RESIGeT On-Line Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica Vol. 02 - Nr. 01 - 2012 Uni-FACEF CENTRO UNIVERSITÁRIO DE FRANCA

ADDRESSDROID: APRESENTAÇÃO DO DESENVOLVIMENTO E FUNCIONAMENTO DE UMA APLICAÇÃO ORIENTADA A OBJETOS PARA ANDROID

Guilherme Henrique Dias dos Santos[1]
guizeradias@gmail.com

Lennon Petrick Spirlandelli[2]
lennonpetrick@gmail.com

Reginaldo Aparecido Gotardo[3]
reggotardo@gmail.com

RESUMO

O presente trabalho apresenta um software desenvolvido pelos alunos do segundo ano do curso de Sistemas de Informação. Este software veio da necessidade de aperfeiçoar os conhecimentos existentes do Sistema Operacional Android, e desenvolver aplicações orientadas a objeto com uma arquitetura embarcada. Trabalhando com a composição e padrões do projeto. Do ponto de vista acadêmico, foram realizadas pesquisas bibliográficas para estudo e implementação deste software. O trabalho foi estruturado a partir do cronograma de implementação seguido, dividindo-se em: estudo, domínio das ferramentas de desenvolvimento e construção da aplicação para Android. No desenvolvimento da aplicação, foi relatado o funcionamento do software mostrando todas as telas do mesmo. Explicações sobre os pacotes utilizados no projeto também foram mostrados, bem como especificações das ferramentas utilizadas, como o programa Eclipse Indigo, o Sistema de Gerenciamento de Banco de Dados (SGBD) SQLite, Plugins e SDK Android e outros. Padrões de projeto como MVC, DAL, BLL, e UI foram detalhados a fim de familiarizar o assunto com o leitor. Foi incluída também a história do surgimento do S.O. Android, com o relato de camadas, e algumas das bibliotecas que são necessários para o seu funcionamento estável, tais como, Java Virtual Machine (Máquina Virtual Java) Dalvik, o SGBD já citado, Bionic e OpenGL/ES, todos construídos na linguagem de programação C/C++. Dentre uma das camadas está a do micronúcleo, que é utilizado o Kernel Linux, assim assemelhando-se neste aspecto com os Sistemas Operacionais de plataforma desktops e mainframes, como os drivers dos componentes: teclado, câmera, Wi-Fi, áudio, display e outros.

Palavras-chave: Mobilidade; Orientação a objetos; Softwares.

[1] *Discente do Curso de Bacharelado em Sistemas de Informação do Centro Universitário de Franca Uni-FACEF*

[2] *Discente do Curso de Bacharelado em Sistemas de Informação do Centro Universitário de Franca Uni-FACEF*

[3] *Docente do Curso de Bacharelado em Sistemas de Informação do Centro Universitário de Franca Uni-FACEF*

ABSTRACT

This paper presents a software developed by the students of second year of Information Systems. This software came from the need to improve existing knowledge of the Android operating system, and develop object-oriented applications with an embedded architecture. Working with the composition and design standards. From an academic standpoint, literature searches were performed to study and implement this software. The work was structured from the implementation schedule followed, divided into: study the field of development tools and application development for Android. In application development, it was reported the operation of the software screens showing all the same. Explanations of the packets used in the project were also shown, as well as specifications of the tools used, how the program Eclipse Indigo, The Management System Database (DBMS) SQLite, Plugins and Android SDK and others. Design patterns like MVC, DAL, BLL, and UI were detailed in order to familiarize the reader with the subject. It also included the story of the rise of the Android OS, with a report of layers, and some of the libraries that are required for stable operation, such as Java Virtual Machine Dalvik, the DBMS already mentioned, Bionic and OpenGL / ES, all built in programming language C / C + +. Among one of the layers is the microkernel, which is used the Linux`s kernel, thus resembling in this respect with the OS platform desktops and mainframes, as the component drivers: keyboard, camera, Wi-Fi, audio, display and others.

Keywords: *Mobility*; *Object oriented*; Software.

1. Introdução

O projeto AddressDroid foi oriundo do interesse em aplicar os conhecimentos adquiridos em Orientação a Objetos, com aplicabilidade nas novas tecnologias de mercado, sendo este também o principal motivo para desenvolvimento deste projeto para Android.

Dispositivos e tecnologias mobile são a tendência do momento, e terão grande impacto nas gerações futuras. (CAVALLINI, et al., 2010)

Este artigo constitui-se na apresentação de um produto, sendo assim, foi utilizada uma pesquisa exploratória, com ênfase na coleta de dados por meio de pesquisa bibliográfica, e comparação do sistema aqui relatado com os sistemas Android existentes para que pudesse adquirir conhecimentos para realização do desenvolvimento. Também foi embasado na exploração no desenvolvimento do aplicativo, não sendo feitos testes com o usuário devido à escassez do tempo que se tinha para a realização deste.

O artigo está dividido em três assuntos referentes ao projeto AddressDroid. O primeiro assunto são os componentes utilizados para a fabricação do software, além de métodos comparativos entre ferramentas existentes no mercado. No segundo assunto são citados os padrões de projeto utilizados para a obtenção deste sistema, sendo seu principal propósito, simplificar a programação e manutenção do código do mesmo. No terceiro é relatado o projeto em si, ao qual é diretamente descrito por um cronograma de desenvolvimento da aplicação.

2. Componentes

Para o desenvolvimento do projeto AddressDroid, sendo uma aplicação para Android foi necessário utilizar a ferramenta de desenvolvimento Eclipse Indigo, e do sistema de gerenciamento de banco de dados (SGBD) SQLite.

