Coordenadoria do Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

Replicação de dados aplicada à um dispositivo móvel: um estudo de caso

Jean Carlo Wai Keung Ma e Rodrigo Hideki Yamashita

Nilton Cézar de Paula (Orientador)

Novembro de 2013

Replicação de dados aplicada à um dispositivo móvel: um estudo de caso

Jean Carlo Wai Keung Ma e Rodrigo Hideki Yamashita

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Jean Carlo Wai Keung Ma e Rodrigo Hideki Yamashita e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 22 de novembro de 2013.

Prof. Dr. Nilton Cézar de Paula (Orientador)

Resumo

O crescimento no desenvolvimento tecnológico proporcionou um aumento na popularização dos dispositivos móveis. A facilidade de acesso às informações é outro fator que influenciou na popularização desses acessórios. É através de redes sem fios que esses dispositivos fazem parte da rede, porém, não é em todos os locais que essas redes existem. Para suprir a necessidade de conexão a todo instante para obter acesso aos dados, neste trabalho será apresentado um estudo sobre replicação de dados aplicada a dispositivos móveis, além de apresentar conceitos sobre o sistema operacional móvel *Android* e os tipos de replicação que existem na literatura. Por fim, é apresentado o desenvolvimento de uma aplicação, de forma detalhada, para o sistema operacional *Android* que se utiliza do conceito de replicação de dados para suprir essa falta de conexão e possuir acesso às informações.

Palavras-chave: *Computação Móvel, Android, Replicação Assíncrona.*

Abstract

The growth in technological development provided an increase in the popularization of a mobile devices. The facility of access to data is another factor that influenced the popularization of these devices. It is through wireless network the these devices are part of the network, however, it is not everywhere that these networks exists. To supply the need for connection all the time for get access to data, this paper will be presented a study of data replication applied to mobile devices, and present concepts about the Android mobile operating system and techniques of data replication in the literature. Finally, it is presented the development of an application in detail, for Android operating system which uses the concepts of data replication to supply this lack of connection and have access to data.

Key-words: *Mobile Computing, Android, Asynchronous Replication.*

Agradecimentos

Aos meus pais, Ma Yick On e Kwok Wing Hung Ma, pelo apoio e incentivo que deram durante a minha graduação.

À minha irmã, Regiane Tu Kun Ma, por esses 4 anos que moramos juntos longe dos nossos pais.

Ao Professor Nilton Cézar de Paula pela paciência, compreensão e orientação nesses anos de graduação.

Ao meu amigo e colega de turma, Rodrigo Hideki Yamashita, que me ajudou a concluir mais uma etapa da minha graduação.

Às pessoas que contribuíram diretamente ou indiretamente a esse trabalho.

Jean Carlo Wai Keung Ma

Aos meus pais por terem incentivado, acreditado e investido nos meus estudos, por estarem do meu lado nos momentos difíceis, apoiando e dando forças para concluir mais essa etapa da minha vida.

Ao Professor Nilton Cézar de Paula que aceitou ser o meu orientador e por ter tido paciência e dedicação comigo.

Ao meu colega de curso e grande amigo Jean Carlo Wai Keung Ma, que juntos debatemos uma melhor solução para o problema e conseguimos concluir com sucesso nosso trabalho de conclusão de curso.

E a todas as pessoas que de alguma forma colaboraram para que eu conseguisse concluir mais essa etapa da minha vida e me formar.

Rodrigo Hideki Yamashita

Sumário

Lista de Siglas

ADB - Android Debug Bridge

ADT - Android Development Kit

API - Applications Programming Interface

AVD - Android Virtual Device

DVM - Dalvik Virtual Machine

FCC - Federal Communications Commission

GPS - Global Positioning System

IDE - Integrated Development Environment

IEEE - Institute of Electrical and Eletronics Engineers

JDK - Java Development Kit

JVM - Java Virtual Machine

LAN - Local Area Network

OHA - Open HandSet Alliance

PDA - Personal Digital Assistant

QVGA - Quarter Video Graphics Array

SDK - Software Development Kit

SGBD - Sistema de Gerenciamento de Banco de Dados

SQL - Structured Query Language

WAN - Wide Area Network

XML - eXtensible Markup Language

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

No decorrer das últimas décadas a tecnologia teve uma grande evolução, e no que se diz respeito à computação, surgiu a computação móvel, que pode ser considerada como um novo paradigma computacional (FIGUEIREDO; NAKAMURA, 2003). A computação móvel permite que os usuários acessem informações e se comuniquem a qualquer instante e em qualquer lugar. Com isso os dispositivos móveis ganharam um grande espaço no mundo, em outras palavras ele se tornou essencial na vida do ser humano.

A todo instante, novos *softwares* são desenvolvidos utilizando as tecnologias oferecidas pela época, tais como: câmera com mais qualidade, processadores mais potentes, função *touchscreen* e entre outros. Um dos fatores que revolucionaram a computação móvel foi a possibilidade de conexão com a Internet em diferentes pontos de acesso.

A maioria dos softwares faz uso da Internet, sendo para buscar atualizações, baixar aplicativos ou entrar em redes sociais, dentre outros. Para alguns a conexão com a Internet não é extremamente necessária. Porém, para outros *softwares* a falta de conexão o torna obsoleto, que é o caso dos *softwares* para redes sociais, ou até mesmo de um GPS que só funciona com Internet. A idéia deste trabalho é fornecer acesso aos dados para os usuários que não possuam conexão com a Internet.

1.1 Objetivos

Este trabalho tem como objetivo geral um estudo sobre o sistema operacional móvel *Android* e a replicação de dados em um dispositivo móvel.

1.1.1 Objetivos específicos

Os objetivos específicos são:

- Pesquisar sobre a computação móvel;
- Pesquisar sobre as características e arquitetura do sistema operacional móvel *Android*;
- Pesquisar sobre as técnicas de replicação de dados existentes na literatura;
- Implementar um aplicativo que realiza a replicação de dados em um dispositivo móvel;
- Testar os softwares desenvolvidos com o intuito de verificar o correto funcionamento de sua execução.

1.2 Justificativa e Motivação

Mesmo com o crescimento dos números de redes sem fio, existem regiões onde os dispositivos não possuem conexão com a Internet, o que torna algumas aplicações indisponíveis para seus usuários. A motivação é tornar estes dados disponíveis a fim de que o usuário final possa usufruir da aplicação sem a necessidade direta de uma conexão.

1.3 Metodologia

Para a realização dos estudos sobre computação móvel, sistema operacional *Android* e replicação de dados foram consultados livros, revistas, artigos, teses e dissertações, existentes em bibliotecas digitais.

A forma de comunicação entre a aplicação móvel e a servidor é realizada através de *sockets* de comunicação.

Para a implementação da aplicação móvel foi escolhido o sistema operacional *Android* 4.3 com API 18, por ser *open source* e possuir bastante documentação sobre sua utilização, utilizado as ferramentas Android SDK e Eclipse IDE. A aplicação servidor e a auxiliar foram desenvolvidas na linguagem JavaSE, utilizando o NetBeans e Eclipse IDE e o SGBD MySQL, e para a visualização dos dados o MySQL Workbench.

Os testes para a aplicação móvel foram realizados com o emulador Android dispositivo: 2.7" QVGA (240x320: ldpi), RAM: 768 MiB e Android 4.3 - API 18.

1.4 Organização do texto

O texto do projeto está organizado em um único volume. Além desse Capítulo, o volume é organizado em outros 5 capítulos e 7 apêndices, cujos conteúdos são sumarizados a seguir.

Capítulo 2

Computação Móvel

No capítulo 2 introduz alguns conceitos e fatos tecnológicos ocorridos que possibilitaram que as pessoas acessassem informações utilizando algum dispositivo móvel. Além de apresentar algumas limitações desses dispositivos.

Capítulo 3

Sistema Operacional Android

No capítulo 3 apresenta-se um estudo geral sobre a arquitetura do sistema operacional, versões e movimentação do mercado com a venda de aplicativos.

Capítulo 4

Replicação de dados

No capítulo 4 desenvolveu-se um estudo sobre as diferentes técnicas de replicação de dados.

Capítulo 5

Estudo de caso

No capítulo 5 é apresentado o estudo de caso com a implementação de um sistema com replicação de dados em uma aplicação móvel e os testes realizados.

Capítulo 6

Considerações Finais

No capítulo 6 apresenta-se a conclusão do trabalho.

Apêndice A

Instalação do NetBeans IDE no Windows 7

No apêndice A apresenta-se a instalação do NetBeans IDE no sistema operacional Windows 7.

Apêndice B

Instalação do Eclipse IDE no Windows 7

No apêndice B apresenta-se a instalação do Eclipse IDE no sistema operacional Windows 7.

Apêndice C

Instalação do Android SDK no Windows 7

No apêndice C apresenta-se a instalação do Android SDK no sistema operacional Windows 7.

Apêndice D

Classes implementadas no servidor

No apêndice D apresenta os códigos-fonte das principais classes utilizadas pelo servidor.

Apêndice E

Classes implementadas no cliente Android

No apêndice E apresenta os códigos-fonte das classes utilizadas pela aplicação cliente.

Apêndice F

Instalação do MySQL no Windows 7

No apêndice F apresenta-se todos os passos necessários para a instalação do MySQL e MySQL Workbench no Windows 7.

Apêndice G

Códigos utilizados na criação do banco de dados

No apêndice I apresenta os códigos na sintaxe SQL utilizados na criação e povoamento das tabelas do banco de dados.

Capítulo 2

Computação Móvel

Nesta seção serão abordados os conceitos e principais motivos do surgimento da computação móvel. Este capítulo está dividido em 4 seções. A seção 2.1 apresenta uma breve introdução sobre a computação móvel. Na seção 2.2, aborda a evolução que os dispositivos móveis tiveram. Na seção 2.3 apresenta as principais tecnologias de rede sem fio. Por fim, na seção 2.4, as limitações desses dispositivos.

2.1 Introdução

A computação móvel foi um fator que revolucionou o mundo da tecnologia, ela possibilitou que as pessoas pudessem acessar informações independentemente da sua localização e a qualquer instante, utilizando apenas algum dispositivo móvel. Em outras palavras, computação móvel nada mais é do que uma junção dos fatores de mobilidade, processamento e comunicação sem fio (FIGUEIREDO; NAKAMURA, 2003).

Por proporcionar mais mobilidade e ser de grande utilidade, cada vez mais as pessoas estão fazendo uso da computação móvel através de algum tipo de dispositivo móvel, tal como PDAs, laptops, palmtops e celulares (FIGUEIREDO; NAKAMURA, 2003). Logo, com essa evolução, a necessidade das pessoas de terem esse “conforto” e mobilidade que a computação móvel proporciona, fez com que praticamente todas as pessoas hoje em dia possuam algum tipo de dispositivo móvel (AQUINO, 2007).

Computação móvel foi e é uma grande evolução no mundo da tecnologia. Cada vez mais são desenvolvidos aplicativos e novas funcionalidades que oferecem mais disponibilidades de serviços aos usuários.

Na computação móvel também entra o conceito de computação distribuída, pois por ela possuir comunicação sem fio, ela elimina a necessidade do usuário de permanecer conectado em algum aparelho fixo (FIGUEIREDO; NAKAMURA, 2003).

É por esses e por outros motivos que grandes empresas estão investindo pesado na construção desses dispositivos móveis, assim como no desenvolvimento de sistemas operacionais que ofereçam mais, conforto, rapidez e qualidade ao usuário. Um exemplo de sistema operacional é o *Android* (AQUINO, 2007).

2.2 Evolução dos dispositivos móveis

O primeiro telefone móvel foi inventado cerca de 100 anos depois da criação do primeiro telefone. O principal objetivo era oferecer uma comunicação à distância. Porém devido à tecnologia precária da época, o primeiro telefone móvel analógico foi criado em 1960 e era grande demais e impossibilitava as pessoas de o carregarem pessoalmente, o que fez com que eles fossem instalados em veículos. Após 13 anos, em 1973, foi inventado o primeiro telefone móvel de mão. E somente em 1977 foram criadas as primeiras redes celulares digitais (CRUZ, 2012).

Antigamente, como foi mencionado, os telefones móveis tinham apenas o objetivo de oferecer comunicação à distância e dar mobilidade, porém hoje em dia os telefones móveis, assim como outros dispositivos móveis, dispõem da capacidade de processamento, e isso faz com que o conceito de computação móvel seja adicionada aos telefones móveis e dispositivos móveis (CRUZ, 2012).

A partir desse ponto, telefones e dispositivos móveis tiveram uma vasta evolução onde novas funcionalidades eram desenvolvidas a todo instante e com isso a computação móvel precisava acompanhar essa evolução com um sistema operacional mais complexo que oferecesse mais rapidez e disponibilidade de recursos ao usuário. Vários sistemas operacionais foram criados, um deles é o *Android,* que será abordado mais adiante (PESSOA, 2013).

2.3 Principais tecnologias de redes sem fio

Para que dispositivos móveis possam trocar informações é necessário que haja uma rede que faça essa ligação entre os dispositivos. Então serão abordadas duas principais redes sem fio que vem sendo muito utilizadas que são: o Padrão IEEE 802.11 e o *bluetooth* (FIGUEIREDO; NAKAMURA, 2003).

2.3.1 Padrão IEEE 802.11

O Padrão IEEE 802.11, também conhecido como rede Wi-Fi ou *Wireless*, foi criado pela IEEE (*Institute of Electrical and Eletronics Engineers*), visando substituir as estruturas de conexão cabeada (FIGUEIREDO; NAKAMURA, 2003). Então em 1990 a IEEE criou o Projeto Padrão IEEE 802.11, porém devido à tecnologia escassa que era oferecida na época o projeto ficou só no papel durante sete anos (GARCIA, 2003). Um exemplo da escassa tecnologia da época que impedia o projeto de sair do papel era a baixa taxa de transferência oferecida, em torno de Kbps (GARCIA, 2003).

Após sete anos, foi aprovada a IEEE 802.11 que dessa vez contava com uma taxa de transferência de 1 a 2 Mbps. Com o passar do tempo essa taxa de transferência começou a crescer bastante, o que fez com as empresas começassem a investir mais na construção de equipamentos de comunicação sem fio, pois devido a esse crescimento da taxa de transferência, essa tecnologia começou a se tornar algo promissor para as empresas (GARCIA, 2003).

Atualmente, essa tecnologia é uma das mais usadas, pois ela oferece uma grande taxa de transferência e é excelente para locais onde não é viável utilizar a infraestrutura cabeada.

2.3.2 Bluetooth

O *bluetooth* é um padrão de comunicação sem fio de baixo custo e curto alcance que utiliza ondas de rádio para realizar a comunicação entre os aparelhos como computadores, celulares, impressoras, dentre outros (FERREIRA; PEREIRA; OLIVEIRA; FÉO, 2005). A ideia foi originada pela empresa Ericsson em 1994 com o intuito de substituir os cabos que eram utilizados na conexão de dispositivos portáteis ou fixos com outros acessórios existentes (STEIN, 2003). Então após quatro empresas como, IBM, Intel e Toshiba se uniram a Ericsson formando a

SIG (*Special Interest Group*), para desenvolver um padrão que seria chamado de Bluetooth (STEIN, 2003). Hoje em dia a SIG conta uma rede de mais de 2000 empresas.

Para a comunicação entre os aparelhos, é necessário que os mesmo tenham rádios embutidos em chips, para que possam se comunicar através de ondas de rádios, utilizando uma tecnologia chamada "transmissão por salto de frequência", que além de ser confiável ela também é a prova de interferência (FIGUEIREDO; NAKAMURA, 2003).

2.4 Limitações dos dispositivos móveis

Apesar de apresentarem uma série de vantagens, os dispositivos móveis também possuem várias desvantagens. Um bom exemplo é a energia, ou seja, todo aparelho possui sua própria bateria que em geral não são muito duráveis, limitando o usuário a carregar sua bateria constantemente (FIGUEIREDO; NAKAMURA, 2003).

A interface gráfica é outro problema da computação móvel que torna a interação do usuário com cada dispositivo diferente, ou seja, a interface gráfica varia de dispositivo para dispositivo, sem possuir um teclado ou mouse padrão, tornando a interação diferente (FIGUEIREDO; NAKAMURA, 2003).

Além das limitações físicas encontradas nesses dispositivos temos as limitações de *software*. Sistemas operacionais móveis pagos são as maiores restrições para o usuário, não oferecendo permissão alguma para modificar o sistema. Entretanto, o grupo Open Handset Alliance liberou o sistema operacional móvel *Android* com o objetivo de acabar com essas limitações. Através do *Android*, o usuário é capaz de modificar seu próprio aparelho, adicionar funções e criar seus aplicativos.

Capítulo 3

Android

O sistema operacional móvel *Android* foi desenvolvido pelo grupo *Open Handset Alliance* (OHA), e é uma plataforma completa para dispositivos móveis que incluem um sistema operacional, *middleware*, aplicativos e uma *interface* do usuário (OHA-AND-REV, 2007).

Apesar do lançamento do sistema operacional ocorrer no final de 2007 (OHA-ANN-*ANDROID*, 2007), o T-Mobile G1, conhecido como HTC-Dream, foi o primeiro dispositivo integrado com o *Android* a ser aprovado pela FCC[1] em 18 de agosto de 2008 (RICKER, 2008). No entanto, a HTC só anunciou oficialmente a aprovação do aparelho em 23 de setembro de 2008 (T-MOBILE, 2008), e o produto começou a ser vendido apenas em 22 de outubro de 2008 (HTC-G1, 2012).

Este capítulo está dividido em 4 seções. A seção 3.1 abordará sobre o grupo *Open Handset Alliance*. A seção 3.2 apresenta informações sobre a arquitetura deste sistema operacional móvel. Na seção 3.3 abordará sobre o mercado de aplicativos *Android*. Por fim, na seção 3.4, abordará as diferentes versões do *Android*.

3.1 Open Handset Alliance

O grupo denominado *Open Handset Alliance* foi criado em 5 de novembro de 2007 (OHA-ANN-ANDROID, 2007) e foi formado por diversas empresas do ramo tecnológico e de telefonia móvel que se uniram com o objetivo de revolucionar o mercado móvel. Os objetivos deste grupo era trazer para seus consumidores dispositivos melhores, proporcionar aos usuários uma experiência móvel mais rica e mais barata financeiramente. Atualmente o grupo é liderado pela empresa Google.

Assim como a criação do grupo *Open Handset Alliance*, apresentou-se juntamente nesta data a plataforma *open source Android* baseada no sistema operacional *Linux* (OHA-AND-REV, 2007). Entretanto o projeto iniciou em 2003 através de Andy Rubin (KOVACH, 2013), e então

[1]http://www.fcc.gov/what-we-do

em 2005 a *Google* adquiriu este projeto (BEAVIS, 2008). A intenção da OHA era criar um sistema operacional de código aberto, permitindo assim que este seja integrado e customizado pelos usuários em qualquer aparelho celular, da mesma forma que acontece com os computadores.

3.2 Arquitetura

A *Open Handset Alliance* refere-se ao *Android* como sendo uma grande pilha de *software*, a base dessa arquitetura é o *Linux Kernel*, acima se encontra as camadas das bibliotecas nativas e o *Android Runtime*. Em seguida, aparece a camada de *frameworks* para desenvolver os aplicativos e, por fim, as aplicações do sistema operacional. A arquitetura pode ser observada na **Figura 3.1**:

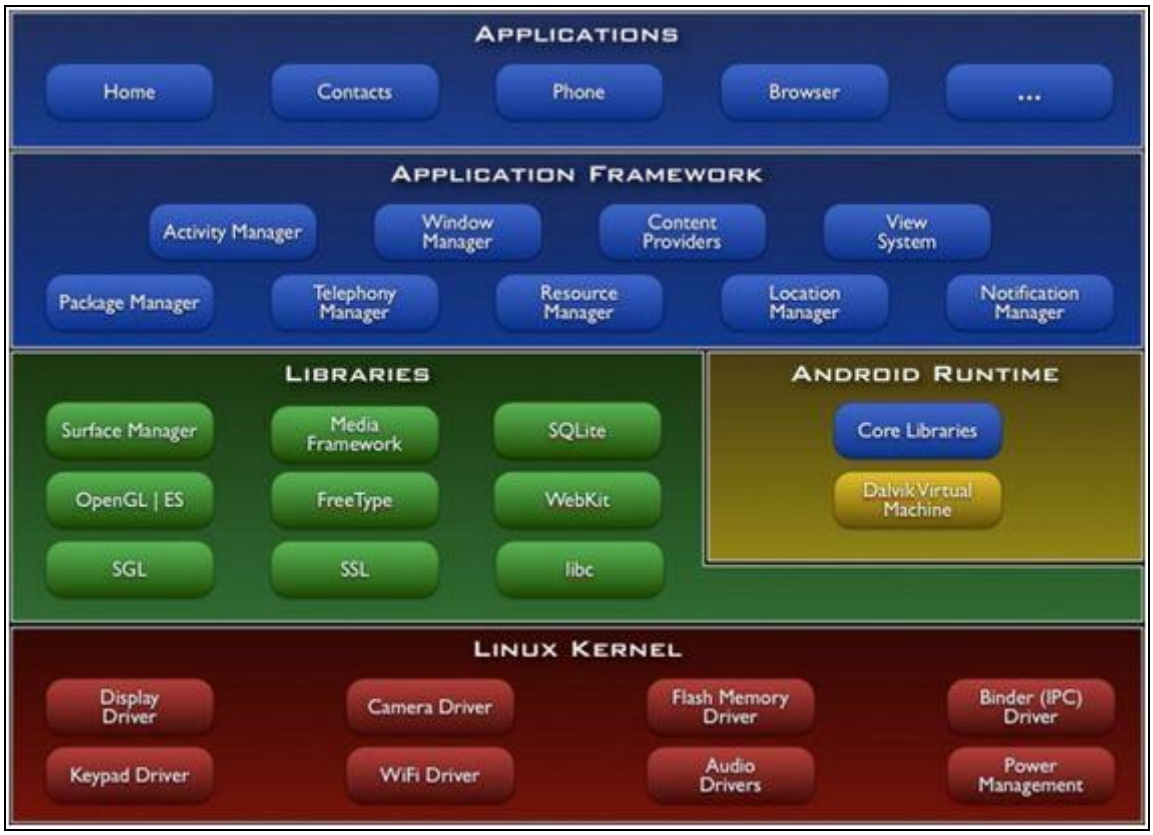


Figura 3.1: Arquitetura do *Android* (SILVA, 2012, p9).

