICHARDSON, Leonard; RUBY, Sam. **RESTful Web Services**. Sebastopol : O'Reilly Media, Inc., 2007.

SANTOS, Rafael. **Introdução à programação orientada a objetos usando Java**. Rio de Janeiro : Elsevier, 2003.