Para compreensão do sistema operacional Android, foram feitas pesquisas bibliográficas em sites de discussão de problemas de software, e sites sobre padrões de projeto, além da leitura de apostilas sobre programação orientada a objeto para Android.

2.1. História do Android

O Sistema Operacional Android surgiu no ano de 2005 pela empresa Android.Inc, uma pequena empresa da Califórnia, USA, que mais tarde foi comprada pela Google. Inicialmente o Android foi comprado com o intuito de competir com o iPhone da Apple. Mais tarde a Google criou a Open Handset Alliance, que é um consórcio de empresas de tecnologia operantes no mercado de dispositivos móveis (HTC, Sony, Intel, Motorola, Samsung, etc.). (PRADO, 2011)

2.2. Por dentro do Android

E ainda de acordo com PRADO, o sistema operacional Android é basicamente dividido em quatro camadas como segue a figura abaixo:

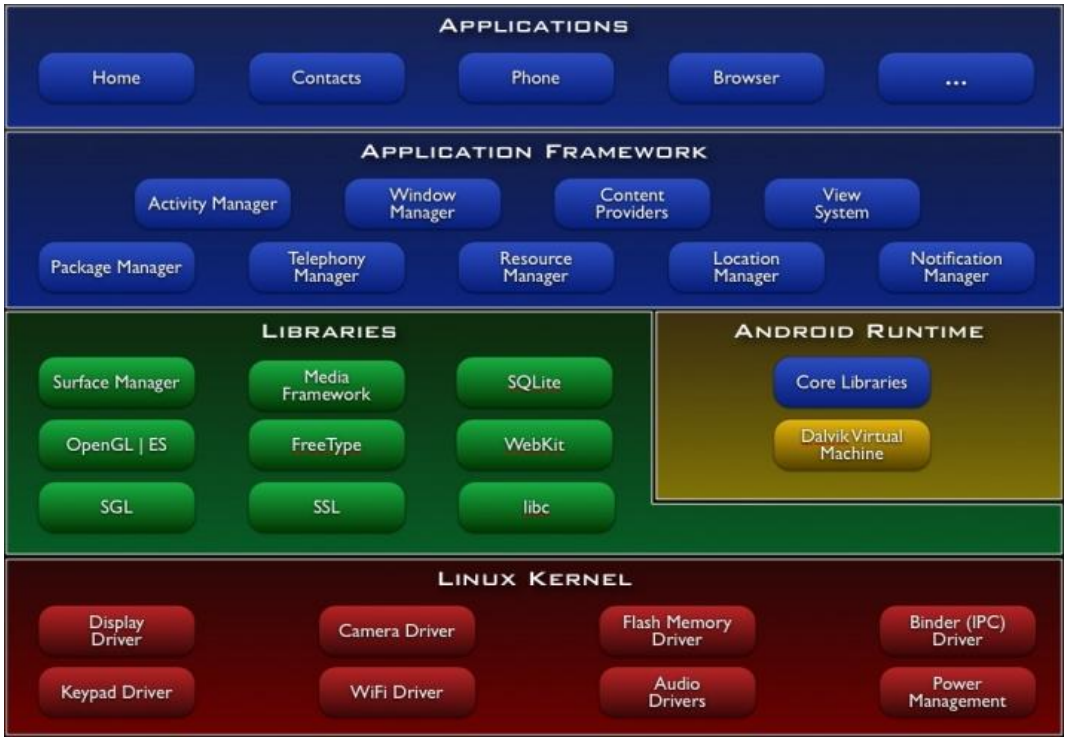

Figura 1. Quatro camadas do sistema operacional Android.

Fonte: http://www.sergioprado.org/2011/08/15/introducao-ao-funcionamento-interno-do-android/

Applications (Aplicações): é a camada que possui todas as aplicações do dispositivo. E as aplicações por sua vez são desenvolvidas em XML, ao qual mantêm a interface com o usuário. (PRADO, 2011)

Application Framework (“Quadro” de Aplicação): é a camada que interliga a camada Applications com as bibliotecas do Android (Libraries), desenvolvida

totalmente em Java, ela deixa acessível ao usuário os recursos providos pelo dispositivo. (PRADO, 2011)

Libraries (Bibliotecas): é a camada que contêm as bibliotecas do sistema, desenvolvida totalmente em C/C++, contêm: a Java Virtual Machine (Máquina Virtual Java) Dalvik, o SQLite para trabalhar com o banco de dados, e muitas outras bibliotecas para otimização do sistema operacional, e incluindo outras estão: Bionic e OpenGL/ES para trabalhar com a parte gráfica do dispositivo.(PRADO, 2011)

Linux Kernel (Micronúcleo dos Linux- Kernel): é a camada do micronúcleo do Android, aonde este utiliza o mesmo micronúcleo oriundo dos sistemas operacionais da linha Linux, portanto apenas algumas alterações. (PRADO, 2011)

2.3. Eclipse Indigo

Inicialmente o projeto Eclipse foi construído pela IBM em novembro de 2001, tendo o apoio de um consórcio de fornecedores de software. Criada em 2004, a Fundação Eclipse tinha como bandeira a criação de software de plataforma aberta, sem fins lucrativos. (ECLIPSE Foundation, 2004)

Para desenvolvimento de aplicações orientadas a objetos, é utilizada dentro da universidade a ferramenta de desenvolvimento NetBeans 6.8, diferente do projeto AdressDroid, que foi utilizado a ferramenta Eclipse.

A ferramenta NetBeans tem maior aplicabilidade na área didática, já o Eclipse possui maior aplicabilidade para softwares desenvolvidos em mercado.