3.2.1 Linux Kernel

Como fora colocado anteriormente, o *Android* utiliza o *Kernel* do *Linux* como base de sua arquitetura. O *kernel* inicial a ser utilizado foi o da versão 2.6 do *Linux*, porém esse *kernel* foi modificado para que atenda melhor as reais necessidades e as limitações dos dispositivos móveis, como o gerenciamento de energia e de memória (BORDIN, 2012).

No repositório do *Android* (AND-GIT-REP, 2013) são encontrados diversos *patches* de atualização que dão suporte a diferentes dispositivos, entre eles, o sistema-em-um-chip (SoC) que são componentes de um computador em um circuito integrado.

3.2.2 Libraries

A camada de bibliotecas é formada por um conjunto de bibliotecas em C/C++ utilizadas pelo sistema e outras bibliotecas que dão suporte aos arquivos de áudio e vídeo, além de possuir suporte a ambientes gráficos em 2D e 3D e a um banco de dados relacional, o SQLite.

Nessa camada, as bibliotecas são responsáveis pelo modo que os dispositivos irão manipular certos tipos de dados, dessa maneira, uma biblioteca do *framework* de mídia é responsável pela gravação e reprodução de arquivos de áudio e vídeo (STRICKLAND, 2007).

As bibliotecas encontradas nesta camada são descritas a seguir:

- *FreeType* – Biblioteca utilizada para renderização de fontes de texto para o formato *bitmap* (FREETYPE, 2013).
- Libc – Derivação da biblioteca padrão do C para sistemas embarcados e com licença BSD (PEREIRA; SILVA, 2009, p7).
- Media *Framework* – Biblioteca de suporte para diversos tipos de mídia, como áudio, vídeo e imagem (AND-DEV-MEDIA, 2013).
- *OpenGL* | ES – Biblioteca de suporte para gráficos de alto desempenho em 2D e 3D destinado a sistemas embarcados. A versão 1.0 e 1.1 do *OpenGL* ES suporta a versão *Android* 1.0 ou superior. O *OpenGL* ES 2.0 suporta as versões 2.2 ou superior (API level 8). Enquanto que a versão 3.0 destina-se ao *Android* 4.3 ou superior (API level 18) *(AND-DEV-OPENGL*, 2013).
- SGL – Biblioteca gráfica 2D (AND-SRC-GRA, 2013).

- SQLite – Biblioteca compacta de código aberto escrita em C que implementa um banco de dados SQL (SQLITE, 2013).
- SSL – Protocolos de comunicação segura entre cliente e servidor (*AND-DEV-SEC*, 2013).
- *Surface Manager* – Biblioteca que gerencia o acesso ao *display* e compõem as camadas gráficas 2D e 3D das aplicações (SYMLAB, 2013).
- WebKit – É uma biblioteca de código aberto com um mecanismo de renderização de páginas da internet utilizado pelos navegadores (WEBKIT, 2013).

3.2.3 Android Runtime

Na camada de tempo de execução do *Android*, encontra-se um conjunto das principais bibliotecas da linguagem Java e a máquina virtual Dalvik (DVM), devido às limitações apresentadas pelos dispositivos móveis, a Dalvik é configurada para oferecer melhor desempenho em ambientes móveis (PEREIRA; SILVA, 2009, p.8).

O modo de funcionamento da máquina virtual Java (JVM) convencional difere-se da Dalvik, a JVM compila um código fonte em Java e gera um novo arquivo chamado de *bytecode* com extensão "*.class*", em seguida, converte o *bytecode* em código executável de máquina.

Na DVM funciona de modo semelhante até o processo de criação do *bytecode*. A partir daí, a DVM utiliza a ferramenta DX, embutida no SDK *Android* (*AND-DEV-TOOLS*, 2013), que transforma o *bytecode* em arquivo com formato ".dex", que então pode ser interpretada pela máquina virtual Dalvik.

Quando a JVM interpreta os códigos do Java, é gerada apenas um arquivo ".class" para cada classe compilada. Os arquivos ".dex" contém várias classes que resultam na redução do tamanho do arquivo e na quantidade de operações de entrada e saída (EHRINGER, 2013).

O *Android* possui a característica de que cada aplicação tem seu próprio processo *Linux*. Os processos deste sistema operacional possuem sua própria máquina virtual, através disso, as aplicações torna-se isoladas uma das outras, o que faz com que não afete outros processos e ajuda a liberar memória para outras aplicações. Entretanto, existe a possibilidade de compartilhar informações entre os aplicativos através de uma solicitação de acesso aos dados (*AND-DEV-FUN*, 2013).

As bibliotecas principais (*Core Libraries*) localizadas nessa camada são compostas por grande parte das funcionalidades disponíveis no *Java Standard Edition* a fim de tornar o ambiente de programação familiar ao da linguagem Java para que os desenvolvedores pudessem criar seus aplicativos (MILK, 2013).

3.2.4 Application Framework

Esta camada possui todos os recursos e funções que estão disponíveis e documentados para que os desenvolvedores tirem proveito de todas as funcionalidades que o *Android* pode oferecer (GARGENTA, 2011, p.11).

3.2.5 Applications

Nessa última camada encontram-se todos os aplicativos que estão presentes no *Android*, isso inclui as aplicações nativas e as que foram desenvolvidas por terceiros e compradas pelos usuários acessando o *Google Play* (GARGENTA, 2011, p.12).

3.3 Mercado de aplicativos

A *Google Play* é principal loja de aplicativos para *Android*, foi lançada em 6 de março de 2012 através da união dos aplicativos *Android Market*, *Google Music* e *Google eBookstore* (GOOGLEBLOG, 2012). Segundo (*GOOGLEPLAY*, 2013), a loja possui mais de 700.000 (setecentos mil) aplicativos de todas as categorias disponíveis e até 2012, já foram baixados mais de 25.000.000.000 (vinte e cinco bilhões) de aplicativos. O aplicativo esta pré-instalado em mais de 400.000 (quatrocentos mil) aparelhos espalhados pelo mundo, então, ao publicar na loja, a probabilidade de encontrar algum cliente é muito alta. A *Google Play* também está presente em mais de 130 países (*AND-DEV-VIS*, 2013).

Quando um desenvolvedor resolve publicar seu trabalho, ele tem a opção de categorizar seu aplicativo a fim de facilitar as buscas do usuário. Além disso, é possível realizar buscas com termos sugestivos para procurar o aplicativo desejado (*AND-DEV-VIS*, 2013).

A *Google Play* possui um sistema na qual os usuários podem avaliar os aplicativos baixados e comentar a experiência que obtiveram, assim os desenvolvedores e outros usuários

podem saber quais pontos fracos e fortes daquela aplicação pelo ponto de vista dos consumidores (*AND-DEV-VIS*, 2013).

3.4 Versões

O propósito do *Android* era de ser aberto para os desenvolvedores e fabricantes que desejassem tornar suas ideias em realidade e melhorar a experiência dos usuários com os dispositivos móveis. A *Google* é a empresa responsável pelo projeto, porém aceita a contribuição de toda a comunidade e mantém uma equipe para gerenciar o projeto (*AND-SRC-CODE*, 2013).

Como um projeto de alto nível, o *Android* passou por várias modificações em um curto espaço de tempo. Abaixo é possível ver a evolução do *Android* 1.0 até a versão 4.3.

Tabela 3.1: Versões do Android.

Codenome	**Versões**
(Nenhum)	1.0 – 1.1
Cupcake	1.5
Donut	1.6
Eclair	2.0 – 2.1
Froyo	2.2.x
Gingerbread	2.3 – 2.3.7
Honeycomb	3.0 – 3.2.x
Ice Cream Sandwich	4.0.0 – 4.0.4
Jelly Bean	4.1.x – 4.3

(*AND-SRC-BUILD*, 2013)

Juntamente com cada versão, são adicionadas novas funcionalidades e corrigidos os erros das versões, em (BARROS, 2013) são descritas algumas informações sobre as versões. Desde a versão 1.5, os nomes das versões são codificados com nomes de doces, além disso, é possível notar que todos os *codenomes* aparecem em ordem alfabética. De acordo com (KELION, 2013), a próxima versão (4.4) será chamado de KitKat, uma possível estratégia de *marketing* da Nestle.

Capítulo 4

Replicação de dados

A replicação de dados é alvo de muito estudo no âmbito de banco de dados distribuídos. Segundo (ÖZSU; VALDURIEZ, 1999) a replicação de dados consiste em manter múltiplas cópias de dados em um ou mais nós da rede, também chamados *sites*. A duplicação dos dados fornece disponibilidade mesmo que algumas máquinas não estejam disponíveis.

De acordo com o autor mencionado anteriormente, a replicação pode ser feita de forma parcial ou total. Em base de dados totalmente replicada, todos os nós possuem todos os dados. Enquanto, que na base de dados parcialmente replicada, os nós apresentam fragmentos de dados.

Em (BURETTA, 1997), apresenta outra classificação para a replicação de dados, e refere-se quanto ao tempo de atualização e entrega dos dados às réplicas, podendo ser realizada de maneira síncrona ou assíncrona.

Este capítulo está dividido em 5 seções. Nas seções 4.1 e 4.2 apresentaremos os modelos síncrono e assíncrono, respectivamente. Na seção 4.3, abordaremos o modelo mestre/escravo e atualizável em qualquer lugar. Na seção 4.4, apresentaremos o tratamento da consistência dos dados. Por fim, na seção 4.5, apresentaremos outras considerações sobre a replicação de dados.

4.1 Modelo síncrono

No modelo síncrono, a replicação ocorre de forma simultânea à atualização dos dados, assim, as réplicas devem possuir os mesmos valores ao mesmo tempo. O controle de concorrência nesse modelo é feito bloqueando os sites durante a atualização das réplicas. Segundo (SAITO; SHAPIRO, 2005), esse modelo é recomendado em redes locais (LAN) onde as latências são baixas e ocorrem poucas falhas. Porém, em redes de longa distância (WAN), este tipo de replicação costuma ter baixo desempenho visto que a latência é maior e a confiabilidade menor nessas redes.

4.2 Modelo assíncrono

No modelo assíncrono, as réplicas são atualizadas após um intervalo de tempo após a atualização dos dados, por exemplo, a atualização das réplicas pode ocorrer ao fim do expediente em uma empresa. Esse modelo é recomendado em sistemas nas quais os nós podem ficar sem conexão em algum momento, mas durante a atualização das réplicas, a conexão deve existir. O principal desafio desse modelo é tratar a consistência dos dados, já que em algum momento, as réplicas podem possuir dados inconsistentes.

Outra classificação adotada por (BURETTA, 1997) se diz respeito quanto a quem pode atualizar a base de dados, sendo mestre/escravo ou atualizável em qualquer lugar.

4.3 Modelo mestre/escravo e atualizável em qualquer lugar

No modelo mestre/escravo, existe uma única base de dados, denominado mestre, capaz de sofrer atualizações. Então, as demais réplicas, chamadas escravos, recebem atualizações apenas da réplica mestre. No modelo atualizável em qualquer lugar, não existe o conceito da réplica mestre, todas as réplicas podem sofrer atualizações e atualizá-las para as demais (BURETTA, 1997).

4.4 Tratamento da consistência dos dados.

O tratamento da consistência de dados no modelo atualizável em qualquer lugar em combinação com a replicação síncrona é realizada através do protocolo *two-phase commit* e com a replicação assíncrona é necessário um mecanismo que detecte e resolva os conflitos encontrados (ÖZSU; VALDURIEZ, 1999).

O protocolo *two-phase commit* garante a consistência dos dados entre banco de dados distribuídos, onde um coordenador realiza a sincronização dos dados. O protocolo é composto por duas fases. Na primeira, o coordenador pergunta para todos os processos participantes se estão prontos para realizar a transação, se algum nó participante não responder ou responder que não esta pronto, então é feito um rollback. A segunda fase começa se todos os participantes responderem sim ao coordenador para realizar um commit nos dados, caso algum nó encontre

uma falha, é enviado um sinal para todos os participantes que a transação seja desfeita (BERNSTEIN *et all*, 1987).

4.5 Outras considerações

Em banco de dados distribuídos, a replicação pode ser empregada combinando as diferentes técnicas de replicação segundo as classificações de (ÖZSU; VALDURIEZ, 1999) e (BURETTA, 1997). Entretanto, a escolha da técnica irá depender do cenário da aplicação.

Em se tratando de aplicações móveis, a replicação síncrona não é bem recomendada, visto que os dispositivos podem ficar sem conexão em algum momento, e isso atrapalharia o sincronismo das replicações. A replicação assíncrona é a mais aceita nesse caso, pelo fato da mobilidade que os dispositivos proporcionam, e não é em todo local que existe uma rede com conexão à Internet, então a replicação pode ser executada posteriormente, quando a conexão existir.

Capítulo 5

Estudo de caso

Este capitulo está divido em 4 seções. A seção 5.1 aborda o funcionamento e a estrutura dos aplicativos cliente e servidor. Na seção 5.2, abordaremos as principais classes da aplicação cliente e servidor. Na seção 5.3, será descrito a preparação para o estudo de caso. Na seção 5.4, os testes realizados.

5.1 Funcionamento das aplicações

O aplicativo cliente desenvolvido para o sistema operacional *Android* neste projeto é destinado a estudantes que desejam ter acesso a suas notas e faltas em qualquer lugar através de seu dispositivo móvel. A aplicação servidor é um intermediador para que a aplicação cliente possa fazer acesso ao banco de dados mestre e realizar a replicação das notas e faltas para o aluno.

Para que a replicação possa ocorrer de forma eficiente em sistemas móveis é necessário que os dados se mantenham constantes ao longo tempo. As atualizações recebidas nesta aplicação representam um sistema acadêmico na qual são gerenciados por meio de tabelas de banco de dados.

O aplicativo servidor e o auxiliar de atualização de notas foi desenvolvido na linguagem de programação JavaSE e o SGBD que gerencia os dados é o MySQL. A interação da linguagem com o banco de dados é realizada pelo Connector/J, o *driver* JDBC oficial para o MySQL.

A forma de comunicação entre servidor e cliente é estabelecida através de *sockets*, a aplicação do servidor possui uma porta em um *loop* infinito que aguarda a conexão dos clientes, quando a conexão é estabelecida, o servidor cria um *thread* para tratar cada conexão do cliente.

5.1.1 Funcionamento e arquitetura da aplicação cliente

A aplicação cliente possui uma tela de *login*, na qual somente é permitida a entrada de alunos devidamente cadastrados no banco de dados. Entretanto, para a primeira utilização do aplicativo é necessário conectar-se com o servidor para que possa ser feita a replicação das informações deste aluno. Caso o aparelho não possua conexão, a aplicação faz acesso aos dados a partir do banco de dados do dispositivo.

Para entender melhor o que o aplicativo cliente faz do ponto de vista do usuário é mostrado no diagrama de caso de uso com as principais funcionalidades do sistema na **Figura 5.1.1**.

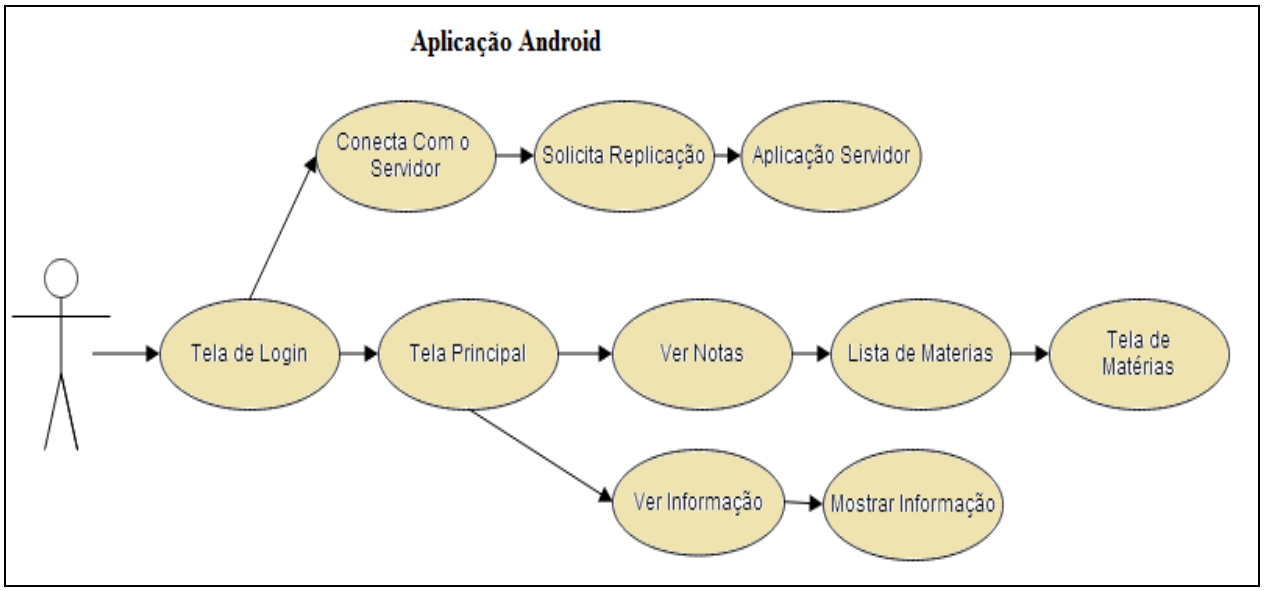


Figura 5.1.1: Diagrama de caso de uso da aplicação cliente.

Ao abrir o aplicativo, o usuário entra na tela de *login*, onde são necessárias as informações de RGM e Senha do mesmo, em seguida é realizada a conexão com a aplicação servidor para verificar se os dados existem no banco de dados mestre, se existirem, a aplicação libera o acesso ao usuário e replica as informações do aluno para a aplicação cliente.

Dentro da tela principal existem três opções. Na primeira opção é possível ver a lista de matérias que o usuário possui notas e faltas. Na segunda opção, são exibidas as informações de cadastro do usuário ativo na aplicação. Por fim, a última opção fecha o aplicativo. Na **Figura 5.1.2**, demonstra o diagrama de atividades da aplicação cliente.

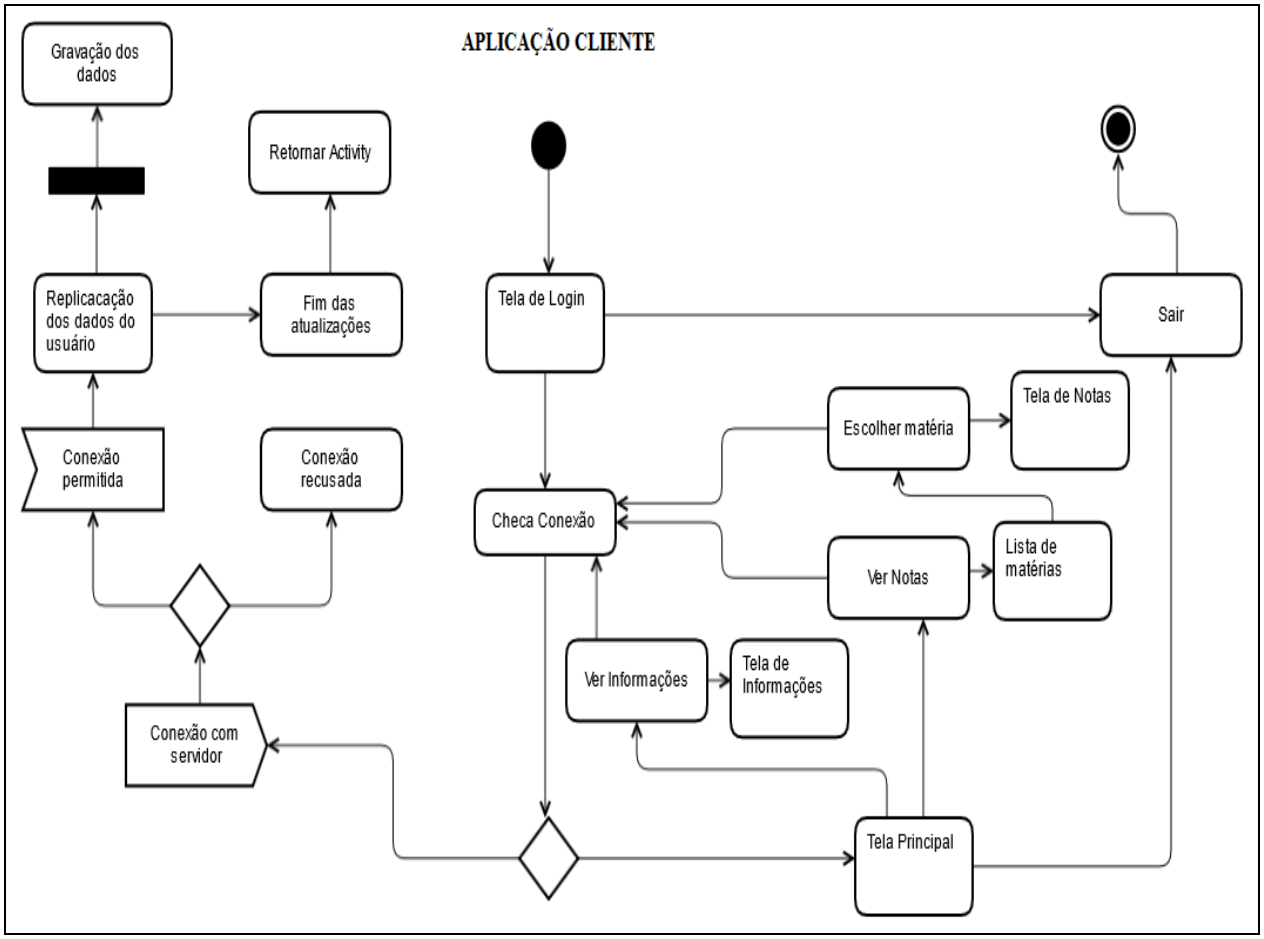


Figura 5.1.2: Diagrama de atividades da aplicação cliente.

5.1.2 Funcionamento e arquitetura da aplicação servidor

O servidor possui uma interface gráfica desenhada a partir da classe JFrame do pacote Swing do Java, nessa interface contém uma área de texto onde é possível visualizar todas as mensagens enviadas pelos clientes para o servidor, um botão para abrir e outro para fechar o socket do servidor.