O Eclipse possui um fator marcante pelo uso da biblioteca gráfica SWT ao invés de Swing. Além dele também possui maior quantidade de plugins disponíveis on-line. (ECLIPSE Foundation, 2004)

Dentre várias versões disponíveis em mercado, foi escolhido o Eclipse Indigo, pois é a última versão encontrada desta ferramenta de desenvolvimento.

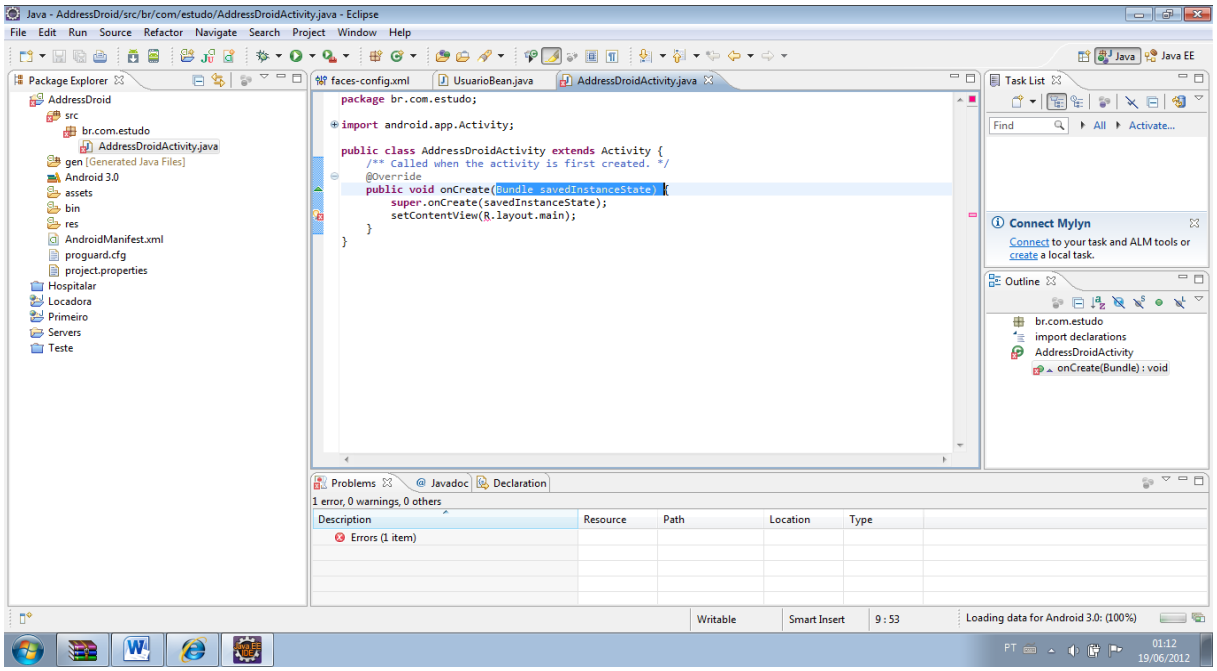

Figura 2. Tela de exemplo de desenvolvimento da aplicação AddressDroid no Eclipse.

Fonte: Ferramenta de Desenvolvimento Eclipse Indigo.

2.4. SQLite

O SQLite é uma biblioteca que possui embutido dentro de seu sistema um banco de dados SQL, que por sua vez é implementado pelo SQLite. Esta biblioteca é voltada a aplicações simples que não exige conectividade com um servidor de grande porte, pois o SQLite já é um servidor, lendo e escrevendo diretamente da aplicação para o arquivo do banco de dados em disco. (SQLITE, 2005)

Esta biblioteca trabalha com banco de dados inteiramente feito em linguagem de programação C, sendo muito utilizado para dispositivos de arquitetura embarcada e recomendado para sites com muitos acessos ou sites com aplicação cliente/servidor. (SQLITE, 2005)

Esta linguagem permite um Manager que pode ser acessado pelo navegador Firefox Mozilla. Abaixo segue imagem da visualização do script em SQLite.

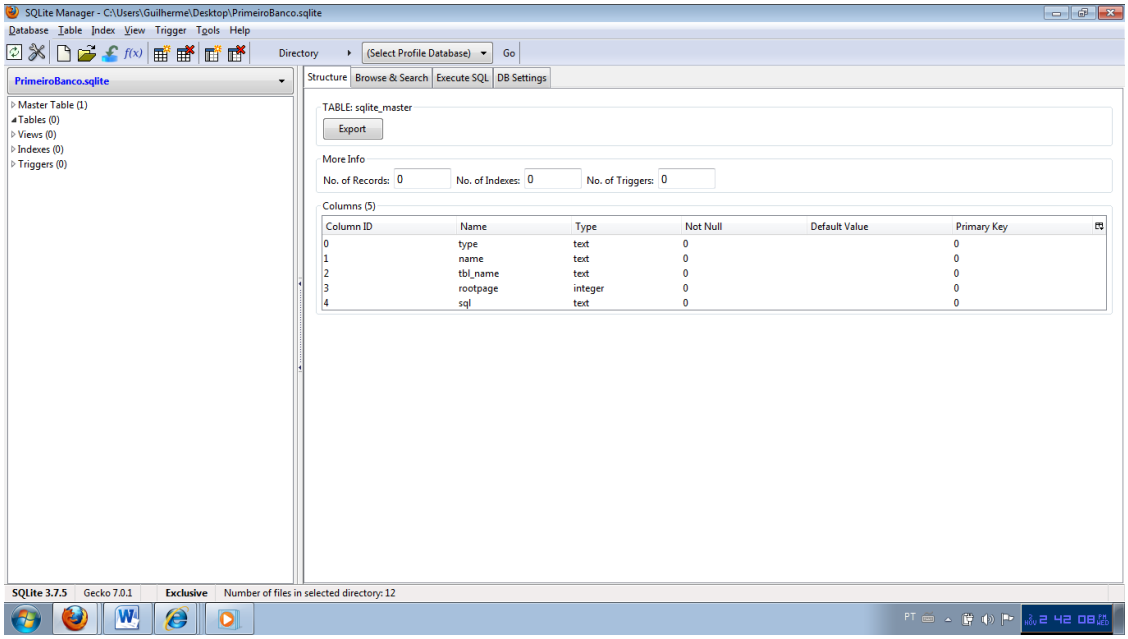

Figura 3. Exibição do SQLite Manager que é acoplado como plugin do navegador FireFox Mozilla.
Fonte: Ferramenta de Consulta SQLite.