O cliente se comunica com o servidor através de envios de mensagens, a **Figura 5.1.3** mostra o diagrama de sequência dos passos de comunicação, processamento e envio de dados entre os dois aplicativos. O servidor espera pela conexão de algum cliente, quando for estabelecida, inicia-se um novo thread da classe ManagerClient que é responsável pelo recebimento de mensagens do cliente. As mensagens que o cliente envia são de quatro tipos. O primeiro trata de verificar se um usuário existe no banco de dados do servidor. A segunda é exibir

os dados de cadastro do usuário. O terceiro tipo mostra para a aplicação cliente a relação de notas que o aluno possui. O ultimo tipo de mensagem refere-se aos dados do aluno em uma matéria específica. Por fim, o servidor envia uma mensagem com a seguinte tag "@finish" sinalizando ao cliente que as mensagens terminaram e que a conexão pode ser encerrada.

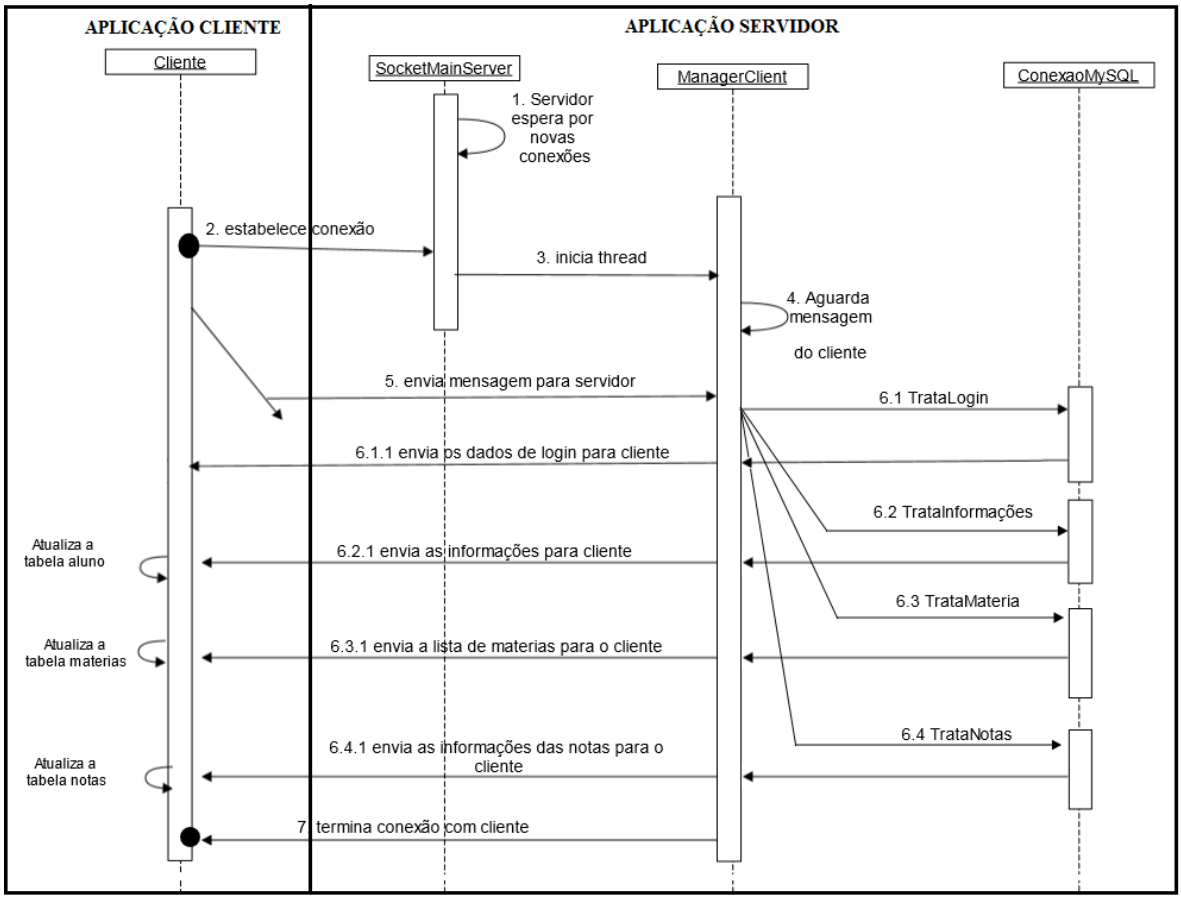


Figura 5.1.3: Diagrama de sequência da comunicação entre aplicação cliente e servidor.

5.2 Classes da implementação dos softwares

O *software* cliente se conectará com o *software* servidor por meio do uso de sockets. A implementação das três aplicações apresentadas até o momento foram realizadas sob a plataforma Windows e utilizando a linguagem Java. Para efetuar a conexão com o banco de dados MySQL, é necessário a adição do connector/J[1] ao *library patch* do *software* servidor.

[1] http://dev.mysql.com/downloads/connector/j/

Esta seção está dividida em 3 seções. Na seção 5.2.1 será apresentada as classes e métodos que constituem a implementação do *software* servidor. Na seção 5.2.2 serão abordados as classes e métodos da implementação do *software* cliente e na seção 5.2.3, as classes e métodos da aplicação auxiliar de atualização das notas.

5.2.1 Módulo Servidor

A **tabela 5.1** mostra as classes que foram implementadas no módulo servidor.

Tabela 5.1: Classes do módulo servidor

Classes	Objetivos
ConexaoMySQL	Classe responsável pelo tratamento das mensagens recebidas pela classe ManagerClient e gerenciamento das mensagens de replicação.
ConnectionFactory	Classe *factory* responsável por realizar a conexão com o SGBD MySQL.
OnMessageReceived	Interface para o recebimento das mensagens.
ServerInterface	Classe responsável pela interface gráfica do servidor.
SocketMainServer	Classe principal responsável por instanciar a aplicação, abrir o socket servidor e aguardar a conexão dos clientes.
ManagerClient	Classe interna à classe SocketMainServer, sua função é tratar as conexões dos clientes e replicar as mensagens para os clientes.

A classe ConexaoMySQL, apresentada no **apêndice D.1**, é responsável por tratar das consultas que os clientes enviam ao servidor. As funções dessa classe são descritas abaixo:

- O método *public void trataMateria(String rgm_cliente, ArrayList<String> cArrayMatID, ArrayList<String> cArrayMatData)* (linha 142) é responsável por encontrar todas as matérias, e seus respectivos professores, que o aluno identificado por *rgm_cliente* possui notas.
- O método *public void trataNotas(String rgm_cliente, String idMat, ArrayList<String> arrayBim, ArrayList<String> arrayData)* (linha 242) é responsável por buscar as notas e faltas da matéria especificado pelo parâmetro *idMat*.

- O método *public String trataLogin(String msg_cliente, ArrayList<String> array)* (linha 29) é responsável por informar se um aluno existe no banco de dados, se for encontrado, suas informações são retornadas para a classe que a chamou.
- O método *public String trataInformacoes(String rgm, String senha, ArrayList<String> array)* (linha 83) é responsável por retornar as possíveis modificações encontradas pelo usuário identificado pelo rgm e senha passados por parâmetro.

A classe *ConnectionFactory* é uma classe *factory* (fábrica, em português) com apenas o método *public Connection getConnection()* (linha 7) que cria a conexão com o banco de dados MySQL.

A classe *ServerInterface* estende a classe *JFrame* do javax.swing, esta classe só possui o construtor da classe (linha 21) que inicia os componentes do frame e inicia o *thread* da classe *SocketMainServer*.

A classe *SocketMainServer* é a classe principal do servidor. Quando o *socket* receber uma nova conexão, a classe *ManagerClient* é instanciada, esta é responsável por receber as mensagens dos clientes e chamar os métodos da classe *ConexaoMySQL* para enviar as informações. Após terminar a execução desses métodos, o thread envia uma mensagem sinalizando o fim das informações.

5.2.1 Módulo Cliente

As classes que compõem a aplicação cliente estão disponíveis na Tabela 5.2.

Tabela 5.2: Classes que compõem a aplicação cliente.

Classes	Objetivo
clientSocket	Classe responsável por realizar a conexão com o servidor e enviar mensagens de solicitação replicação recebidas no construtor da classe.
DatabaseConnection	Classe auxiliar que abre e fecha a conexão com o banco de dados SQLite, além de executar e realizar as consultas ao banco de dados quando a aplicação não tiver conexão com a internet.
DatabaseCreateUpdate	Classe responsável por criar as tabelas do banco de dados.
DatabaseManager	Classe onde é de fato criada a conexão com banco de dados.
InfoActivity	Classe que informa os dados pessoais do cliente.
LoginActivity	Classe para a tela de inicial, o usuário deve informar seus dados para entrar na tela principal do aplicativo.
MainActivity	Classe com a tela principal, o usuário possui as opções de ver suas notas e informações.
MateriaActivity	Classe que contém a lista de matérias que o usuário possui notas.
NotasActivity	Classe responsável por apresentar as notas e faltas a partir de uma matéria escolhida na sua lista de matérias.

As classes *InfoActivity*, *LoginActivity*, *MainActivity*, *MateriaActivity* e *NotasActivity* estendem a classe *Activity*, que é responsável pela interação do usuário com o aplicativo. Existem dois casos nos quais a aplicação faz o acesso a partir de seu banco de dados local, quando dispositivo não possui conexão e quando o servidor não está disponível, essas verificações são realizadas nas classes *LoginActivity*, *MainActivity* e *MateriaActivity.*

A classe *clientSocket* estende a classe *Thread* e é responsável por realizar a conexão com o servidor, enviando inicialmente uma mensagem de consulta e recebendo os resultados das operações até receber a mensagem "*@finish*", onde significa que a conexão pode ser desligada. As mensagens recebidas para atualizar o banco de dados são enviadas ao método *executaSQL()* da classe *DatabaseConnection.*

As classes *DatabaseConnection*, *DatabaseCreateUpdate* e *DatabaseManager* são classes que realizam as operações de criar ou abrir o banco de dados e criar ou atualizar suas tabelas.

A classe *LoginActivity* é responsável pela tela inicial do aplicativo, esperando que o usuário preencha os campos RGM e Senha para que possa entrar na tela principal do aplicativo. Essa classe verifica se o dispositivo possui conexão com a internet, se possuir conexão, a

verificação de login é feita pelo banco de dados do servidor, senão, a aplicação faz a verificação a partir do banco de dados local.

A classe *MainActivity* é responsável pela tela principal do aplicativo, o usuário tem três opções nessa tela, a primeira opção é ver suas notas, ao selecionar essa opção, a classe MainActivity verifica se existe conexão com o servidor para listar todas as matérias que o usuário possuir notas. Se não existir conexão, a lista de matéria será carregada a partir do banco de dados do dispositivo através da função *public List<String> buscaMaterias(SQLiteDatabase connection)* (linha 101), a partir dos itens exibidos na lista, o usuário pode clicar nos itens exibidos e escolher a matéria que deseja ver suas notas e faltas.

A classe NotasActivity recebe o identificador e o nome da matéria selecionada, e os vetores contendo as informações enviadas pelo servidor, se não possuir conexão, o método *public void buscaNotas(SQLiteDatabase connection, String materiaPos)* (linha 123) é chamado para escrever as notas, faltas e os bimestres, encontrados no banco de dados do dispositivo, em três *ArrayList*. Em seguida, o método *public void mostraResultados()* (linha 58) exibe os dados no *TextView* das informações.

A segunda opção é visualizar suas informações pessoais, ao selecionar essa opção, a aplicação vai tentar se conectar ao servidor para receber as informações do aluno ativo, se não conseguir, os dados serão buscados pela função *void buscaDados(SQLiteDatabase connection)* (linha 58) da classe *InfoActivity*, neste método serão exibidos os dados como: o RGM, nome, endereço e e-mail. A terceira opção corresponde à saída do programa, quando clicado no botão de *Sair*, o aplicativo finaliza a *Activity*, o processo do aplicativo e chama o método OnDestroy() que faz uma limpeza final antes de destruir a *activity* (linhas 124 à 131).

5.3 Preparação para o estudo de caso.

5.3.1 Componentes necessários para a aplicação do servidor

- Baixar e instalar a IDE Eclipse;
- Baixar e instalar o SGBD MySQL Server 5.6;
- Baixar e adicionar o connector/J do MySQL à *library patch* do projeto do servidor;

5.3.2 Componentes necessários para a aplicação cliente

- Baixar e instalar a IDE Eclipse;
- Baixar e instalar o plugin ADT para o *Android*;
- Baixar e instalar o *Android* SDK;

5.4 Resultados e Discussões

Os testes de software determinam se o produto atingiu suas especificações e funcionou da maneira correta para o ambiente da qual ele foi projetado. Os testes realizados nesta seção referem-se à execução do programa propriamente dito, através de entradas manuais de dados.

5.4.1 Detalhes dos testes

Os testes concebidos foram realizados em duas máquinas com as seguintes especificações:

- Processador Intel Core i5 580M 2.66GHz, 4GB DDR3 1066MHz, HD Sata 640 GB.
- Processador Intel Core 2 Duo P8600 2.40GHz, 4GB DDR2 800MHz, HD Sata 320 GB.

O banco de dados *universidade* foi criado com as seguintes tabelas: *aluno*, *professor*, *matéria* e *notas*. Na tabela *aluno,* foi povoado com 15 alunos fictícios. Na tabela professor, com 6 professores fictícios, na tabela *matéria* com 7 matérias fictícias e na tabela *notas* com 210 dados fictícios. No apêndice G mostra os códigos utilizados para a criação do banco de dados *universidade*.

Os testes realizados com a aplicação cliente utilizaram-se do emulador QVGA com sistema operacional *Android* 4.3 – API 18. A primeira aplicação a ser executada para o início dos testes é a aplicação servidor, e deve ser clicado no botão *Iniciar Servidor* para abrir o *socket* servidor, como pode ser observado na **Figura 5.4.1.**

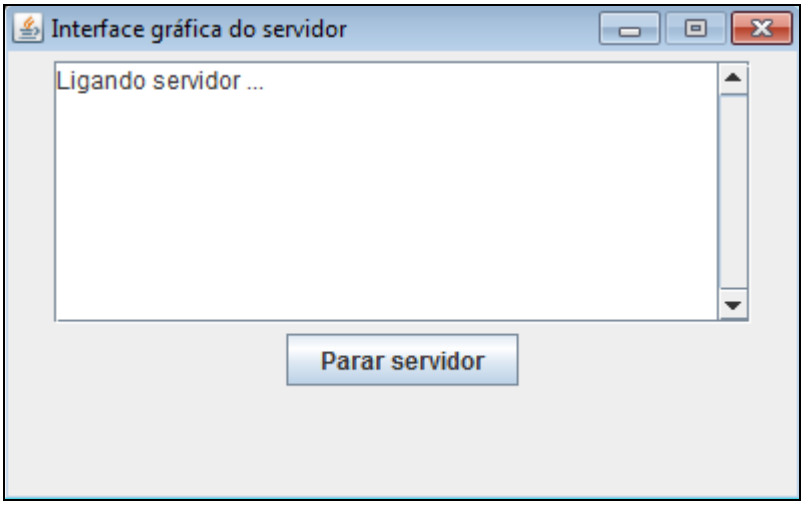


Figura 5.4.1: Interface gráfica do servidor com socket servidor aberto.

Em seguida, iniciamos o emulador com a aplicação cliente, a primeira execução da aplicação irá criar o banco de dados universidade e as tabelas *aluno*, *professor*, *materia* e *notas*, sem conter nenhuma informação do banco de dados do servidor. Na **Figura 5.4.2**, mostra o cliente em execução pela primeira vez, e após preencher os campos com os dados *1* para o RGM e *65859052* para o campo de senha. Em seguida, a aplicação tenta se conectar ao servidor para realizar a verificação de login. Se o servidor estiver indisponível ou o dispositivo não estiver com conexão à Internet, a aplicação fará a verificação de login com o banco de dados local.

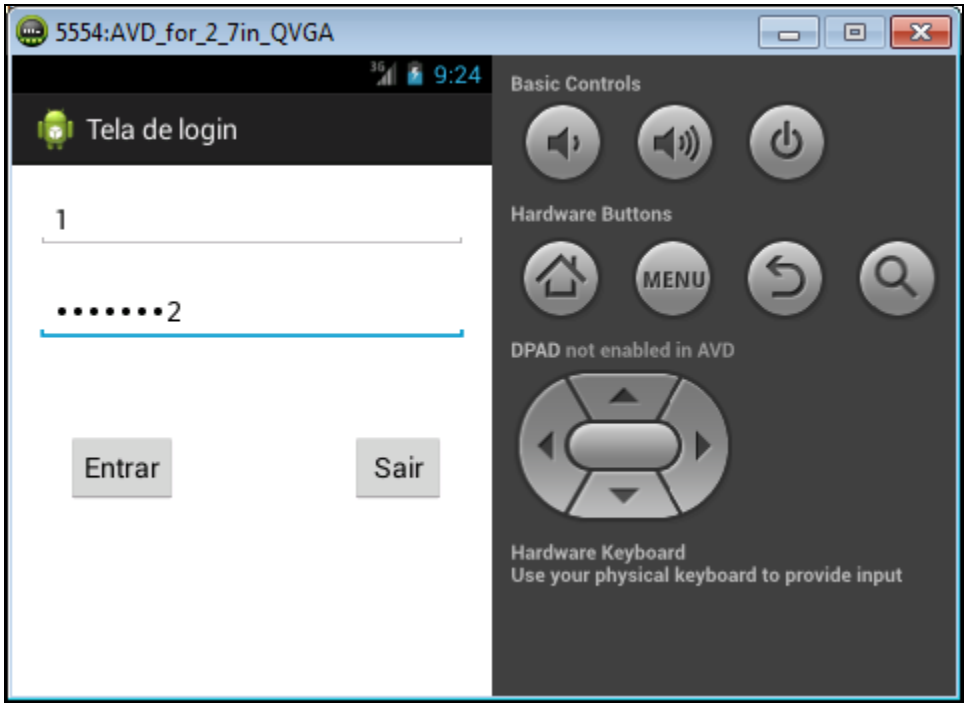


Figura 5.4.2: Tela de entrada da aplicação cliente.

O servidor por sua vez, recebe uma mensagem emitida pelo cliente com a tag *@solicitacao_login*, significando que a comunicação é para realizar a verificação de login, conforme a **Figura 5.4.3**.

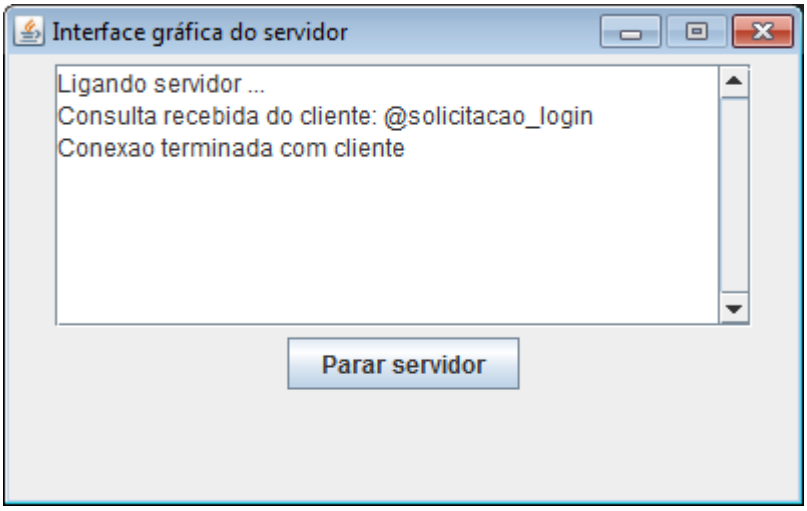
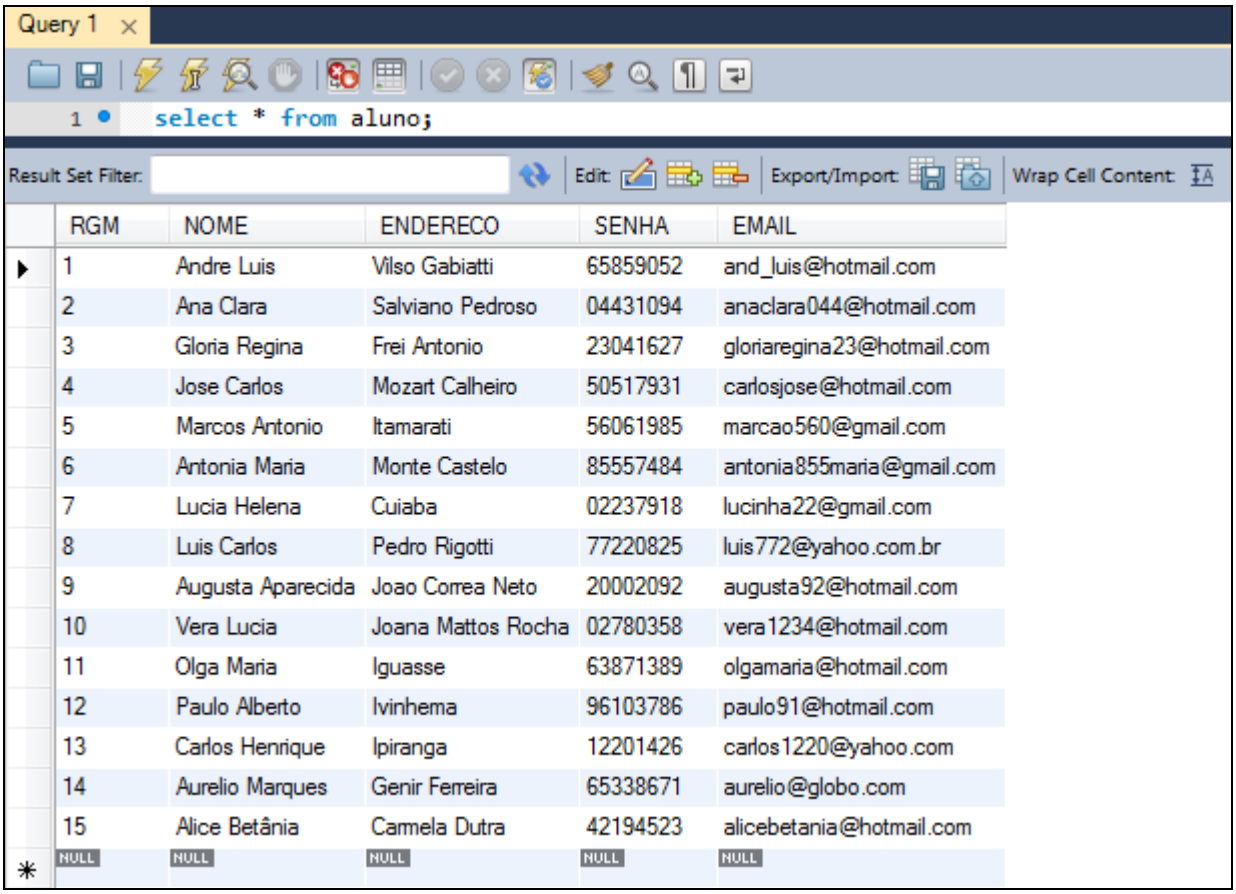


Figura 5.4.3: Servidor recebendo solicitação de login com RGM e senha do usuário.

Ao verificar a mensagem recebida, o servidor chama os método *trataLogin()* da classe *ConexaoMySQL*, faz a verificação do RGM e senha, e envia para a classe ManagerClient uma mensagem avisando que o login foi aceito, senão, o cliente recebe uma mensagem de dados incorretos. A **Figura 5.4.4** corresponde aos dados encontrados na tabela *aluno* do banco de dados do servidor, o aluno em questão tem o nome de "Andre Luis", endereço "Vilso Gabiatti" e *e-mail* "and_luis@hotmail.com".



RGM	NOME	ENDERECO	SENHA	EMAIL
1	Andre Luis	Vilso Gabiatti	65859052	and_luis@hotmail.com
2	Ana Clara	Salviano Pedroso	04431094	anaclara044@hotmail.com
3	Gloria Regina	Frei Antonio	23041627	gloriaregina23@hotmail.com
4	Jose Carlos	Mozart Calheiro	50517931	carlosjose@hotmail.com
5	Marcos Antonio	Itamarati	56061985	marcao560@gmail.com
6	Antonia Maria	Monte Castelo	85557484	antonia855maria@gmail.com
7	Lucia Helena	Cuiaba	02237918	lucinha22@gmail.com
8	Luis Carlos	Pedro Rigotti	77220825	luis772@yahoo.com.br
9	Augusta Aparecida	Joao Correa Neto	20002092	augusta92@hotmail.com
10	Vera Lucia	Joana Mattos Rocha	02780358	vera1234@hotmail.com
11	Olga Maria	Iguasse	63871389	olgamaria@hotmail.com
12	Paulo Alberto	Ivinhema	96103786	paulo91@hotmail.com
13	Carlos Henrique	Ipiranga	12201426	carlos1220@yahoo.com
14	Aurelio Marques	Genir Ferreira	65338671	aurelio@globo.com
15	Alice Betânia	Carmela Dutra	42194523	alicebetania@hotmail.com
NULL	NULL	NULL	NULL	NULL

Figura 5.4.4: Visualização dos dados pelo MySQL Workbench.

Na **Figura 5.4.5** apresenta a tela principal do cliente quando entrar no sistema. Nele, os usuários possuem 3 opções.

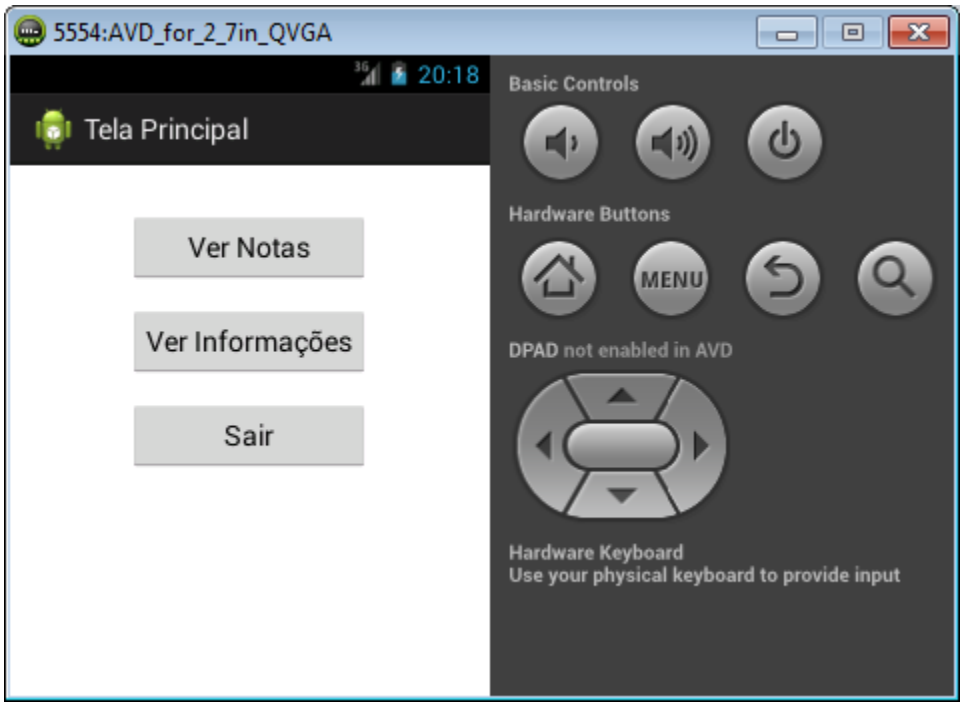


Figura 5.4.5: Tela principal do cliente.

Quando escolhida a opção “Ver Informações”, é enviada uma mensagem ao servidor solicitando as informações do usuário, então o servidor responde enviando os dados de cadastro deste aluno. A classe *InfoActivity* recebe os dados e preenche os campos com as informações do usuário enviados pelo servidor. Na **Figura 5.4.6**, mostra o resultado dessa operação. Todos os valores das colunas, exceto o da senha, da tabela aluno com o RGM utilizado no login são mostrados para o usuário.

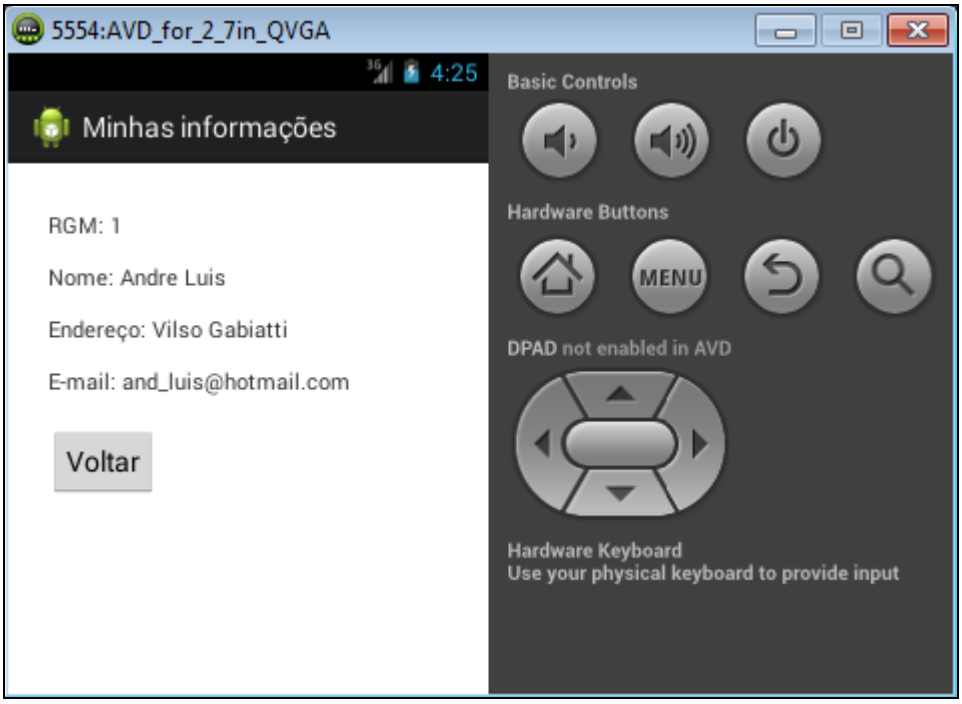


Figura 5.4.6: Informações do usuário ativo no sistema.

Ao retornar à tela anterior e selecionar a opção “Ver Notas”, a classe clientSocket enviará uma mensagem para o servidor solicitando as matérias que o usuário possui notas e replicando as matérias que o banco de dados do dispositivo não possuir em sua tabela. A classe MateriaActivity recebe essa lista e exibe para o usuário, **Figura 5.4.7**.

Figura 5.4.7: Lista de matérias do aluno.

Ao escolher um item da lista, a aplicação cliente envia uma mensagem solicitando as notas referentes à matéria selecionada. O servidor recebe o identificador da matéria e as notas que o usuário já dispõe em seu sistema. O método trataNotas() da classe ConexaoMySQL recebe esses parâmetros, prepara as mensagens de replicação e busca todas notas e faltas para essa matéria. O servidor responde à solicitação do cliente enviando todas as notas, faltas e as mensagens de replicação. As mensagens de replicação recebidas pelo cliente são processadas pelo método executaSQL() da classe *DatabaseConnection*, e as notas e faltas são enviadas para a classe *NotasActivity*.

A **Figura 5.4.8** mostra que o usuário possui duas notas, referentes ao primeiro e segundo bimestre e com 24 faltas nesses dois bimestres.

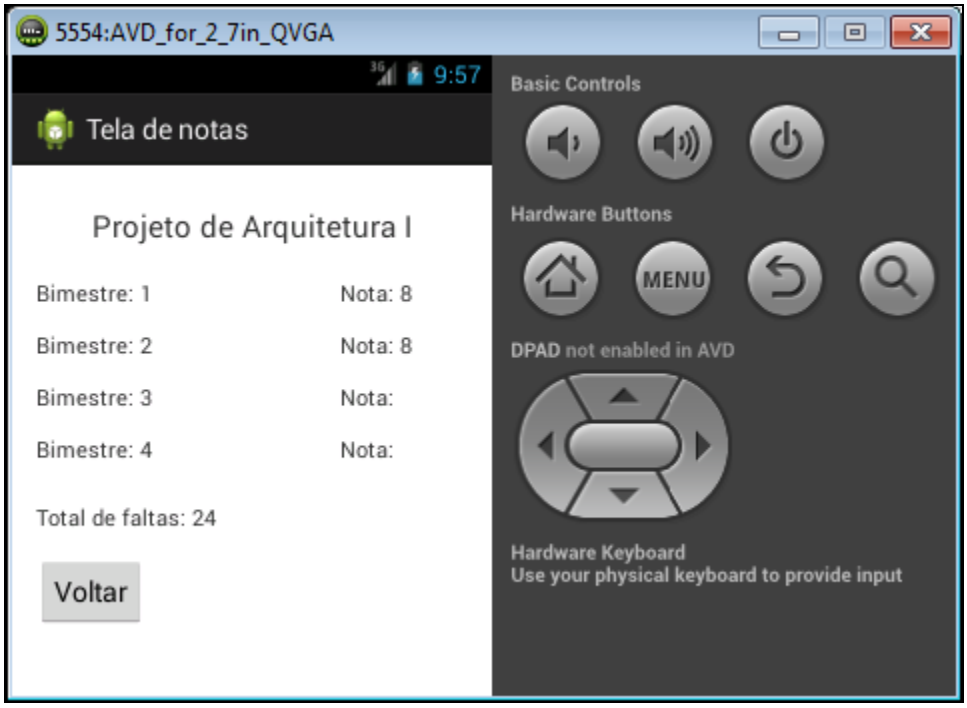


Figura 5.4.8: Notas e faltas do aluno.

Em seguida, inserimos o valor 6 para nota do terceiro bimestre do aluno "Andre Luis" e o valor 8 para a quantidade de faltas neste bimestre. Observando os dados pelo MySQL Workbench, na **Figura 5.4.9**, confirma-se que a nota e as faltas do terceiro bimestre foram inseridas.

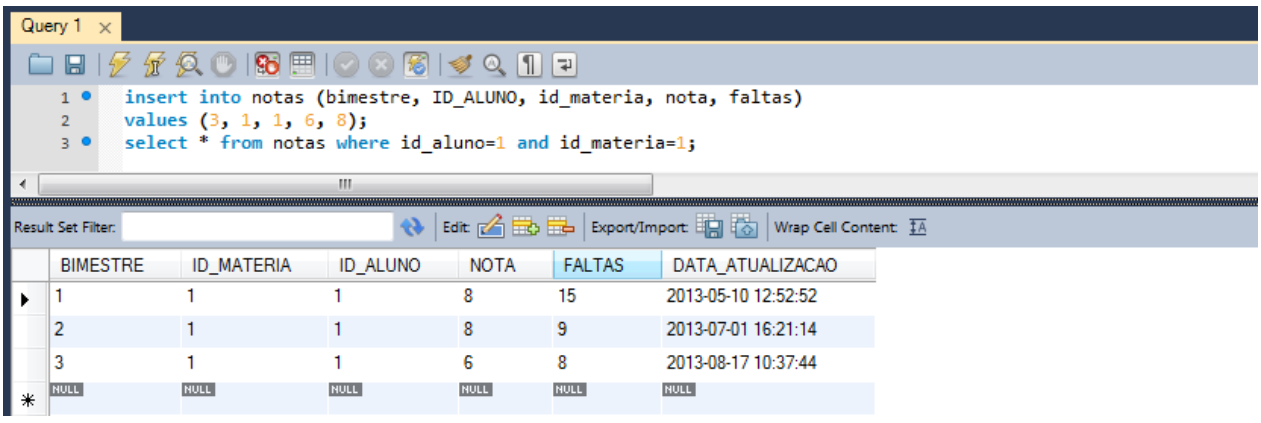


Figura 5.4.9: Nota e faltas do terceiro bimestre inseridas.

Na aplicação cliente, o usuário pode receber a replicação desta nota retornando para a tela com a lista de matérias e selecionando novamente a matéria “Projeto de Arquitetura I”. O servidor vai buscar todas as notas que o usuário possuir na matéria “Projeto de Arquitetura I” e retornar as mensagens de replicação juntamente com as notas e faltas para o cliente. O cliente recebe os dados e executa as mensagens de replicação em seu banco de dados. A **Figura 5.4.10** mostra os dados atualizados na activity das notas.

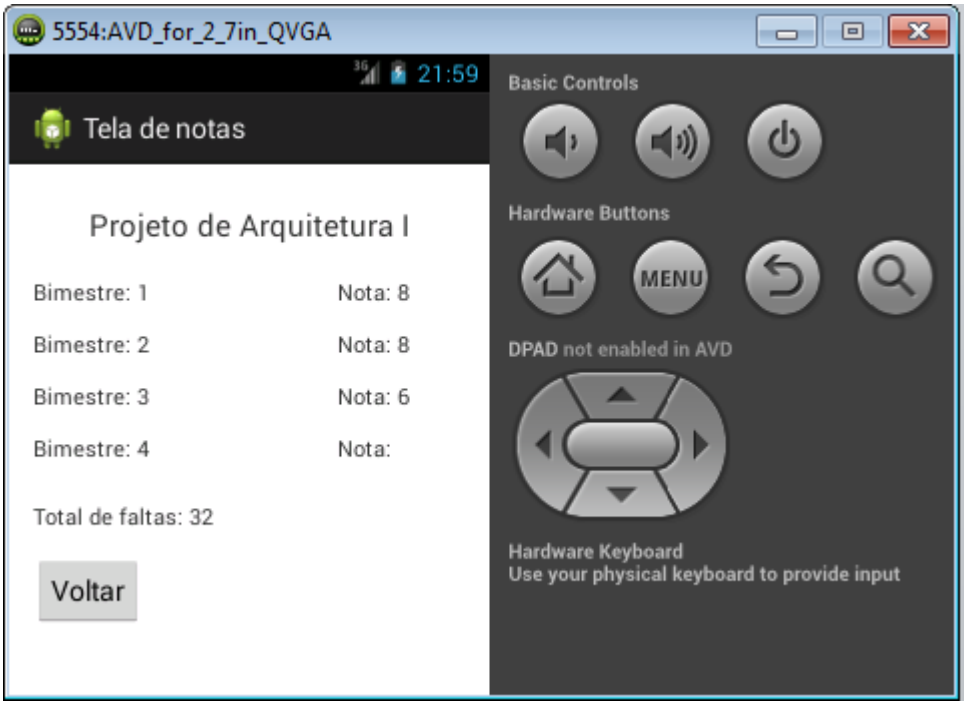


Figura 5.4.10: Visualização dos dados na aplicação cliente.

Entrando no prompt de comando e executando o ADB é possível acessar o banco de dados do dispositivo para verificar que a replicação dos dados foi realizada corretamente, conforme mostra a **Figura 5.4.11.**

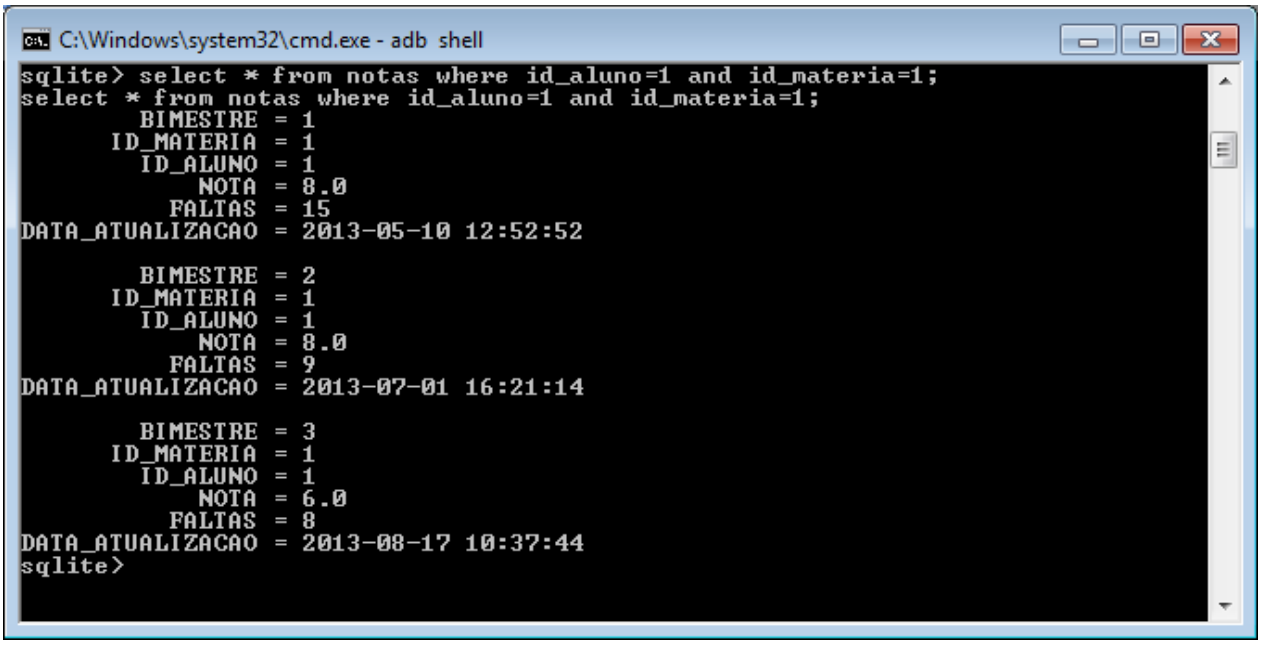


Figura 5.4.11: Notas e faltas replicadas.

Realizando uma comparação com a utilização de uma coluna adicional na tabela notas que corresponde à data de atualização do registro e a não utilização deste campo verificou se que a quantidade de bytes enviados para o cliente é muito menor com a utilização deste campo adicional. Em média para cada instrução de inserção na tabela corresponde a 250 bytes e de atualização é de 135 bytes. Se houver um momento em que for necessário enviar quatro instruções de inserção na tabela notas (correspondentes aos quatros bimestres) serão enviados 1 kilobyte de dados tanto com ou sem o campo adicional.

Quando for necessário enviar mensagens de atualização dos registros, sem a utilização do campo são enviados 540 bytes. Com a utilização do campo adicional este número diminui de acordo com a a data do registro contido no banco de dados do cliente. A mensagem só vai ser enviada para o cliente se a data do registro no banco do servidor for mais recente que a data contida no banco de dados do cliente. Assim, excluímos a replicação dos dados nas quais o cliente já recebeu anteriormente.

Capítulo 6

Considerações Finais

Com os estudos e trabalhos realizados, podemos observar o grande avanço na área da computação móvel, bem como em suas tecnologias empregadas. A popularização das redes sem fio também contribuiu significativamente para a facilidade de acesso às informações, essa facilidade de acesso veio acompanhada dos tamanhos dos dispositivos que se encontram no mercado atualmente, que estão cada vez menores e mais potentes.

Entretanto, mesmo com a popularização dessas redes, encontramos locais e momentos nas quais estamos desconectados do mundo virtual. A solução proposta neste trabalho de final de curso baseia-se na capacidade com que os dispositivos têm em armazenar dados, especificadamente na utilização do banco de dados nativo do sistema operacional móvel *Android*, o SQLite.

Para que o usuário tenha acesso às informações mesmo estando fora da rede, foi utilizado técnicas de replicação de dados estudadas no decorrer do projeto. As técnicas de replicação podem variar de acordo com o ambiente e domínio da aplicação. Contudo, tratando-se de computação móvel, o método mais aceito é a replicação assíncrona, onde a aplicação pode ficar sem conexão durante um tempo e executar a replicação assim que a conexão existir.

Outro fator que pode ser determinante na escolha de realizar ou não as replicações são os tipos de dados a serem replicados. Em uma aplicação onde há alterações constantes nas informações, a replicação pode não ser uma tarefa fácil de ser executada e bem vista pelo usuário, já que as informações podem não estar consistentes na maior parte do tempo em que o aplicativo não estiver sincronizado com banco de dados mestre ou servidor mestre.

Neste estudo foi apresentado um método de replicação onde o cliente, com a aplicação *Android*, e o servidor, aplicação para desktop, se comunicam a partir de *sockets*, permitindo a troca de mensagens entre as aplicações. É importante ressaltar que, durante os testes e execução

da aplicação *Android*, o usuário ou cliente da aplicação deve conhecer o endereço de IP e porta do servidor.

As mensagens enviadas pelo servidor correspondem aos dados de seu banco de dados, sob a forma da linguagem de consulta SQL, quando chegada à aplicação cliente, a mensagem é executada no banco de dados do cliente. Assim, quando a aplicação cliente não conseguir estabelecer conexão com o servidor, o aplicativo acessa o banco de dados do dispositivo e fornece os dados para o usuário.

6.1 Trabalhos futuros

Nesta seção serão apresentados algumas possíveis funcionalidades que podem ser agregadas e estudadas mais adiante.