3. Padrões de projeto

O projeto AddressDroid foi desenvolvido através do padrão de arquitetura de software Model-View-Control (MVC), utilizando-se de outros padrões, como: DAL (Data Access Layout) e BLL (Business Logic Layout) e UI (User Interface). O motivo por utilizar estes últimos padrões para o padrão MVC, foi a necessidade de facilitar a implementação dos níveis do MVC.

Como estrutura de dados, foram utilizados vetores e tabela. Os vetores surgiram da necessidade da aplicação retornar todos os clientes ou eventos, que o usuário solicita em uma consulta. E as tabelas surgiram da necessidade do banco de dados utilizar-se desta estrutura.

Em relação à consulta, foi utilizado o método de busca do próprio SGBD, pois o usuário visualizará todos os clientes ou eventos cadastrados por ele, sem a necessidade de solicitar a busca do objeto requerido.

3.1. MVC

O padrão de projetos MVC utiliza três níveis para otimização do projeto: Model, View e Controller.

O nível Model é a camada que ficará em responsável pela envio e coleta dos dados entre a aplicação e o banco de dados da aplicação, definindo assim, o modelo a ser seguido por este último. (XEROX PARC, 1978)

O nível Controller é a camada responsável pelo controle entre a interface da aplicação e o banco de dados, controlando assim os níveis Model e View.(XEROX PARC, 1978)

O nível View é a camada responsável pela interface da aplicação, em outras palavras, é o que o usuário enxerga da aplicação. (XEROX PARC, 1978)

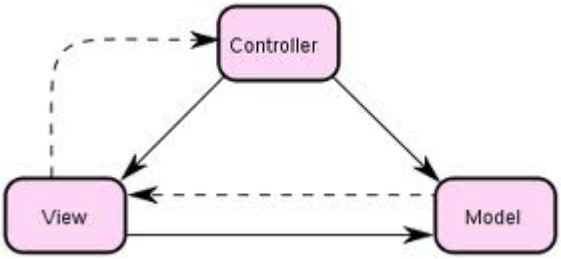

Figura 4. Diagrama em UML do padrão de projeto MVC.
Fonte: http://pt.wikipedia.org/wiki/Ficheiro:ModelViewControllerDiagram2.svg

3.1.1. UI - User Interface

A Interface com o Usuário foi o primeiro padrão de projeto utilizado no padrão de projetos MVC, que tornou prático o nível View deste mesmo padrão. Como o próprio nome diz, é a camada que interliga o software ao usuário. (AMBLER, 2003)

3.1.2. BLL - Business Logic Layout

A Camada Lógica de Negócios foi o segundo padrão de projeto utilizado no padrão de projetos MVC, que tornou prático o nível Controller. Consiste em uma camada que recebe a informação solicitada do usuário e através de um parâmetro, aponta para o índice do registro solicitado na Camada Lógica de Negócios da aplicação.(MICROSOFT Corporation, 2011)

3.1.3. DAL - Data Access Layout

A Camada de Acesso aos Dados foi o terceiro padrão de projeto utilizado no padrão de projetos MVC, onde foi inserido para tornar prático o nível Model. Por isso sua camada é a mais próxima do banco de dados da aplicação, sendo assim, ela é que será responsável pelo controle das Stored Procedures do banco de dados,

controlando a CRUD (Create, Retrieve, Update e Delete) do mesmo.(MICROSOFT Corporation, 2011)

4. AddressDroid

Para desenvolvimento de qualquer aplicação, o SDK do Android previamente inclui no código da ferramenta de desenvolvimento Eclipse a classe Java R, que é uma classe adaptadora, cuja função é unir a interface (XML) com o código da aplicação (Java); a classe Activity, que também é uma classe Java, e têm propósito instanciar e iniciar as demais telas do projeto.

A transferência da arquitetura mobile para outra arquitetura deixará este software totalmente inviável, pois só consegue ser executado nesta arquitetura.

Como foi citado anteriormente, o Android possui quatro camadas para a implementação de uma aplicação que funcione neste dispositivo, mas as camadas desenvolvidas diretamente neste projeto foram apenas três: a camada Applications para ser desenvolvida em XML, a camada Application Frameworks para ser desenvolvida em Java e a camadas Libraries para uso do banco de dados SQLite.

4.1. Banco de Dados SQLite

A estrutura do banco de dados, como tabelas e gatilhos, foram construídos na própria aplicação da agenda, sendo necessário utilizar o manager do SQLite apenas para testes. A estrutura foi implementada na classe Banco.java, adicionando a ela funções de criação de tabela e gatilho.