6.1.1 Aumento na quantidade de notas finais

O sistema poderia possibilitar a escolha da quantidade de notas finais em uma matéria, não ficando restrito a apenas quatro.

6.1.2 Inclusão de notas livres

O sistema poderia adicionar notas extras para as matérias, então, a nota final seria calculada com base nessas notas mais a nota da prova.

6.1.3 Listagem de aluno por matéria

Implementação de uma lista de alunos que fazem uma determinada matéria.

Apêndice A

Instalação do NetBeans IDE no Windows 7

1. Baixar o instalador do site oficial[1].

   Obs: Para concluir a instalação com sucesso é necessário ter instalado o JDK 7. Caso não tenha o JDK, encontra-se em sua página oficial[2].

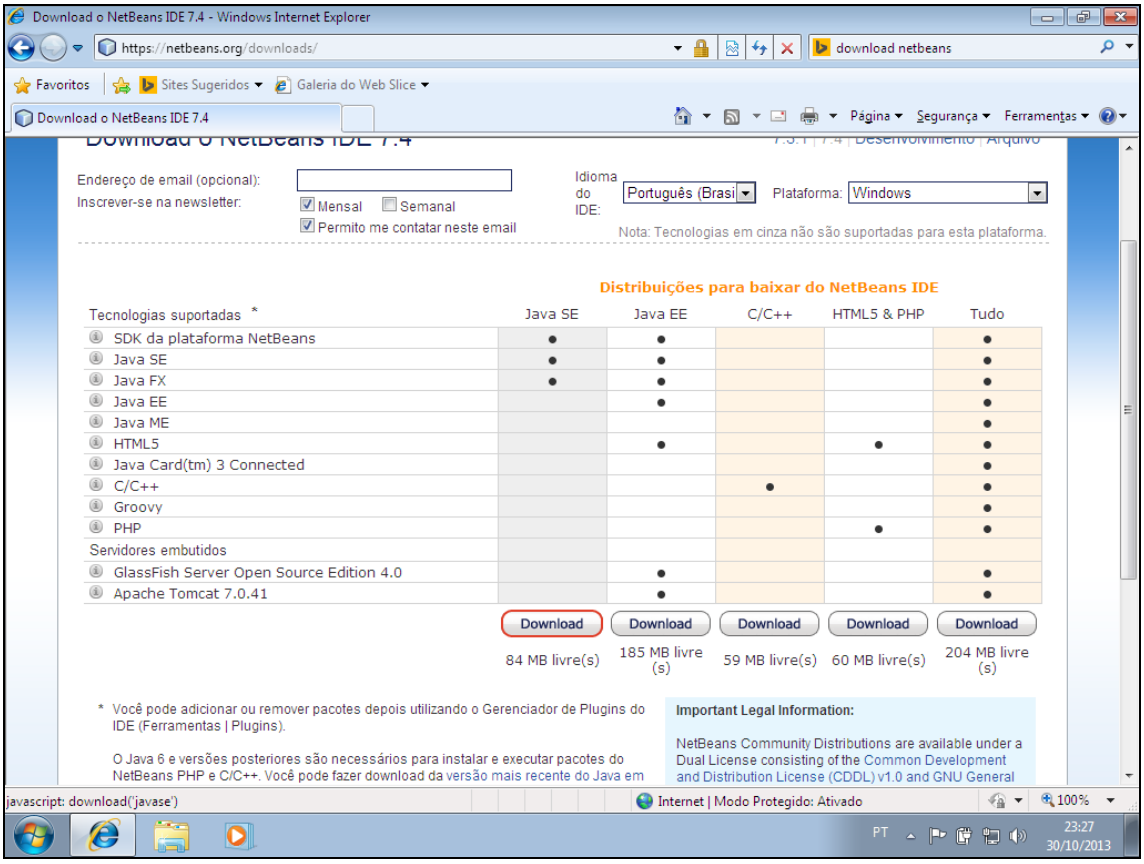


Figura A.1: Página oficial de download do NetBeans.

2. Após realizar o download, abra o instalador e clique em “Próximo”.

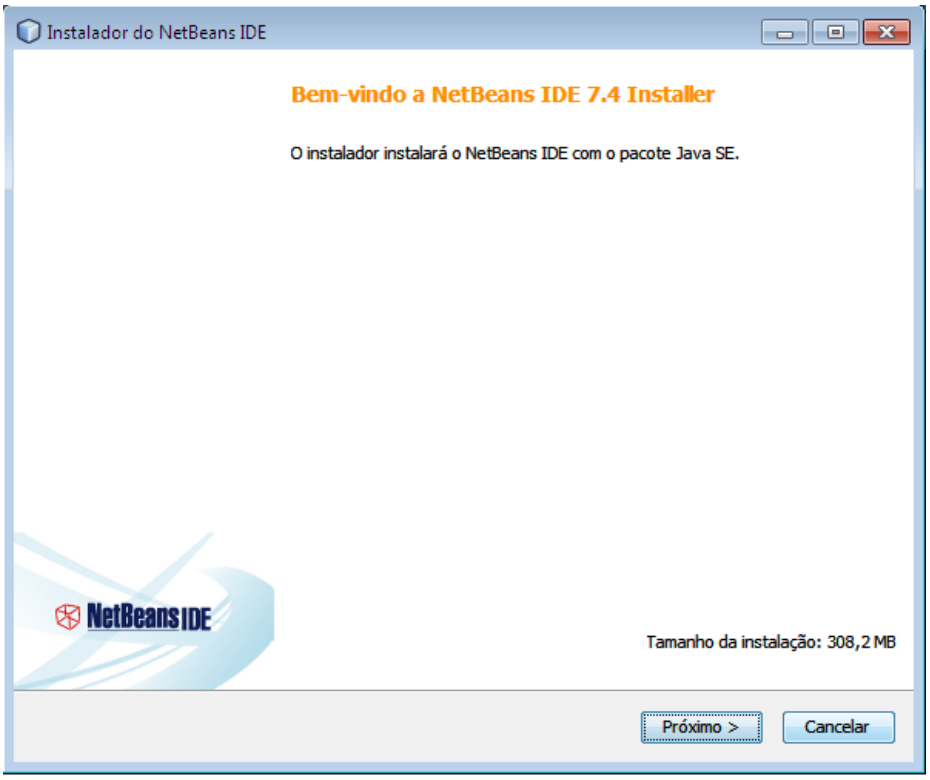


Figura A.2: Instalador do NetBeans IDE.

3. Leia o contrato de licença, se você estiver de acordo, marque a caixa para aceitar os termos do contrato e clique em “Próximo”.

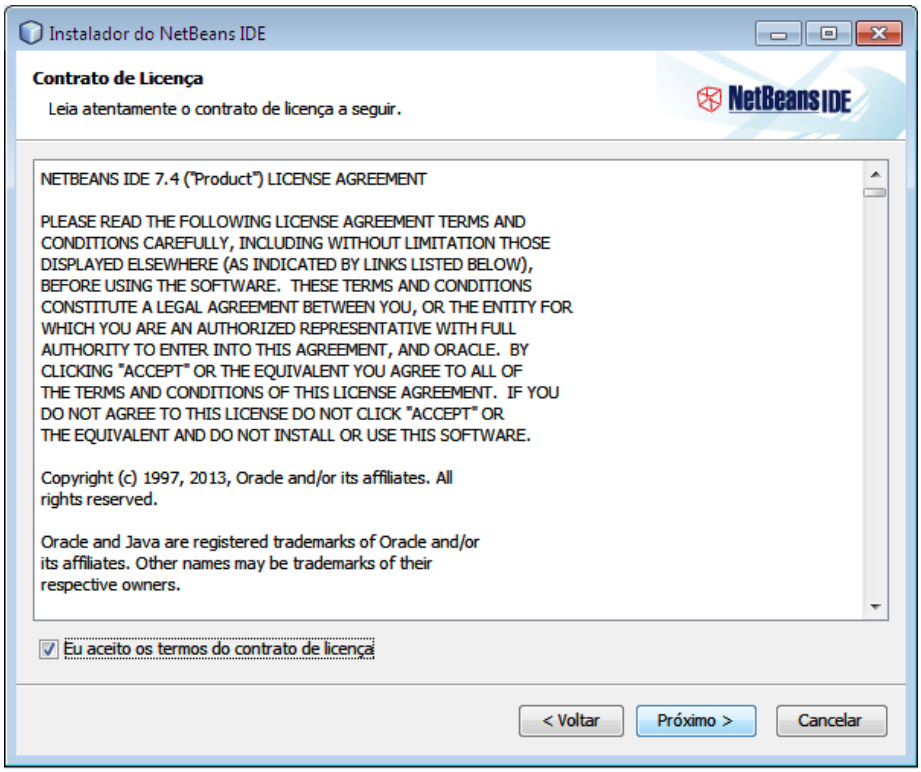


Figura A.3: Termos do contrato de licença do NetBeans IDE.

4. Leia o contrato de licença do JUnit, se estiver de acordo e desejar instalar, marque em aceito os termos e instalar JUnit, caso contrário, marque não instalar e clique em “Próximo”.

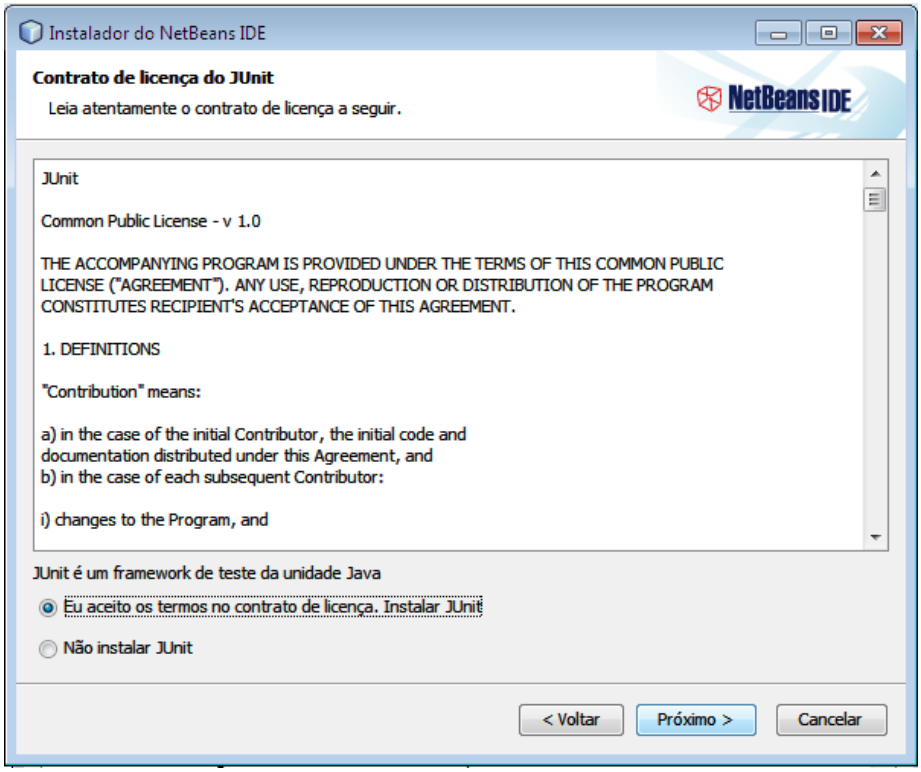


Figura A.4: Contrato de licença do JUnit.

5. Escolha o local onde o NetBeans IDE será instalado e o local onde está instalado o JDK 7.

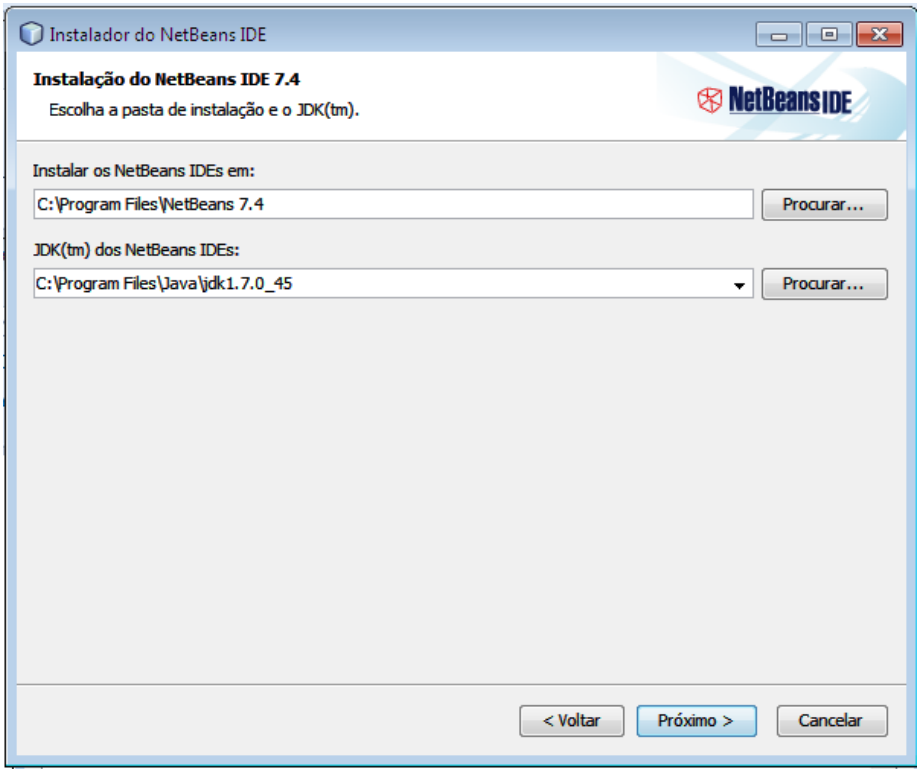


Figura A.5: Diretório de instalação do NetBeans IDE.

6. Se quiser verificar se há uma atualização mais recente em seus *plugins*, marque a opção "Verificar Atualizações". Em seguida, clique em instalar e aguarde até concluir a instalação.

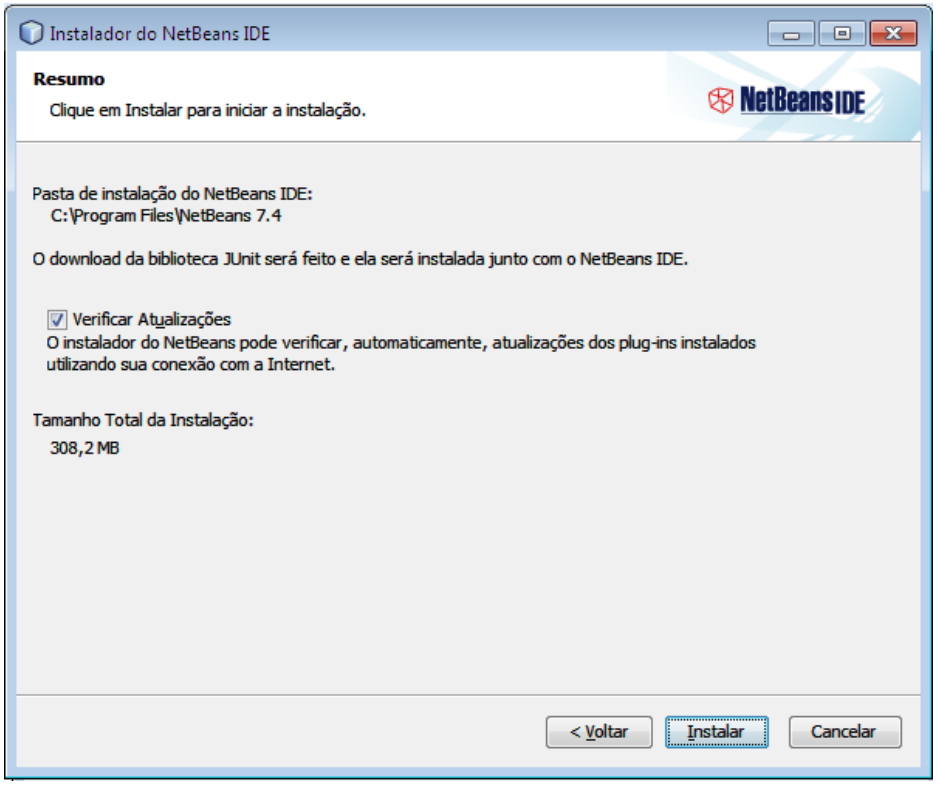


Figura A.6: Clique em instalar para instalar o NetBeans IDE.

7. Por fim, clique em “Finalizar” para concluir a instalação e fechar o instalador.

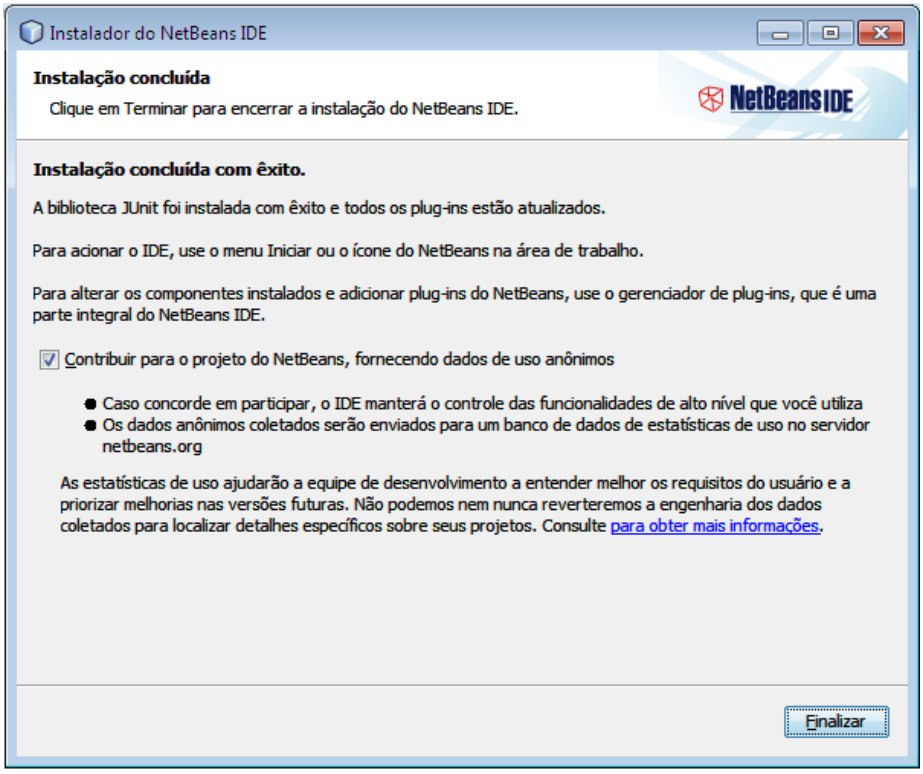


Figura A.7: Clique em “Finalizar” para concluir a instalação.

Apêndice B

Instalação do Eclipse

1. Entre na página oficial de download[1] do Eclipse IDE, escolha o pacote Eclipse Standard 4.3.1 ou Eclipse IDE for Java Developers, em seguida, escolha o pacote ideal para a versão do seu Windows (32 ou 64 bits).

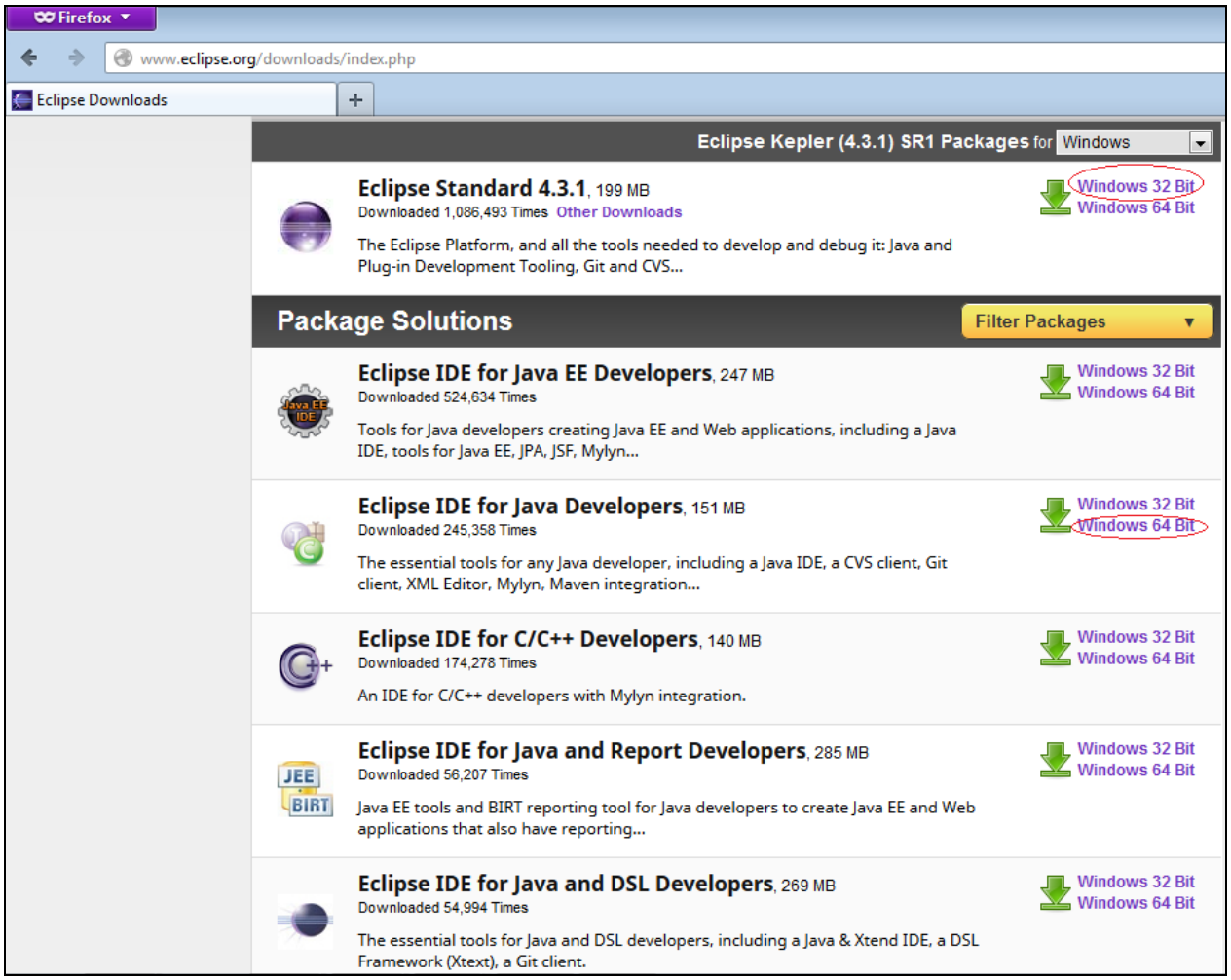


Figura B.1: Escolha a opção 32 ou 64 bits, de acordo com a versão do seu Windows.