Diferentemente do SQL, o SQLite não utiliza a tecnologia Stored Procedure (Procedimento Armazenado). Esta tecnologia é responsável pela criação de uma rotina, dentro do próprio banco de dados, com o intuito de otimizar a aplicação, não necessitando, esta última, de criar uma rotina para tratar o que a própria Stored Procedure executará. Por outro lado o SQLite possui Trigger (Gatilho), que é uma função executa da automaticamente, antes, durante ou depois de uma ação de CRUD. Particularmente o gatilho do SQLite tem divergências de sintaxe. Como por exemplo, o SQL SERVER 2008 utiliza-se os nomes inserted e deleted para referenciar as tabelas auxiliares, já no SQLite os nomes são new e old.

Os tipos de dados que são armazenados no SQLite, por se tratar de uma biblioteca feita em C, são nomeados como: Integer, Real, Blob, Text, Numeric e Null. (SQLITE, 2005)

O tipo Null, que é um tipo de dado nulo, permite criação de tabelas com atributos nulos. Logo o tipo de dados Integer, que é um tipo de dado inteiro, permite criação de tabelas com atributos: Int, Integer, Tinyint, Smallint, Mediumint, Bigint, Unsigned Big Int, Int2 e Int8. (SQLITE, 2005)

Em seguida, os tipos de dados Real e Numeric, que são tipos de dados de ponto-flutuante, permitem tabelas com os seguintes atributos: real, Double, Double Precision, Float, Numeric, Decimal, Boolean, Date e Datetime. (SQLITE, 2005)

Já o tipo Text permite os seguintes atributos: Character, Varchar, Char, Varying Character, Nchar, Native Character, Nvarchar, Text e Clob. E por fim o tipo de dado Blob, que permite dados do tipo binários. (SQLITE, 2005)

4.1.1. DER: Diagrama de Entidade e Relacionamento

De acordo com a figura do DER abaixo, os requisitos para o projeto são:

- Entidade Cliente: Possui um código, nome do cliente, data de nascimento para que o aplicativo possa avisar a data de aniversário do cliente, email, telefone para fazer ligações para o mesmo e um código que referência o usuário que fez o cadastro deste.

- Entidade Evento: Possui um código, data e hora para o evento, descrição do mesmo e também um código que referência o usuário.

- Entidade Usuário: Também possui um código, login e senha para que o acesso aos dados seja restrito apenas a o próprio usuário.

- Relacionamento entre Usuário/Cliente e Usuário/Evento: Este relacionamento é necessário, pois cada usuário cadastrado no aplicativo terá suas respectivas informações sobre seus clientes e seus eventos. Assim um usuário poderá ter vários clientes e este cliente será somente deste usuário, e para que seja cadastrado um cliente é necessário pelo menos um usuário. O mesmo acontece com o relacionamento Usuário/Evento.

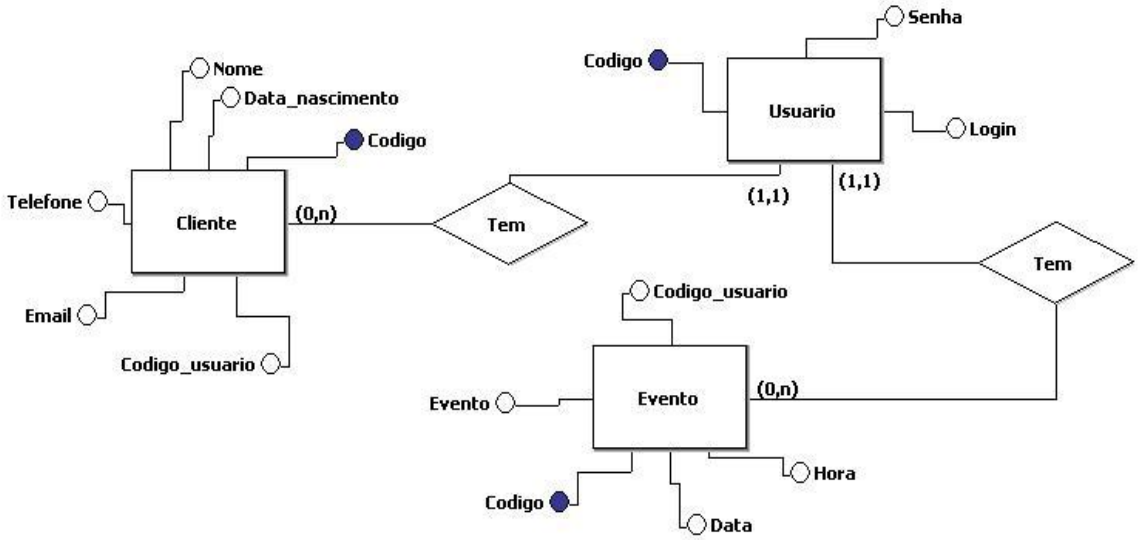

Figura 5. Diagrama entidade relacionamento do AddressDroid gerado pelo brModelo.
Fonte: Diagrama gerado pelo brModelo.

4.1.2. Modelo Relacional

A figura abaixo representa as informações da figura 5 no modelo de entidade relacional. Como se pode observar, em cada tabela é descrito os tipos de dados de cada atributo, definindo também quais atributos são obrigatórios devido à cor azul mais escuro. O vermelho representa o atributo estrangeiro que é um atributo chave de outra tabela, no caso a tabela Usuário. Os atributos chaves são os campos que possuem um desenho de uma chave amarela e são indexadas para maior velocidade e desempenho na hora da busca pelas informações.

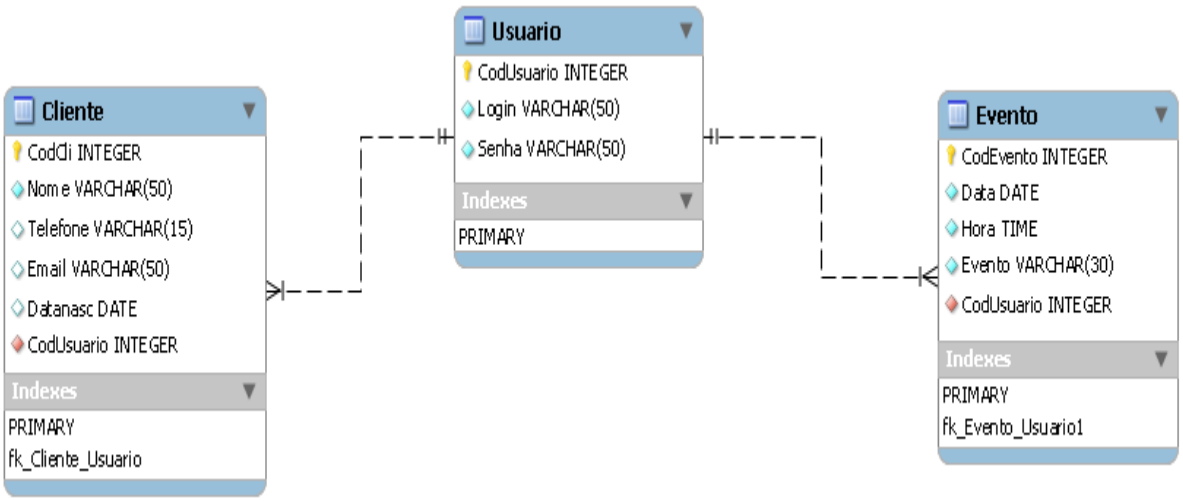

Figura 6. Modelo entidade relacionamento do banco de dados implementado no projeto AddressDroid.

Fonte: Diagrama gerado pelo MySQL Workbench 5.2.

4.1.3. Fluxograma

O fluxo de dados abaixo mostra as etapas do aplicativo. A primeira etapa é a tela de login, se já possuir um usuário cadastrado, poderá efetuar o login, caso ainda não tenha, é possível criar um novo usuário e ao sair da tela configuração de usuário já estará logado com o novo cadastro. Ambas as etapas de novo usuário ou efetuar login cairão na tela de menu. Nesta tela é possível entrar em uma tela para cadastrar os clientes, outra tela para pesquisar os mesmos ou entrar na tela onde pode ser visualizados os aniversariantes do mês. Ainda no menu, também é possível entrar para ver a agenda e nesta pode cadastrar um novo evento. Todas as telas podem voltar a qualquer momento para o menu pressionando o botão voltar.

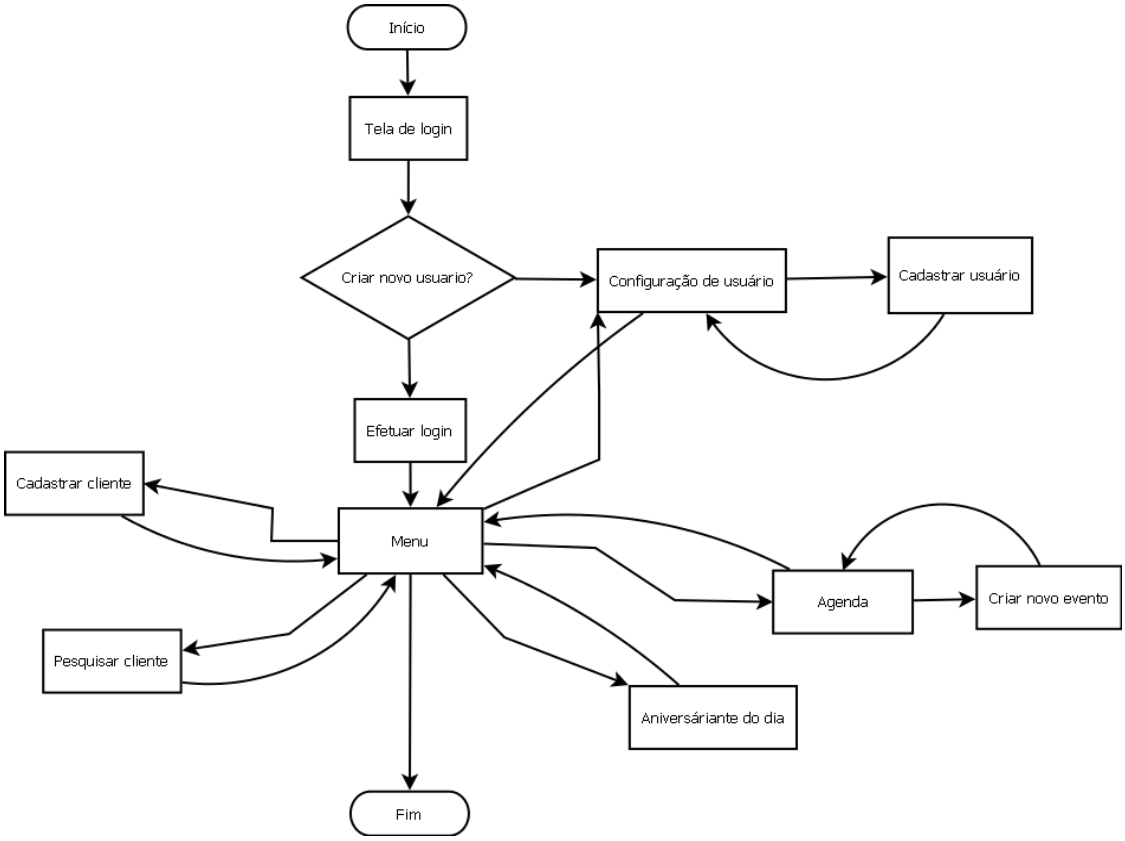

Figura 7. Fluxo de dados do AddressDroid explicando a organização das telas e processos do sistema AddressDroid.
Fonte: Fluxograma gerado pelo software Dia.