[1]http://www.eclipse.org/downloads/

2. Após realizar o download, descompacte o arquivo no diretório à sua escolha, abra a pasta descompactada e execute o arquivo eclipse.exe. O Eclipse IDE não requer instalação, somente a descompactação do arquivo baixado.

Apêndice C

Instalação do Android SDK no Windows 7

1. Faça o download do Android SDK no site oficial[1]. O pacote padrão do Android SDK já contém o Eclipse IDE.
2. Após o download, descompacte o arquivo em um diretório à sua escolha.
3. Vá até o diretório descompactado e execute o arquivo SDK Manager.exe.
4. Selecione Android 4.3 (API 18) e clique em "Install Packages".

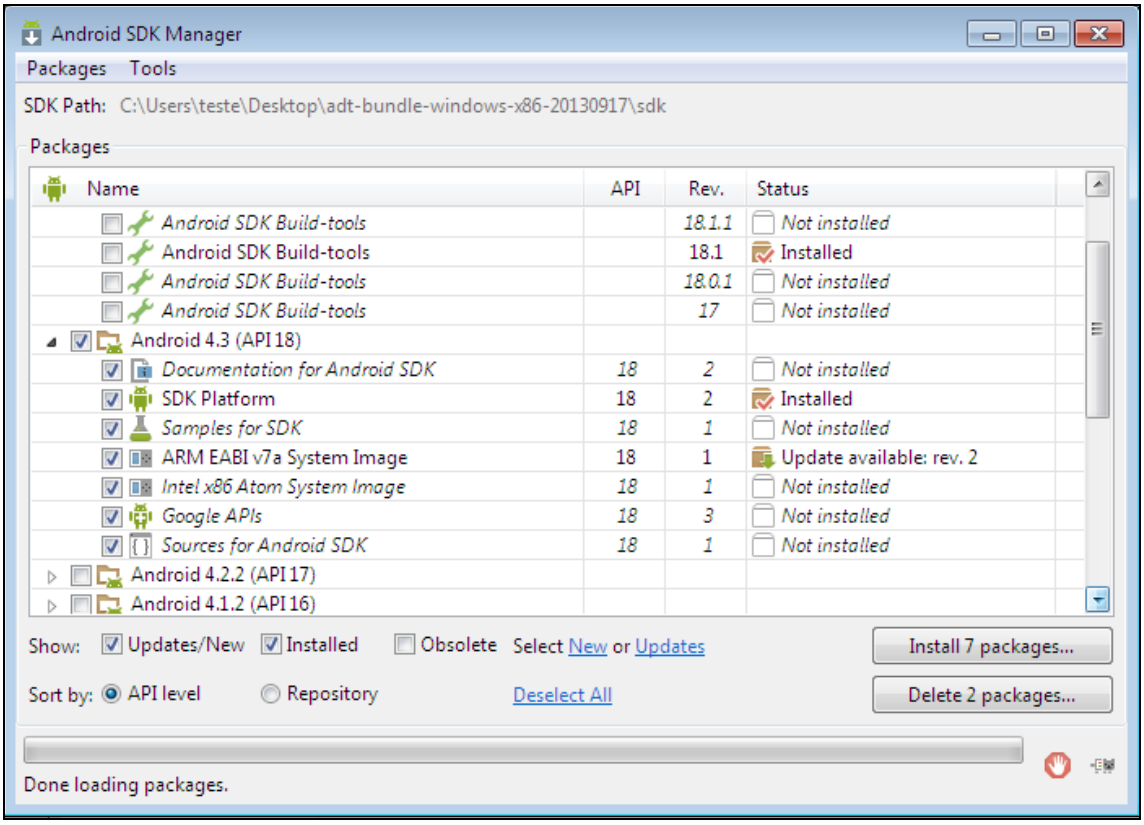


Figura C.1: Escolha a versão 4.3 do Android com API 18.

[1]http://developer.android.com/sdk/index.html

5. Aceite os termos do contrato de licença e espere os pacotes serem baixados. Após a conclusão dos downloads, abra o Eclipse e clique na aba “Help” e selecione a opção “*Install New Software...*”.
6. Na janela que abrir, clique em “Add”. Em *name* digite “*ADT Plugin*” e em *location* preencha com: https://dl-ssl.google.com/android/eclipse/;

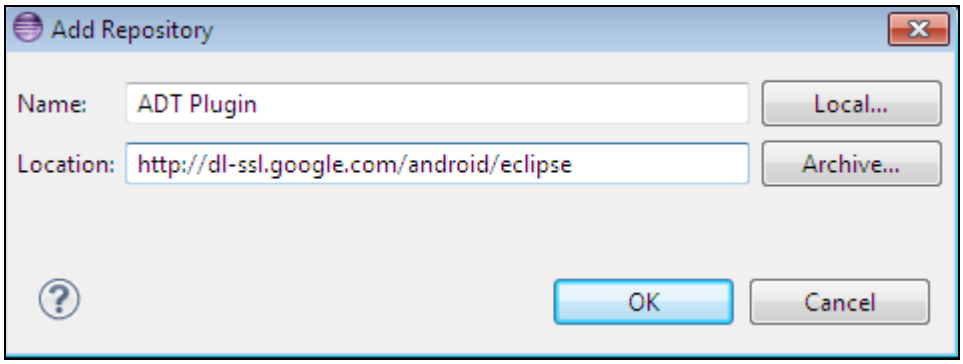


Figura C.2: Janela para adicionar o repositório do plugin ADT.

7. Em seguida, selecione “Developer Tools” ou seleciona apenas as ferramentas que deseja baixar e clique em “Next”, como mostra a **Figura C.3**.
8. Leia e aceite as licenças, e clique em “Finish”. Após completar a instalação, reinicie o Eclipse.

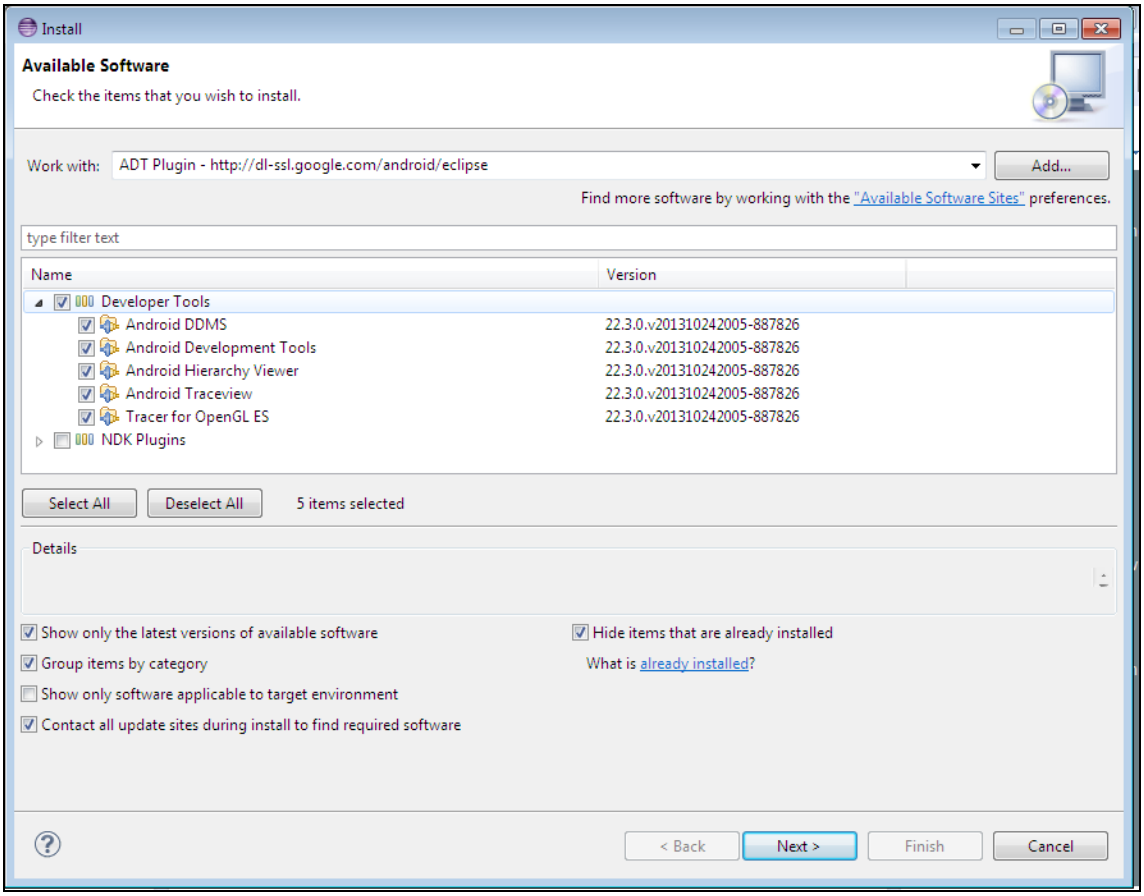


Figura C.3: Selecione as ferramentas que deseja baixar.

Para a criação e configuração do emulador AVD (Android Virtual Device), siga os passos abaixo:

1. Abra o Eclipse, selecione a aba “Window”, clique em Android Virtual Device Manager.
2. Na janela do AVD Manager, selecione a aba “*Device Definitions*”. Escolha um dispositivo de sua preferência e clique em “*Create AVD*”. Como mostra na **Figura C.4**.

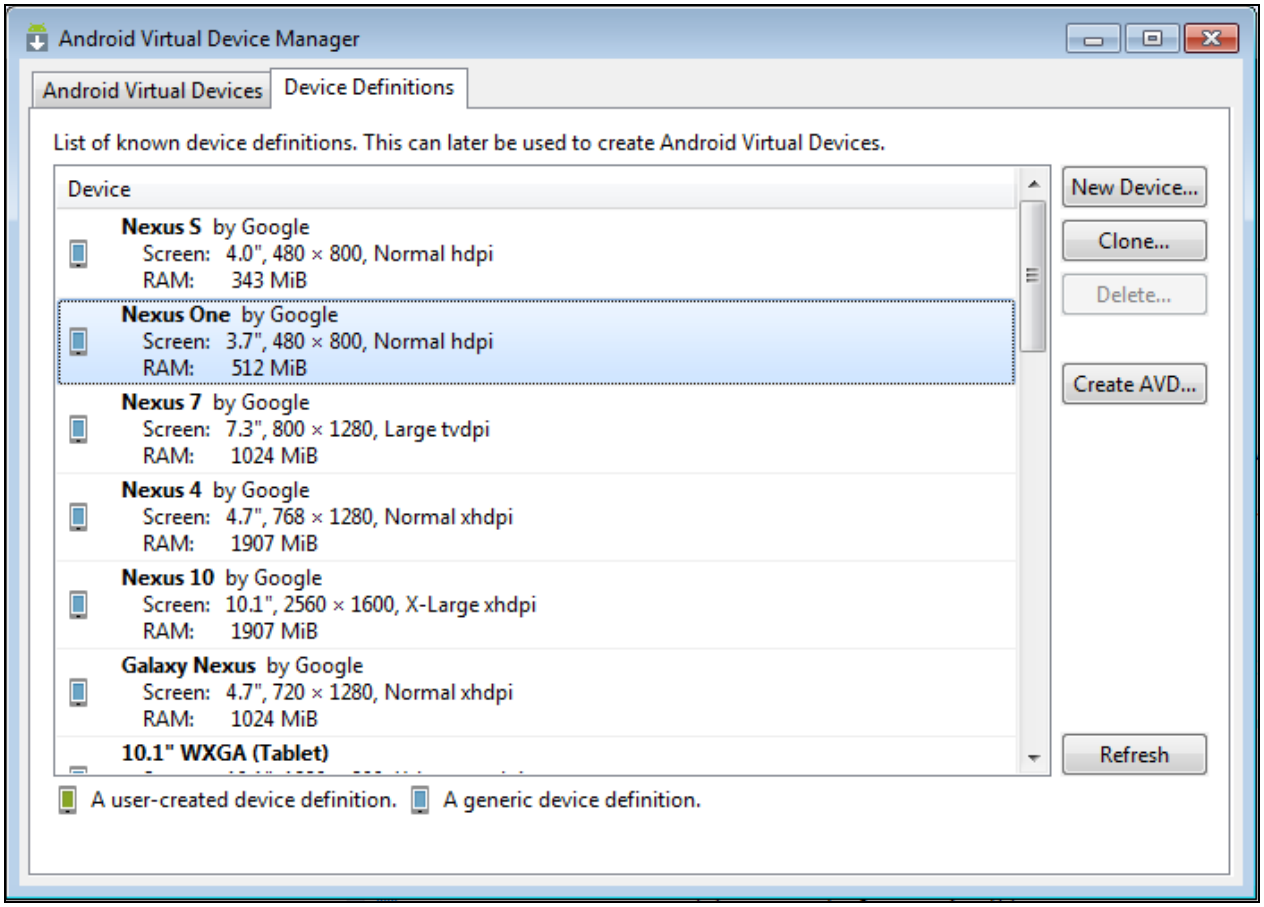


Figura C.4: AVD Manager

3. Em seguida, preencha os campos solicitados e em Target, selecione a opção *Android 4.3 (API 18)*. Em seguida, escolha a quantidade de memória do emulador e clique em "OK" para criar o emulador.

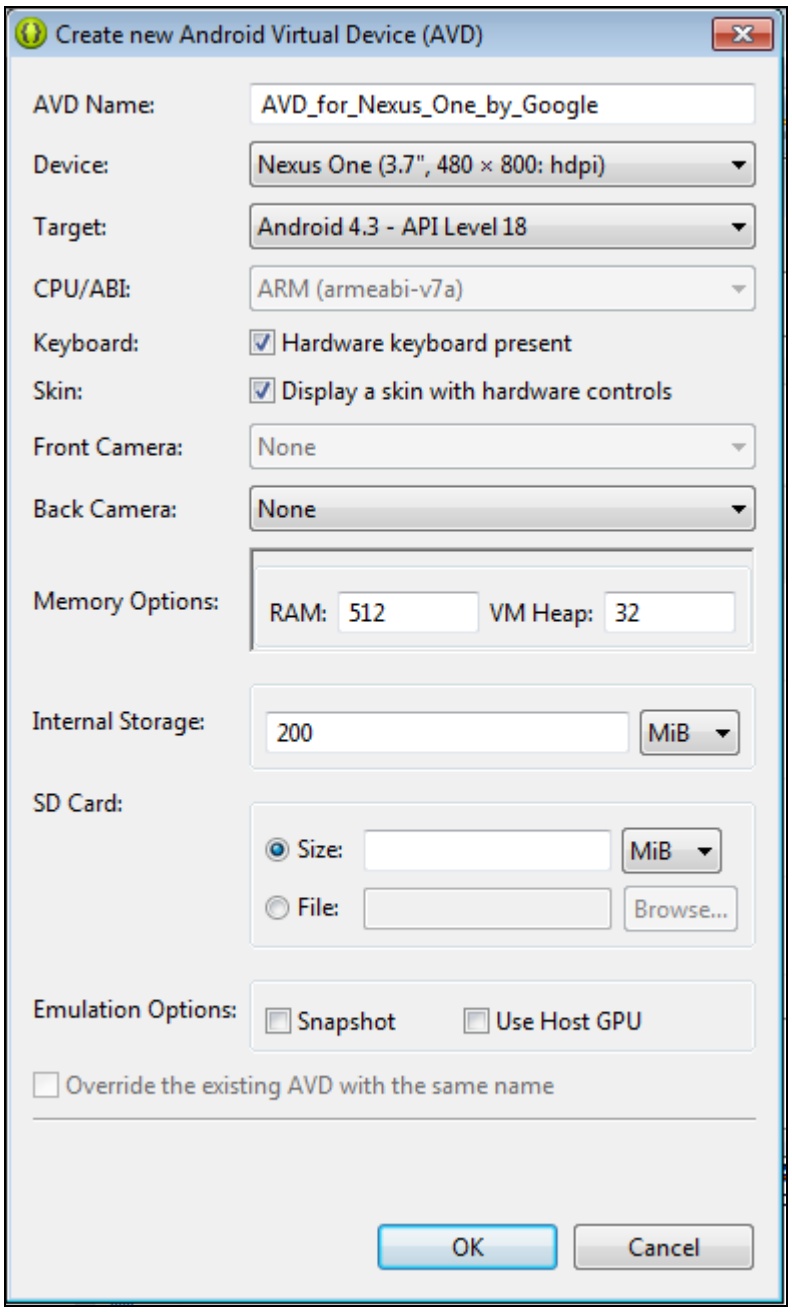


Figura C.5: Janela de criação do novo AVD.

Apêndice D

Classes implementadas no servidor

D.1 Classe ConexaoMySQL

```
1   import java.io.PrintWriter;
2   import java.sql.Connection;
3   import java.sql.PreparedStatement;
4   import java.sql.ResultSet;
5   import java.sql.SQLException;
6   import java.text.ParseException;
7   import java.text.SimpleDateFormat;
8   import java.util.ArrayList;
9   import java.util.Date;
10
11  public class ConexaoMYSQL {
12          static Connection conexao;
13          ArrayList<String> arrayInfo = new ArrayList<String>();
14          ArrayList<String> arrayProf = new ArrayList<String>();
15          ArrayList<String> arrayAtMateria = new ArrayList<String>();
16          ArrayList<String> arrayInMateria = new ArrayList<String>();
17          ArrayList<String> arrayIDMateria = new ArrayList<String>();
18          ArrayList<String> arrayNomeMateria = new ArrayList<String>();
19          ArrayList<String> arrayBimestre = new ArrayList<String>();
20          ArrayList<String> arrayNotas = new ArrayList<String>();
21          ArrayList<String> arrayFaltas = new ArrayList<String>();
22          ArrayList<String> arrayNovasNotas = new ArrayList<String>();
23          ArrayList<String> arrayAtuaNotas = new ArrayList<String>();
24
25          public ConexaoMYSQL() {
26                  conexao = new ConnectionFactory().getConnection();
27          }
28
29          public String trataLogin(String rgm, String senha, ArrayList<String> array){
30                  String sql = "SELECT * FROM ALUNO WHERE RGM = ? AND SENHA = ?";
31                  String sql_retorno = "@nao_possui_login";
32
33                  try {
34                          PreparedStatement stm = conexao.prepareStatement(sql);
35                          stm.setString(1, rgm);
36                          stm.setString(2, senha);
37                          ResultSet rs = stm.executeQuery();
38                          //Banco mestre possui dados do usuario
39                          if(rs != null){
40                                  while(rs.next()){
41                                          sql_retorno = "@possui_login";
42                                          //Android possui dados do usuario (realiza uma comparacao)
43                                          SimpleDateFormat  df  =  new  SimpleDateFormat("yyyy-MM-dd
44  HH:mm:ss");
45                                          String                string_data_tabela                =
46  df.format(rs.getTimestamp("DATA_ATUALIZACAO"));
47
48                                          if(array!=null){
49                                                  /* Pega a data do banco mestre */
50                                                  Date data_tabela, data_android;
```

```
                                                try {
                                                        data_tabela                                 =
df.parse(string_data_tabela);
                                                        data_android = df.parse(array.get(5));

                                                        //Dados do banco mestre mais atual que o
do Android
                                                        if(data_tabela.after(data_android)){
                                                                sql_retorno  =  "UPDATE  ALUNO  SET
NOME='"+rs.getString("NOME")+"', ENDERECO='"

        +rs.getString("ENDERECO")+"', EMAIL='"+rs.getString("EMAIL")+"', DATA_ATUALIZACAO='"

        +string_data_tabela+"' WHERE RGM='"+rs.getString("RGM")+"' AND SENHA='"

        +rs.getString("SENHA")+"'";

                                                        }
                                                } catch (ParseException e) {
                                                        // TODO Auto-generated catch block
                                                        e.printStackTrace();
                                                }
                                        }
                                        //Cliente nao possui cadastro do aluno
                                        else{
                                                sql_retorno   =   "INSERT   INTO   ALUNO   VALUES
('"+rs.getString("RGM")+"', '"+rs.getString("NOME")+
                                                                "',
'"+rs.getString("ENDERECO")+"', '"+rs.getString("SENHA")+"', '"+rs.getString("EMAIL")+
                                                                "', '"+string_data_tabela+"')";
                                        }

                                        //System.out.println(sql_retorno);
                                        return sql_retorno;
                                }
                        }
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                return sql_retorno;
        }

        public String trataInformacoes(String rgm, String senha, ArrayList<String> array){
                String sql = "SELECT * FROM ALUNO WHERE RGM = ? AND SENHA = ?";
                String sql_retorno = "@nao_possui_informacoes";
                arrayInfo.clear();
                try {
                        PreparedStatement stm = conexao.prepareStatement(sql);
                        stm.setString(1, rgm);
                        stm.setString(2, senha);
                        //System.out.println(rgm + " " + senha);
                        ResultSet rs = stm.executeQuery();
                        //Banco mestre possui dados do usuario
                        if(rs != null){
                                while(rs.next()){
                                        sql_retorno = "@possui_informacoes";
                                        //Android possui dados do usuario (realiza uma comparacao)
                                        SimpleDateFormat  df  =  new  SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                                        String                string_data_tabela                =
df.format(rs.getTimestamp("DATA_ATUALIZACAO"));

                                        if(array!=null){
                                                /* Pega a data do banco mestre */
                                                Date data_tabela, data_android;
                                                try {
```

```
                                                        data_tabela                                     =
df.parse(string_data_tabela);
                                                        data_android = df.parse(array.get(5));

                                                        //Dados do banco mestre mais atual que o
do Android
                                                        if(data_tabela.after(data_android)){
                                                                sql_retorno  =  "UPDATE  ALUNO  SET
NOME='"+rs.getString("NOME")+"', ENDERECO='"

        +rs.getString("ENDERECO")+"', EMAIL='"+rs.getString("EMAIL")+"', DATA_ATUALIZACAO='"

        +string_data_tabela+"' WHERE RGM='"+rs.getString("RGM")+"' AND SENHA='"

        +rs.getString("SENHA")+"'";
                                                        }
                                                } catch (ParseException e) {
                                                        // TODO Auto-generated catch block
                                                        e.printStackTrace();
                                                }
                                        }
                                        //Cliente nao possui cadastro do aluno
                                        else{
                                                sql_retorno   =   "INSERT   INTO   ALUNO   VALUES
('"+rs.getString("RGM")+"', '"+rs.getString("NOME")+
                                                                "',
'"+rs.getString("ENDERECO")+"', '"+rs.getString("SENHA")+"', '"+rs.getString("EMAIL")+
                                                                "', '"+string_data_tabela+"')";
                                        }
                                        arrayInfo.add(0, rs.getString("RGM"));
                                        arrayInfo.add(1, rs.getString("NOME"));
                                        arrayInfo.add(2, rs.getString("ENDERECO"));
                                        arrayInfo.add(3, rs.getString("EMAIL"));
                                        arrayInfo.add(4, string_data_tabela);

                                        return sql_retorno;
                                }
                        }
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                return sql_retorno;
        }

        public    void    trataMateria(String    rgm_cliente,    ArrayList<String>    cArrayMatID,
ArrayList<String> cArrayMatData){
                String sql = "SELECT DISTINCT notas.ID_MATERIA as ID_MATERIA, materia.NOME_MATERIA
as NOME_MATERIA, " +
                                "professor.ID_PROFESSOR   as   ID_PROFESSOR,   professor.NOME   as
NOME_PROFESSOR, professor.SENHA " +
                                "as      SENHA,      materia.DATA_ATUALIZACAO      as      DATAMAT,
professor.DATA_ATUALIZACAO as DATAPROF " +
                                "FROM  NOTAS,  MATERIA,  PROFESSOR  WHERE  notas.ID_ALUNO=?  AND
notas.ID_MATERIA=materia.ID_MATERIA and" +
                                " materia.ID_PROFESSOR=professor.ID_PROFESSOR";
                arrayProf.clear();
                arrayAtMateria.clear();
                arrayInMateria.clear();
                arrayIDMateria.clear();
                arrayNomeMateria.clear();
                int size_mat, i;
                String idmat, idprof;
                boolean encontrado;

                try {
                        PreparedStatement stm = conexao.prepareStatement(sql);
```

```
                        stm.setString(1, rgm_cliente);
                        ResultSet rs = stm.executeQuery();


                        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                        size_mat = cArrayMatID.size();

                        rs.beforeFirst();
                        while(rs.next()){
                                String                  string_mat_data_tabela                  =
df.format(rs.getTimestamp("DATAMAT"));
                                String                  string_prof_data_tabela                 =
df.format(rs.getTimestamp("DATAPROF"));
                                try {
                                        Date mat_data_tabela = df.parse(string_mat_data_tabela);
                                        Date prof_data_tabela = df.parse(string_prof_data_tabela);

                                        idmat = rs.getString("ID_MATERIA");
                                        idprof = rs.getString("ID_PROFESSOR");
                                        encontrado = false;

                                        i=0;
                                        //Tenta encontrar a materia na lista de materia
                                        while(i<size_mat && encontrado==false){
                                                if(cArrayMatID.get(i).equals(idmat)){
                                                        encontrado=true;
                                                }
                                                else
                                                        i++;
                                        }
                                        //Se encontrou compara as datas de atualizacao
                                        if(encontrado==true){
                                                Date                and_mat_data                =
df.parse(cArrayMatData.get(i));
                                                if(mat_data_tabela.after(and_mat_data)){
                                                        arrayAtMateria.add("UPDATE   materia   SET
NOME_MATERIA='"+rs.getString("materia.NOME_MATERIA")+
                                                                                "'
DATA_ATUALIZACAO='"+string_mat_data_tabela+"' WHERE" +
                                                                                "
materia.ID_MATERIA='"+cArrayMatID.get(i)+"'");
                                                }
                                        }
                                        //Senao, insere professor e materia
                                        else{
                                                String  sqlInsertProf  =  "INSERT  OR  IGNORE  INTO
professor (ID_PROFESSOR, NOME, SENHA, DATA_ATUALIZACAO) " +
                                                                        "SELECT  ?,  ?,  ?,  ?  WHERE  NOT
EXISTS (SELECT * FROM PROFESSOR WHERE ID_PROFESSOR=?) LIMIT 1;";
                                                PreparedStatement               st              =
conexao.prepareStatement(sqlInsertProf);
                                                st.setString(1, rs.getString("ID_PROFESSOR"));
                                                st.setString(2, rs.getString("NOME_PROFESSOR"));
                                                st.setString(3, rs.getString("SENHA"));
                                                st.setString(4, string_prof_data_tabela);
                                                st.setString(5, rs.getString("ID_PROFESSOR"));
                                                String aux = st.toString();
                                                String  novoSql  =  aux.substring(aux.indexOf(  ": "
)+2);
                                                arrayProf.add(novoSql);
                                                st.close();

                                                String sqlInsertMateria = "INSERT OR IGNORE INTO
materia (ID_MATERIA, NOME_MATERIA, ID_PROFESSOR, DATA_ATUALIZACAO)" +
                                                                        "  SELECT  ?,  ?,  ?,  ?  WHERE  NOT
EXISTS (SELECT * FROM materia WHERE ID_MATERIA=? AND ID_PROFESSOR=?) LIMIT 1;";
                                                st = conexao.prepareStatement(sqlInsertMateria);
```

```
252                                             st.setString(1, rs.getString("ID_MATERIA"));
253                                             st.setString(2, rs.getString("NOME_MATERIA"));
254                                             st.setString(3, rs.getString("ID_PROFESSOR"));
255                                             st.setString(4, string_mat_data_tabela);
256                                             st.setString(5, rs.getString("ID_MATERIA"));
257                                             st.setString(6, rs.getString("ID_PROFESSOR"));
258                                             aux = st.toString();
259                                             novoSql = aux.substring(aux.indexOf( ": " )+2);
260                                             arrayInMateria.add(novoSql);
261                                             st.close();
262
263                                     }
264
265                                     } catch (ParseException e) {
266                                             // TODO Auto-generated catch block
267                                             e.printStackTrace();
268                                     }
269                                     arrayIDMateria.add(rs.getString("ID_MATERIA"));
270                                     arrayNomeMateria.add(rs.getString("NOME_MATERIA"));
271                             }
272                             rs.close();
273                             stm.close();
274                     } catch (SQLException e) {
275                             // TODO Auto-generated catch block
276                             e.printStackTrace();
277                     }
278                     //System.out.println(arrayIDMateria.size());
279                     //System.out.println(arrayNomeMateria.size());
280             }
281
282             public  void  trataNotas(String  rgm_cliente,  String  idMat,  ArrayList<String>  arrayBim,
283     ArrayList<String> arrayData){
284                     arrayBimestre.clear();
285                     arrayNotas.clear();
286                     arrayFaltas.clear();
287                     arrayNovasNotas.clear();
288                     arrayAtuaNotas.clear();
289
290                     boolean encontrado;
291
292                     String sql = "SELECT * FROM NOTAS WHERE ID_ALUNO=? AND ID_MATERIA=?";
293                     try {
294                             PreparedStatement stm = conexao.prepareStatement(sql);
295                             stm.setString(1, rgm_cliente);
296                             stm.setString(2, idMat);
297
298                             ResultSet rs = stm.executeQuery();
299                             SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
300
301                             while(rs.next()){
302                                     encontrado = false;
303                                     String bim = rs.getString("BIMESTRE");
304                                     String                      StringDataTabela                      =
305     df.format(rs.getTimestamp("DATA_ATUALIZACAO"));
306                                     int i=0, tam = arrayBim.size();
307
308                                     try {
309                                             Date dataTabela = df.parse(StringDataTabela);
310                                             while(i<tam && encontrado==false){
311                                                     if(arrayBim.get(i).equals(bim)){
312                                                             encontrado=true;
313                                                     }
314                                                     else{
315                                                             i++;
316                                                     }
317                                             }
318                                             if(encontrado==true){
```

```
                                                Date dataAndroid = df.parse(arrayData.get(i));
                                                if(dataTabela.after(dataAndroid)){
                                                        arrayAtuaNotas.add("UPDATE    notas    SET
NOTA='"+ rs.getString("NOTA")+"', FALTAS='" +

        rs.getString("FALTAS")+"', DATA_ATUALIZACAO='"+StringDataTabela+"' WHERE ID_ALUNO='"+
                                                                                rgm_cliente+"'         AND
ID_MATERIA='"+idMat+"' AND BIMESTRE='"+bim+"'");
                                                }
                                        }
                                        else{
                                                String  sqlIns  =  "INSERT  OR  IGNORE  INTO  notas
(BIMESTRE, ID_ALUNO, ID_MATERIA, NOTA, FALTAS," +
                                                                " DATA_ATUALIZACAO) SELECT ?, ?,
?, ?, ?, ? WHERE NOT EXISTS (SELECT * FROM notas WHERE" +
                                                                "  BIMESTRE=?  AND  ID_ALUNO=?  AND
ID_MATERIA=?) LIMIT 1;";

                                                PreparedStatement               st               =
conexao.prepareStatement(sqlIns);
                                                st.setString(1, bim);
                                                st.setString(2, rgm_cliente);
                                                st.setString(3, idMat);
                                                st.setString(4, rs.getString("NOTA"));
                                                st.setString(5, rs.getString("FALTAS"));
                                                st.setString(6, StringDataTabela);
                                                st.setString(7, bim);
                                                st.setString(8, rgm_cliente);
                                                st.setString(9, idMat);
                                                String aux = st.toString();
                                                String  novoSql  =  aux.substring(aux.indexOf(  ": "
)+2);
                                                arrayNovasNotas.add(novoSql);
                                                st.close();

                                        }

                                } catch (ParseException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                                arrayBimestre.add(rs.getString("BIMESTRE"));
                                arrayNotas.add(rs.getString("NOTA"));
                                arrayFaltas.add(rs.getString("FALTAS"));

                        }
                        rs.close();
                        stm.close();
                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
}
```

D.2 Classe SocketMainServer

```
1  import java.io.BufferedReader;
2  import java.io.BufferedWriter;
3  import java.io.IOException;
```

```
4   import java.io.InputStreamReader;
5   import java.io.ObjectInputStream;
6   import java.io.ObjectOutputStream;
7   import java.io.OutputStreamWriter;
8   import java.io.PrintWriter;
9   import java.net.ServerSocket;
10  import java.net.Socket;
11  import java.util.ArrayList;
12
13  public class SocketMainServer extends Thread{
14          private int SERVERPORT = 5657;
15          private ServerSocket serverSocket;
16          private static boolean running = false;
17          final static String mensagem_final = "@finish";
18
19          OnMessageReceived messageListener;
20          static ConexaoMYSQL conexaoBD;
21
22          public SocketMainServer(OnMessageReceived messageListener){
23                  this.messageListener = messageListener;
24          }
25
26          /* abre o socket servidor na porta SERVERPORT */
27          public void openSocket(){
28                  if(serverSocket==null){
29                          running = true;
30                          try {
31                                  serverSocket = new ServerSocket(SERVERPORT);
32                          } catch (IOException e) {
33                                  e.printStackTrace();
34                          }
35                  }
36          }
37
38          /* abre um thread para cada cliente */
39          public synchronized void run(){
40
41                  try {
42                          while(running){
43                                  Socket cliente = serverSocket.accept();
44
45                                  ManagerClient trataCliente = new ManagerClient(cliente);
46
47                                  Thread t = new Thread(trataCliente);
48                                  t.start();
49                          }
50                  } catch (IOException e) {
51                          e.printStackTrace();
52                  }
53                  finally{
54                          try {
55                                  serverSocket.close();
56                          } catch (IOException e) {
57                                  e.printStackTrace();
58                          }
59                  }
60          }
61
62          public void stopServer(){
63                  if(serverSocket!=null && running==true){
64                          running=false;
65                          try {
66                                  serverSocket.close();
67                          } catch (IOException e) {
68                                  e.printStackTrace();
69                          }
70                  }
```

```
71          }
72
73          public class ManagerClient extends Thread{
74                  Socket clienteSocket;
75                  PrintWriter out;
76                  ObjectOutputStream objectOutput;
77                  ObjectInputStream objectInput;
78
79
80                  public ManagerClient(Socket cliente){
81                          clienteSocket = cliente;
82                  }
83
84                  @Override
85                  public synchronized void run(){
86                          try {
87                                  out      =      new      PrintWriter(new      BufferedWriter(new
88  OutputStreamWriter(clienteSocket.getOutputStream())), true);
89                                  BufferedReader       in       =       new       BufferedReader(new
90  InputStreamReader(clienteSocket.getInputStream()));
91                                  objectOutput                          =                          new
92  ObjectOutputStream(clienteSocket.getOutputStream());
93                                  objectInput                          =                          new
94  ObjectInputStream(clienteSocket.getInputStream());
95                                  String rgm_cliente, senha_cliente;
96                                  System.out.println("Cliente connectado com servidor");
97
98                                  rgm_cliente = in.readLine();
99                                  senha_cliente = in.readLine();
100                                 String msg_cliente = in.readLine();
101                                 messageListener.messageReceived("Consulta recebida do cliente: " +
102 msg_cliente);
103                                 System.out.println("Consulta    recebida    do    cliente:    "    +
104 msg_cliente);
105
106                                 if(msg_cliente.equals("@solicitacao_login")){
107
108                                         msg_cliente = in.readLine();
109                                         ArrayList<String> array = null;
110                                         if(msg_cliente.equals("@com_login")){
111                                                 try {
112                                                         Object obj = objectInput.readObject();
113                                                         array = (ArrayList<String>) obj;
114                                                 } catch (ClassNotFoundException e) {
115                                                         // TODO Auto-generated catch block
116                                                         e.printStackTrace();
117                                                 }
118                                         }
119                                         //System.out.println(msg_cliente);
120                                         String     log     =     conexaoBD.trataLogin(rgm_cliente,
121 senha_cliente, array);
122
123                                         sendMessage(log);
124                                         sendMessage(mensagem_final);
125                                 }
126                                 if(msg_cliente.equals("@solicitacao_informacoes")){
127
128                                         msg_cliente = in.readLine();
129                                         //System.out.println(msg_cliente);
130                                         ArrayList<String> array = new ArrayList<String>();
131                                         if(msg_cliente.equals("@com_informacoes")){
132                                                 try {
133                                                         Object obj = objectInput.readObject();
134                                                         array = (ArrayList<String>) obj;
135                                                 } catch (ClassNotFoundException e) {
136                                                         // TODO Auto-generated catch block
137                                                         e.printStackTrace();
```

```
138                                             }
139                                     }
140                                     String   inf   =   conexaoBD.trataInformacoes(rgm_cliente,
141     senha_cliente, array);
142                                     sendMessage(inf);
143
144                                     if(!inf.contains("@nao_possui_informacoes")){
145                                             ArrayList<String> novo = conexaoBD.getArrayInfo();
146                                             sendMessage("@listaInformacoes");
147                                             while(!novo.isEmpty()){
148                                                     sendMessage(novo.get(0));
149                                                     novo.remove(0);
150                                             }
151                                     }
152                                     sendMessage(mensagem_final);
153                             }
154                             if(msg_cliente.equals("@solicitacao_materia")){
155
156                                     ArrayList<String> arrayMatID = new ArrayList<String>();
157                                     ArrayList<String> arrayData = new ArrayList<String>();
158                                     Object objMat, objData;
159                                     try {
160                                             objMat = objectInput.readObject();
161                                             objData = objectInput.readObject();
162
163                                             arrayMatID = (ArrayList<String>) objMat;
164                                             arrayData = (ArrayList<String>) objData;
165                                     } catch (ClassNotFoundException e) {
166                                             // TODO Auto-generated catch block
167                                             e.printStackTrace();
168                                     }
169                                     conexaoBD.trataMateria(rgm_cliente,            arrayMatID,
170     arrayData);
171                                     ArrayList<String> arrayProf = conexaoBD.getArrayProf();
172                                     ArrayList<String>              arrayInsMat              =
173     conexaoBD.getArrayInMateria();
174                                     ArrayList<String>              arrayAtuaMat             =
175     conexaoBD.getArrayAtMateria();
176                                     ArrayList<String>               arrayIDMat              =
177     conexaoBD.getArrayIDMateria();
178                                     ArrayList<String>              arrayNomeMat             =
179     conexaoBD.getArrayNomeMateria();
180                                     sendArray(arrayProf);
181                                     sendArray(arrayInsMat);
182                                     sendArray(arrayAtuaMat);
183                                     sendArray(arrayIDMat);
184                                     sendArray(arrayNomeMat);
185
186                             }
187                             if(msg_cliente.equals("@solicitacao_notas")){
188                                     String idMat = in.readLine();
189                                     //String nomeMat = in.readLine();
190
191                                     ArrayList<String> arrayBim = new ArrayList<String>();
192                                     ArrayList<String> arrayData = new ArrayList<String>();
193                                     Object objBim, objData;
194                                     try {
195                                             objBim = objectInput.readObject();
196                                             objData = objectInput.readObject();
197                                             arrayBim = (ArrayList<String>) objBim;
198                                             arrayData = (ArrayList<String>) objData;
199
200                                     } catch (ClassNotFoundException e) {
201                                             // TODO Auto-generated catch block
202                                             e.printStackTrace();
203                                     }
```

```
                                        conexaoBD.trataNotas(rgm_cliente,     idMat,     arrayBim,
arrayData);

                                        ArrayList<String>              arrayBimestre             =
conexaoBD.getArrayBimestre();
                                        ArrayList<String> arrayNotas = conexaoBD.getArrayNotas();
                                        ArrayList<String>               arrayFaltas              =
conexaoBD.getArrayFaltas();
                                        ArrayList<String>             arrayNovasNotas            =
conexaoBD.getArrayNovasNotas();
                                        ArrayList<String>             arrayAtualNotas            =
conexaoBD.getArrayAtuaNotas();
                                        sendArray(arrayNovasNotas);
                                        sendArray(arrayAtualNotas);
                                        sendArray(arrayBimestre);
                                        sendArray(arrayNotas);
                                        sendArray(arrayFaltas);

                                }
                                in.close();
                                objectInput.close();
                                out.close();
                                objectOutput.close();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                        finally{
                                try {
                                        clienteSocket.close();
                                        messageListener.messageReceived("Conexao   terminada   com
cliente");
                                        System.out.println("Conexao terminada com cliente");
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                }

                /**
                 * Envia mensagem para o cliente
                 * @param message mensagem a ser enviada
                 */
                public void sendMessage(String message){
                        try{
                                if (out != null && !out.checkError()){
                                        /* Envia pro cliente */
                                        out.println(message);
                                        out.flush();
                                }
                        }
                        catch (Exception e){
                                e.printStackTrace();
                        }
                }

                /**
                 * Envia mensagem para o cliente
                 * @param message mensagem a ser enviada
                 */
                public void sendArray(ArrayList<String> array){
                        if(objectOutput != null){
                                try {
                                        objectOutput.writeObject(array);
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
```

```
271                         }
272                 }
273         }
274
275         public static void main(String[] args) {
276                 conexaoBD = new ConexaoMYSQL();
277
278                 ServerInterface teste = new ServerInterface();
279                 teste.setVisible(true);
280         }
```