4.2. View

O projeto AddressDroid foi construído através de oito telas em XML. E estas oito telas representam o componente UI utilizado no padrão de projetos MVC, constituindo a View da aplicação. Todos os componentes destas oito classes são instanciados para a classe AddressDroidManifest.xml, que é a classe responsável

por iniciar a primeira tela. E em particular, todas elas possuem pontos funcionais, sendo necessário o usuário preencher todos os campos, quando isto for necessário.

4.2.1. Tela de Login

A primeira tela feita para o projeto foi a Tela de Login (main.xml), além do que ela é também a principal tela da aplicação, pois consiste no cadastro de usuário para acesso à agenda. Esta surgiu na necessidade de aumentar o grau de segurança do software. Sempre poderá utilizar a rotina “Primeiro acesso?” para cadastrar um novo usuário, depois de cadastrado, poderá entrar na aplicação digitando nome e senha previamente cadastrados.

Figura 8. Tela de Login
Fonte: Emulador do Android.

4.2.2. Tela de Configuração de Usuário

Para complemento da rotina “Primeiro acesso?” foi inserida a tela de Configuração de Usuário (configusuario.xml), onde será cadastrado o primeiro usuário para acessar a agenda. Nesta tela é permitido inserir uma nova conta de usuário, assim como, editar e deletar as existentes, essas sendo necessário informar a senha antiga.

Figura 9. Tela de Configuração de Usuário
Fonte: Emulador do Android.

4.2.3. Tela de Menu Principal

Realizadas as configurações de conta de acesso pelo usuário, surgirá a tela do Menu Principal (menu.xml). Esta tela contém um relógio sincronizado com o relógio do Android, onde o usuário não perderá o horário de seus compromissos. Nesta tela também cadastrar-se-á clientes e eventos relacionados aos mesmos. É possível ele configurar as contas de usuário novamente, e buscar os clientes que ele deseja visualizar; além de poder exibir os aniversariantes do dia. Deixando as vantagens desta aplicação tornarem-se visíveis em relação às aplicações existentes no Android, aonde uma única aplicação consegue substituir outras duas (Contacts e Clock).

Figura 10. Tela do Menu Principal.
Fonte: Emulador do Android.

4.2.4. Tela de Cadastro de Clientes

A Tela de Cadastro de Clientes (cadastrocli.xml), como o próprio nome diz, é o local onde o usuário cadastrará seus clientes, sendo necessário apenas inserir: nome, telefone, e-mail e data de nascimento do cliente. Todos os campos são obrigatórios para preenchimento, caso o usuário se esqueça de algum, a própria aplicação emite um alerta. Além de um novo cadastro, existe a possibilidade de alterar ou excluir os clientes existentes.

Figura 11. Tela de Cadastro de Clientes.

Fonte: Emulador do Android.

4.2.5. Tela de Pesquisa de Clientes

A Tela de Pesquisa de Clientes (pesquisacli.xml), é o local aonde a usuário visualizará seus clientes cadastro, permitir que ele delete, ligue ou envie uma mensagem para o cliente selecionado.

Figura 12. Tela de Pesquisa de Clientes.

Fonte: Emulador do Android.

4.2.6. Tela Agenda

Na Tela Agenda (proxeventos.xml) o usuário poderá agendar e visualizar os seus eventos pessoais. Quando data e horário coincidirem, o Android emitirá um alerta exibindo o nome do evento na tela do dispositivo móvel.

Figura 13. Tela Agenda.
Fonte: Emulador do Android.

4.2.7. Tela Novo Cadastro

Esta tela é uma sequencia da Tela Agenda (agendar.xml), onde o usuário cadastrará novos eventos entrando com os dados: horário do evento, data do evento e o nome do evento.

Figura 14. Tela Novo Cadastro.
Fonte: Emulador do Android.

4.2.8. Tela de Aniversariantes do Dia

Esta telinha é responsável por exibir os aniversariantes do mês, podendo o usuário enviar ao aniversariante um SMS, efetuar uma ligação ou até mesmo excluir o cliente.

Figura 15. Tela de Aniversariantes do Mês.
Fonte: Emulador do Android.

4.3. Model

Foi utilizado para camada de persistência do banco de dados a camada DAL (Data Access Layout), e através dela foram construídos os pacotes: br.gameover.addressdroid.DAL.banco, br.gameover.addressdroid.DAL.modelos, br.gameover.addressdroid.DAL.transacoes.cliente, br.gameover.addressdroid.DAL.transacoes.eventoagenda, br.gameover.addressdroid.DAL.transacoes.usuario.

4.3.1. br.gameover.addressdroid.DAL.banco

Este pacote possui a classe Banco.java, sendo esta a classe de modelagem do banco de dados, criando todas as tabelas e uma trigger.

4.3.2. br.gameover.addressdroid.DAL.modelos

Este pacote possui as classes Cliente.java, EventoAgenda.java e Usuario.java. Todas elas possuem métodos de inserção e recuperação de dados de

dentro do objeto referenciador. Logo, estes métodos são também conhecidos como métodos setters (inserção) e getters (recuperação).

4.3.3. br.gameover.addressdroid.DAL.transacoes.cliente

Este pacote possui as classes IClienteDAL.java e ClienteDAL.java. Todas as duas estão relacionadas à classe Cliente.java, tendo como função a manipulação dos dados (inserção, remoção, atualização e recuperação) entre esta classe e o banco de dados. A IClienteDAL.java é uma classe interface, cujo propósito é ser implementada pela classe ClienteDAL.java.

4.3.4. br.gameover.addressdroid.DAL.transacoes.eventoagenda

Este pacote possui as classes IEventoAgendaDAL.java e EventoAgendaDAL.java. Sendo as duas relacionadas à classe EventoAgenda.java, com o mesmo propósito e função das citadas anteriormente, tendo como único aspecto diferencial os atributo do objeto referencial.

4.3.5. br.gameover.addressdroid.DAL.transacoes.usuario

Neste pacote encontrar-se-ão as classes IUsuarioDAL.java e UsuarioDAL.java, que por sua vez, referenciam a classe Usuario.java. Diferenciam-se dos demais pacotes apenas pelo fato de possuírem atributos em seu objeto referenciador diferente dos demais, sendo suas funções idênticas às classes anteriormente citadas.

4.4. Controller

Nesta camada foram construídos os pacotes, cujo propósito é arbitrariamente controlar o fluxo de dados dentro da aplicação, não permitindo dados dispersos ou inválidos que prejudiquem a otimização da aplicação. Para mantimento das regras de negócios da aplicação, os pacotes são:

- br.gameover.addressdroid.BLL.CaracterEspecial,
- br.gameover.addressdroid.BLL.cliente,
- br.gameover.addressdroid.BLL.eventoagenda,
- br.gameover.addressdroid.BLL.usuario.

4.4.1. br.gameover.addressdroid.BLL.CaracterEspecial

Este pacote possui apenas uma classe, a CaracterEspecial.java, cujo propósito é impedir que o usuário entre com caracteres que são considerados inválidos para a aplicação Android.

4.4.2. br.gameover.addressdroid.BLL.cliente

Respeitando a estrutura da camada DAL, a mesma foi aplicada a esta camada de negócio, constituindo-se pela classe interface IClienteBLL.java,que é implementada pela classe ClienteBLL.java. Já as suas funções diference das classes DAL, pois agora, mesmo estas também referenciando a classe Cliente.java, elas inibirão as dados inválidos para a aplicação, através de estruturas de seleção.

4.4.3. br.gameover.addressdroid.BLL.eventoagenda

Logo as classes IEventoAgendaBLL.java e EventoAgendaBLL.java, contidas neste pacote, têm as mesmas funções das anteriormente citadas, diferenciando apenas a classe referencial, sendo nesta a EventoAgenda.java.

4.4.4. br.gameover.addressdroid.BLL.usuario

As classes IUsuarioBLL.java e UsuarioBLL.java, possuem o propósito de inibir os dados considerados anômalos pela aplicação Android. Nisto sua classe referenciada Usuario.java manterá sempre integridade dos dados recebidos e enviados.

Considerações Finais

Este artigo apresentou o projeto AddressDroid, cuja aplicação é executada através do sistema operacional Android. Foi citada a estrutura do projeto, com todas as suas classes de variadas linguagens de programação, assim como, os padrões de projeto utilizados para a construção do mesmo. Contudo não foram feitos testes com usuários devido à escassez de tempo, mas pode ser que futuramente isso venha a ser realizado.

Conclui-se que o ato de inovar o software, projetando em uma arquitetura mobile, traz bastantes desafios quanto à implementação deste, de acordo com a dificuldade que se teve de desenvolver em uma linguagem orientada a objetos para dispositivos embarcados que consiste em pouco poder de processamento, menor quantidade de memória e em relação ao desenvolvimento, dificuldades na sintaxe do SDK (Framework para desenvolvimento Android) que é nova em relação ao ensinado na universidade.

Referências

AMBLER, Scott W. 2003. Agile Modeling. *http://www.agilemodeling.com.* [Online] Agile Modeling, 2003. [Citado em: 08 de Novembro de 2011.] http://www.agilemodeling.com/artifacts/uiPrototype.htm.

CAVALLINI, Ricardo, XAVIER, Léo e SOCHACZEWSKI, Alon. 2010.*#Mobile.* São Paulo : dos Autores, 2010. 978-85-908688-3-5.

ECLIPSE Foundation. 2004. www.eclipse.org. *Eclipse - The Eclipse Foundation open source Community Website.* [Online] Oracle Corporation, 15 de novembro de 2004. [Citado em: 2 de novembro de 2011.] www.eclipse.org/org.

MICROSOFT Corporation. 2011. www.asp.net. *Microsoft Asp.net.* [Online] Microsoft Corporation, 2011. [Citado em: 08 de Novembro de 2011.] http://www.asp.net/data-access/tutorials/creating-a-business-logic-layer-vb.

MICROSOFT Corporation. 2011. www.asp.net. *Microsoft Asp.net.* [Online] Microsoft Corporation, 2011. [Citado em: 08 de Novembro de 2011.] http://www.asp.net/data-access/tutorials/creating-a-data-access-layer-vb.

PRADO, Sergio. 2011. www.sergioprado.org. *Sergio Prado.org.* [Online] Embedded Labworks, 15 de agosto de 2011. [Citado em: 1 de novembro de 2011.] http://www.sergioprado.org/2011/08/15/introducao-ao-funcionamento-interno-do-android/.

SQLITE. 2005. SQlite. *www.sqlite.org.* [Online] SQlite, 19 de Fevereiro de 2005. [Citado em: 20 de Novembro de 2011.] http://www.sqlite.org/about.html.

SQLITE. 2005. SQLITE. *www.SQlite.org.* [Online] SQlite, 19 de Fevereiro de 2005. [Citado em: 20 de Novembro de 2011.] http://www.sqlite.org/datatype3.html.

XEROX PARC. 1978. http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html. *MVC.* [Online] Xerox Parc, 1978. [Citado em: 08 de Novembro de 2011.] http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html.