Apêndice E

Classes implementadas no cliente Android

E.1 Classe clientSocket

```
1   package com.example.clientsocketmain;
2
3   import java.io.BufferedReader;
4   import java.io.BufferedWriter;
5   import java.io.IOException;
6   import java.io.InputStreamReader;
7   import java.io.ObjectInputStream;
8   import java.io.ObjectOutputStream;
9   import java.io.OutputStreamWriter;
10  import java.io.PrintWriter;
11  import java.net.InetAddress;
12  import java.net.InetSocketAddress;
13  import java.net.Socket;
14  import java.net.UnknownHostException;
15  import java.util.ArrayList;
16
17  import android.util.Log;
18
19  public class clientSocket extends Thread{
20          boolean running = false, verificou_login = false, logado = false;
21          int porta = 5657, tam_msg;
22          String endereco_ip = "192.168.1.2"; // your computer IP address
23          static private PrintWriter out = null;
24          static private BufferedReader in = null;
25      static private ObjectOutputStream objectOutput = null;
26      private ObjectInputStream objectInput = null;
27      private String sql_solicitada;
28          private String usuario_rgm;
29          private String usuario_senha;
30      static private String msg_do_server;
31      Socket socket;
32      DatabaseConnection db;
33      final static String mensagem_final = "@finish";
34      boolean falha_conexao;
35      OnMessageReceived MessageListener;
36      ArrayList<String> arrayInf = new ArrayList<String>();
37
38          public  clientSocket(String  sql_atualizacao,  String  usuario_rgm,  String  usuario_senha,
39  DatabaseConnection db, final OnMessageReceived listener){
40                  this.MessageListener = listener;
41          this.sql_solicitada = sql_atualizacao;
42          this.usuario_senha = usuario_senha;
43          this.usuario_rgm = usuario_rgm;
44          this.db = db;
45          this.falha_conexao = false;
46      }
47
48          public void sendMessage(String message){
49          if (out != null && !out.checkError()) {
50              out.println(message);
51              out.flush();
```

```
52          }
53      }
54          public void sendArray(ArrayList<String> array){
55                  if(objectOutput != null){
56                          try {
57                                  objectOutput.writeObject(array);
58                          } catch (IOException e) {
59                                  // TODO Auto-generated catch block
60                                  e.printStackTrace();
61                          }
62                  }
63          }
64
65          @Override
66          public void run() {
67                  falha_conexao=false;
68                  try {
69                          InetAddress servAddr;
70                          servAddr = InetAddress.getByName(endereco_ip);
71                          socket = new Socket();
72                          socket.connect(new InetSocketAddress(servAddr, porta), 1000*5);
73
74                          if(socket!=null){
75                                  running=true;
76                                  verificou_login = false;
77                                  logado = false;
78                          }
79
80                          out         =         new         PrintWriter(new         BufferedWriter(new
81  OutputStreamWriter(socket.getOutputStream())), true);
82                          in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
83
84                          objectOutput = new ObjectOutputStream(socket.getOutputStream());
85                          objectInput = new ObjectInputStream(socket.getInputStream());
86
87                          sendMessage(usuario_rgm);
88                          sendMessage(usuario_senha);
89                          sendMessage(sql_solicitada);
90                          if(sql_solicitada.equals("@solicitacao_login")){
91
92                                  ArrayList<String>       obj=       db.getAluno(db.getConnection(),
93  usuario_rgm, usuario_senha);
94                                  if(obj != null){
95                                          sendMessage("@com_login");
96                                          sendArray(obj);
97                                  }
98                                  else{
99                                          sendMessage("@sem_login");
100                                 }
101                                 while(running){
102                                         msg_do_server = in.readLine();
103                                         //Log.e("msg_do_server", msg_do_server);
104                                         verificou_login = true;
105                                         if(!msg_do_server.contains("@nao_possui_login")         &&
106 !msg_do_server.equals(mensagem_final)){
107                                                 logado = true;
108                                                 if(!msg_do_server.contains("@possui_login")){
109                                                         if(MessageListener!=null){
110
111         MessageListener.messageReceived(msg_do_server);
112                                                         }
113                                                         else{
114                                                                 db.executaSQL(db.getConnection(),
115 msg_do_server);
116                                                         }
117                                                 }
118                                         }
```

```
                                        if(msg_do_server.equals(mensagem_final)){
                                                stopClient();
                                        }
                                }
                        }
                        if(sql_solicitada.equals("@solicitacao_informacoes")){

                                ArrayList<String>       obj=       db.getAluno(db.getConnection(),
usuario_rgm, usuario_senha);

                                if(obj != null){
                                        sendMessage("@com_informacoes");
                                        sendArray(obj);
                                }
                                else{
                                        sendMessage("@sem_informacoes");
                                }
                                while(running){
                                        msg_do_server = in.readLine();
                                        Log.e("msg", msg_do_server);
                                        if(!msg_do_server.equals(mensagem_final)                &&
!msg_do_server.equals("@nao_possui_informacoes")){
                                                if(msg_do_server.contains("INSERT")             ||
msg_do_server.contains("UPDATE")){
                                                        db.executaSQL(db.getConnection(),
msg_do_server);
                                                }
                                                if(msg_do_server.contains("@listaInformacoes")){
                                                        String aux = in.readLine();
                                                        ArrayList<String>     array     =     new
ArrayList<String>();
                                                        while(!aux.equals(mensagem_final)){
                                                                array.add(aux);
                                                                aux = in.readLine();
                                                        }
                                                        arrayInf       =       (ArrayList<String>)
array.clone();
                                                        msg_do_server = aux;
                                                }
                                        }
                                        if(msg_do_server.equals(mensagem_final)){
                                                stopClient();
                                        }
                                }
                                if(arrayInf.isEmpty()==false){
                                        MessageListener.messageReceived(arrayInf);
                                }
                        }
                        if(sql_solicitada.equals("@solicitacao_materia")){

                                ArrayList<String> objMat = db.getMateriaOrData(db.getConnection(),
usuario_rgm, 0);
                                ArrayList<String>                    objData                    =
db.getMateriaOrData(db.getConnection(), usuario_rgm, 1);
                                sendArray(objMat);
                                sendArray(objData);
                                Object objprof, objInsMat, objAtuaMat, objIDMat, objNomeMat;

                                ArrayList<String> arrayProf = new ArrayList<String>();
                                ArrayList<String> arrayInsMat = new ArrayList<String>();
                                ArrayList<String> arrayAtuaMat = new ArrayList<String>();
                                ArrayList<String> arrayIDMat = new ArrayList<String>();
                                ArrayList<String> arrayNomeMat = new ArrayList<String>();

                                try {
                                        objprof = objectInput.readObject();
                                        objInsMat = objectInput.readObject();
```

```
                                        objAtuaMat = objectInput.readObject();
                                        objIDMat = objectInput.readObject();
                                        objNomeMat = objectInput.readObject();
                                        //executar a sql dos 3 primeiro e passar 2 por param
                                        arrayProf = (ArrayList<String>) objprof;
                                        arrayInsMat = (ArrayList<String>) objInsMat;
                                        arrayAtuaMat = (ArrayList<String>) objAtuaMat;
                                        arrayIDMat = (ArrayList<String>) objIDMat;
                                        arrayNomeMat = (ArrayList<String>) objNomeMat;
                                } catch (ClassNotFoundException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                                while(!arrayProf.isEmpty()){
                                        db.executaSQL(db.getConnection(), arrayProf.get(0));
                                        arrayProf.remove(0);
                                }
                                while(!arrayInsMat.isEmpty()){
                                        db.executaSQL(db.getConnection(), arrayInsMat.get(0));
                                        arrayInsMat.remove(0);
                                }
                                while(!arrayAtuaMat.isEmpty()){
                                        db.executaSQL(db.getConnection(), arrayAtuaMat.get(0));
                                        arrayAtuaMat.remove(0);
                                }
                                MessageListener.messageReceived(arrayIDMat);
                                MessageListener.messageReceived(arrayNomeMat);
                        }
                        if(sql_solicitada.equals("@solicitacao_notas")){
                                sendMessage(db.getIdMateria());
                                //sendMessage(db.getNomeMateria());
                                ArrayList<String>                       objBim                  =
db.getBimOrDataAtual(db.getConnection(), usuario_rgm, db.getIdMateria(), 0);
                                ArrayList<String>                      objData                  =
db.getBimOrDataAtual(db.getConnection(), usuario_rgm, db.getIdMateria(), 1);

                                sendArray(objBim);
                                sendArray(objData);
                                Object   objBimestre,   objNotas,   objAtuaNotas,   objNovasNotas,
objFaltas;

                                ArrayList<String> arrayBim = new ArrayList<String>();
                                ArrayList<String> arrayNotas = new ArrayList<String>();
                                ArrayList<String> arrayAtuaNotas = new ArrayList<String>();
                                ArrayList<String> arrayNovasNotas = new ArrayList<String>();
                                ArrayList<String> arrayFaltas = new ArrayList<String>();

                                try {
                                        objNovasNotas = objectInput.readObject();
                                        objAtuaNotas = objectInput.readObject();
                                        objBimestre = objectInput.readObject();
                                        objNotas = objectInput.readObject();
                                        objFaltas = objectInput.readObject();

                                        arrayNovasNotas = (ArrayList<String>) objNovasNotas;
                                        arrayAtuaNotas = (ArrayList<String>) objAtuaNotas;
                                        arrayBim = (ArrayList<String>) objBimestre;
                                        arrayNotas = (ArrayList<String>) objNotas;
                                        arrayFaltas = (ArrayList<String>) objFaltas;
                                } catch (ClassNotFoundException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                                while(!arrayNovasNotas.isEmpty()){
                                        db.executaSQL(db.getConnection(), arrayNovasNotas.get(0));
                                        arrayNovasNotas.remove(0);
                                }
```

```
253                                 while(!arrayAtuaNotas.isEmpty()){
254                                         db.executaSQL(db.getConnection(), arrayAtuaNotas.get(0));
255                                         arrayAtuaNotas.remove(0);
256                                 }
257                                 MessageListener.messageReceived(arrayBim);
258                                 MessageListener.messageReceived(arrayNotas);
259                                 MessageListener.messageReceived(arrayFaltas);
260                         }
261                         in.close();
262                         objectInput.close();
263                         out.close();
264                         objectOutput.close();
265                 } catch (UnknownHostException e) {
266                         this.falha_conexao=true;
267                         Log.e("UnknownHostException", "Host não encontrado");
268                 } catch (IOException e) {
269                         this.falha_conexao=true;
270                         Log.e("IOException", "Erro de I/O");
271                         e.printStackTrace();
272                 } catch(IllegalArgumentException e){
273                         this.falha_conexao=true;
274                         Log.e("IllegalArgumentException", "Timeout socket");
275                 }finally{
276                         try {
277                                 socket.close();
278                         } catch (IOException e) {
279                                 Log.e("IOException", "Exceção gerada por um erro de I/O");
280                         }
281                 }
282         }
283 }
```

E. 2 Classe DatabaseConnection

```
1  package com.example.clientsocketmain;
2
3  import java.util.ArrayList;
4
5  import android.database.Cursor;
6  import android.database.sqlite.SQLiteDatabase;
7  import android.util.Log;
8
9  public class DatabaseConnection {
10         DatabaseManager bancoManager;
11         SQLiteDatabase sqld;
12         String nomeMateria;
13         String idMateria;
14
15         public DatabaseConnection(DatabaseManager bancoManager) {
16                 this.bancoManager = bancoManager;
17         }
18
19         /* Cria ou abre a conexao */
20         public void openConnection() {
21                 sqld = bancoManager.getWritableDatabase();
22         }
23
24         /* Pega a conexao do banco de dados */
25         public SQLiteDatabase getConnection() {
26                 if(sqld!=null && sqld.isOpen()){
27                         return sqld;
28                 }
29                 return null;
30         }
```

```
31
32          /* Fecha a conexao do banco de dados */
33          public void closeConnection(){
34                  bancoManager.close();
35          }
36
37          public ArrayList<String> getAluno(SQLiteDatabase conexaodb, String rgm, String senha){
38                  ArrayList<String> array = new ArrayList<String>();
39                  Cursor cursor = conexaodb.rawQuery("SELECT * from ALUNO WHERE RGM = ? AND SENHA =
40  ?", new String[]{rgm, senha});
41                  cursor.moveToFirst();
42                  if(cursor.getCount() > 0){
43                          array.add(0, cursor.getString(cursor.getColumnIndex("RGM")));
44                          array.add(1, cursor.getString(cursor.getColumnIndex("NOME")));
45                          array.add(2, cursor.getString(cursor.getColumnIndex("ENDERECO")));
46                          array.add(3, cursor.getString(cursor.getColumnIndex("SENHA")));
47                          array.add(4, cursor.getString(cursor.getColumnIndex("EMAIL")));
48                          array.add(5, cursor.getString(cursor.getColumnIndex("DATA_ATUALIZACAO")));
49                          return array;
50                  }
51                  cursor.close();
52                  return null;
53          }
54
55          public ArrayList<String> getMateriaOrData(SQLiteDatabase conexaodb, String rgm, int tipo){
56                  ArrayList<String> aux = new ArrayList<String>();
57                  Cursor   cursor   =   conexaodb.rawQuery("SELECT   DISTINCT   notas.ID_MATERIA   as
58  ID_MATERIA, materia.NOME_MATERIA as NOME_MATERIA," +
59                                  " materia.DATA_ATUALIZACAO as DATA_ATUALIZACAO" +
60                                  "   from   NOTAS,   MATERIA   WHERE   notas.ID_ALUNO   =   ?   AND
61  notas.ID_MATERIA=materia.ID_MATERIA", new String[]{rgm});
62
63                  int i=0;
64                  int tam = cursor.getCount();
65                  cursor.moveToFirst();
66                  while(i<tam){
67                          //Envia lista de materia
68                          if(tipo==0){
69                                  aux.add(i, cursor.getString(cursor.getColumnIndex("ID_MATERIA")));
70                          }
71                          //Envia lista de datas
72                          else{
73                                  aux.add(i,
74  cursor.getString(cursor.getColumnIndex("DATA_ATUALIZACAO")));
75                          }
76                          cursor.moveToNext();
77                          i++;
78                  }
79                  cursor.close();
80
81                  return aux;
82          }
83
84          public  ArrayList<String>  getBimOrDataAtual(SQLiteDatabase  conexaodb,  String  rgm,  String
85  idMat, int tipo){
86                  ArrayList<String> aux = new ArrayList<String>();
87
88                  Cursor  cursor  =  conexaodb.rawQuery("SELECT  *  FROM  NOTAS  WHERE  ID_ALUNO=?  AND
89  ID_MATERIA=?", new String[]{rgm, idMat});
90                  int i=0;
91                  int tam = cursor.getCount();
92                  cursor.moveToFirst();
93                  while(i<tam){
94                          if(tipo==0){
95                                  aux.add(i, cursor.getString(cursor.getColumnIndex("BIMESTRE")));
96                          }
97                          else{
```

```
                                aux.add(i,
cursor.getString(cursor.getColumnIndex("DATA_ATUALIZACAO")));
                        }
                        i++;
                        cursor.moveToNext();
                }
                cursor.close();
                return aux;
        }

        public void executaSQL(SQLiteDatabase conexaodb, String msg){
                conexaodb.execSQL(msg);
        }
}
```

E.3 Classe InfoActivity

```
package com.example.clientsocketmain;

import java.util.ArrayList;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class InfoActivity extends Activity {

        String usuario_rgm;
        TextView textoRGM, textoNome, textoEndereco, textoEmail;
        DatabaseConnection db;
        Button buttonInfoVoltar;
        boolean conexaoInternet, conexaoServidor;
        ArrayList<String> array;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_info);

                Intent intent = getIntent();
                Bundle params = intent.getExtras();
                if(params!=null){
                        usuario_rgm = params.getString("field_rgm");
                        conexaoInternet = params.getBoolean("conexaoInternet");
                        conexaoServidor = params.getBoolean("conexaoServidor");
                        if(conexaoInternet==true && conexaoServidor==true){
                                array = params.getStringArrayList("arrayInfo");
                        }
                }

                textoRGM = (TextView)findViewById(R.id.textViewRGM);
                textoNome = (TextView)findViewById(R.id.textViewNome);
                textoEndereco = (TextView)findViewById(R.id.textViewEndereco);
                textoEmail = (TextView)findViewById(R.id.textViewEmail);

                db = new DatabaseConnection(new DatabaseCreateUpdate(this));
```

```
                db.openConnection();
                buscaDados(db.getConnection());

                db.closeConnection();

                buttonInfoVoltar = (Button)findViewById(R.id.buttonInfoVoltar);

                buttonInfoVoltar.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                onBackPressed();
                        }
                });

        }

        public void buscaDados(SQLiteDatabase connection){
                if(conexaoInternet==true && conexaoServidor==true){
                        textoRGM.setText("RGM: " + array.get(0));
                        textoNome.setText("Nome: " + array.get(1));
                        textoEndereco.setText("Endereço: " + array.get(2));
                        textoEmail.setText("E-mail: " + array.get(3));
                }
                else{
                        Cursor cursor = connection.rawQuery("select * from ALUNO where RGM==?",
new String[]{usuario_rgm});
                        textoRGM.setText("RGM: " + usuario_rgm);

                        while(cursor.moveToNext()){
                                textoNome.setText("Nome:                    "                    +
cursor.getString(cursor.getColumnIndex("NOME")));
                                textoEndereco.setText("Endereço:                "                +
cursor.getString(cursor.getColumnIndex("ENDERECO")));
                                textoEmail.setText("E-mail:                  "                   +
cursor.getString(cursor.getColumnIndex("EMAIL")));
                        }
                        cursor.close();
                }
        }
}
```

E.4 Classe LoginActivity

```
package com.example.clientsocketmain;

import java.sql.Connection;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;

import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
```

```
import android.os.Message;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class LoginActivity extends Activity {

        boolean conexaoInternet;
        EditText editRGM, editPassword;
        Button buttonLogin, buttonSair;
        DatabaseConnection db;
        static String login, password;
        boolean logado = false;
        clientSocket socket_do_cliente;


        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_login);

                /* Pega os dados dos componentes */
                editRGM = (EditText)findViewById(R.id.editTextRGM);
                editPassword = (EditText)findViewById(R.id.editTextPassword);
                buttonLogin = (Button)findViewById(R.id.buttonLogin);
                buttonSair = (Button)findViewById(R.id.buttonSair);

                /* Abre conexao com SQLite */
                db = new DatabaseConnection(new DatabaseCreateUpdate(this));
                db.openConnection();

                buttonLogin.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                login = editRGM.getText().toString();
                                password = editPassword.getText().toString();
                                logado = false;
                                //Tratamento de campos vazios
                                if(login.equals("") || password.equals("")){
                                        Toast.makeText(getBaseContext(),  "Todos  os  campos  devem
ser preenchidos", Toast.LENGTH_SHORT).show();
                                }
                                else{
                                        if(conexaoInternet==true){
                                                socket_do_cliente              =              new
clientSocket("@solicitacao_login", login, password, db, new clientSocket.OnMessageReceived(){

                                                        @Override
                                                        public     void     messageReceived(Object
message) {
                                                                // TODO Auto-generated method stub
                                                                Log.e("MessageReceived",  "Cliente
adicionado " + String.valueOf(message));
                                                                db.executaSQL(db.getConnection(),
String.valueOf(message));
                                                        }
                                                });
                                                socket_do_cliente.start();

        while(socket_do_cliente.getVerificou_login()==false                                             &&
socket_do_cliente.getFalha_conexao()==false){

                                                }
```

```
                                                if(socket_do_cliente.getFalha_conexao()==true){
                                                        logado = verificaLogin(db.getConnection(),
login, password);
                                                        Log.e("FalhaConexaoServidor",         "Nao
conectou ao server");
                                                        Toast.makeText(getBaseContext(), "Tentando
localmente", Toast.LENGTH_LONG).show();
                                                }
                                                else{
                                                        logado = socket_do_cliente.getLogado();

                                                }
                                        }
                                        else{
                                                logado = verificaLogin(db.getConnection(), login,
password);
                                        }
                                        if(logado == true){
                                                db.closeConnection();
                                                openNextActivity(login, password);
                                                finish();
                                        }
                                        else{
                                                Toast.makeText(getBaseContext(),       "Combinação
inválida", Toast.LENGTH_SHORT).show();
                                        }
                                }
                                editPassword.setText("");
                                editRGM.setText("");
                                editRGM.requestFocus();

                        }
                });

                buttonSair.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                finish();
                                android.os.Process.killProcess(android.os.Process.myPid());
                                onDestroy();
                        }
                });
        }

        /**
         * Método para verificar se o dispositivo possui conexão com a Internet
         * @return se existe ou não conexão
         */
        public boolean isOnline() {
            ConnectivityManager              cm              =              (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo netInfo = cm.getActiveNetworkInfo();

            if (netInfo != null && netInfo.isConnectedOrConnecting()) {
                return true;
            }
            return false;
        }

        @Override
        protected void onResume() {

                Date data_agora = new Date();
                SimpleDateFormat formata_data = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
                String data = formata_data.format(data_agora);

```

```
151                 conexaoInternet = false;
152                 conexaoInternet = isOnline();
153                 if(conexaoInternet){
154                         Log.e("Conexao Internet", data + " Conexão com internet");
155                 }
156                 else{
157                         Log.e("Conexao Internet", data + " Sem conexão com internet");
158                 }
159                 super.onResume();
160         }
161
162         @Override
163         protected void onActivityResult(int requestCode, int resultCode, Intent data) {
164                 if (requestCode == 1) {
165                         if(resultCode == RESULT_OK){
166                                 String msgSQL = "select * from ALUNO where RGM=" + login + " and
167 senha='" + password + "';";
168                                 clientSocket client;
169
170                                 client  =  new  clientSocket(msgSQL,  login,  password,  db,  new
171 clientSocket.OnMessageReceived(){
172
173                                         @Override
174                                         public void messageReceived(Object message) {
175                                                 // TODO Auto-generated method stub
176
177                                         }
178                                 });
179                                 client.start();
180                                 Toast.makeText(getBaseContext(), "Dados de cadastro atualizados",
181 Toast.LENGTH_SHORT).show();
182                         }
183                         if (resultCode == RESULT_CANCELED){
184                                 Toast.makeText(getBaseContext(),      "Combinação      inválida",
185 Toast.LENGTH_SHORT).show();
186                         }
187                 }
188                 editPassword.setText("");
189                 editRGM.setText("");
190                 editRGM.requestFocus();
191         }
192
193         public boolean verificaLogin(SQLiteDatabase connection, String login, String password){
194
195                 Cursor  cursor  =  connection.rawQuery("select  *  from  aluno  where  RGM=?  AND
196 senha=?;", new String[]{login, password});
197                 int count;
198                 count = cursor.getCount();
199                 cursor.close();
200
201                 if(count==0)
202                         return false;
203
204                 return true;
205         }
206
207         public void openNextActivity(String rgm, String password) {
208                 // TODO Auto-generated method stub
209                 Intent intent = new Intent(this, MainActivity.class);
210                 Bundle params = new Bundle();
211                 params.putString("field_rgm", rgm);
212                 params.putString("field_password", password);
213                 intent.putExtras(params);
214                 startActivity(intent);
215         }
```

E.5 Classe MainActivity

```
1   package com.example.clientsocketmain;
2
3   import java.util.ArrayList;
4
5   import android.net.ConnectivityManager;
6   import android.net.NetworkInfo;
7   import android.os.Bundle;
8   import android.app.Activity;
9   import android.content.Context;
10  import android.content.Intent;
11  import android.util.Log;
12  import android.view.Menu;
13  import android.view.View;
14  import android.widget.Button;
15  import android.widget.Toast;
16
17  public class MainActivity extends Activity {
18
19          DatabaseConnection db;
20          String usuario_rgm, usuario_senha;
21          clientSocket client;
22          Button buttonNotas, buttonInfo, buttonMainSair;
23          ArrayList<String> arrayInformacoes = new ArrayList<String>();
24          ArrayList<String> arrayIDMateria = new ArrayList<String>();
25          ArrayList<String> arrayNomeMateria = new ArrayList<String>();
26          boolean conexaoInternet;
27
28          @Override
29          protected void onCreate(Bundle savedInstanceState) {
30                  super.onCreate(savedInstanceState);
31                  setContentView(R.layout.activity_main);
32
33                  Intent intent = getIntent();
34                  Bundle params = intent.getExtras();
35                  if(params!=null){
36                          usuario_rgm = params.getString("field_rgm");
37                          usuario_senha = params.getString("field_password");
38                  }
39
40                  buttonNotas = (Button)findViewById(R.id.buttonNotas);
41                  buttonInfo = (Button)findViewById(R.id.buttonInfo);
42                  buttonMainSair = (Button)findViewById(R.id.buttonMainSair);
43
44                  db = new DatabaseConnection(new DatabaseCreateUpdate(this));
45                  db.openConnection();
46
47
48                  /* Abre a activity de notas */
49                  buttonNotas.setOnClickListener(new View.OnClickListener() {
50                          int i=0, count=0;
51                          @Override
52                          public void onClick(View v) {
53                                  if(conexaoInternet==true){
54                                          client    =    new    clientSocket("@solicitacao_materia",
55  usuario_rgm, usuario_senha, db, new clientSocket.OnMessageReceived(){
56
57                                                  @Override
```

```
                                                public void messageReceived(Object message) {
                                                        // TODO Auto-generated method stub
                                                        if(count==0){
                                                                arrayIDMateria.clear();
                                                                arrayIDMateria                  =
(ArrayList<String>) message;
                                                                count++;
                                                        }
                                                        else{
                                                                arrayNomeMateria.clear();
                                                                arrayNomeMateria                =
(ArrayList<String>) message;
                                                                count++;
                                                                i=1;
                                                        }
                                                }
                                        });
                                        client.start();
                                        while(i==0 && client.getFalha_conexao()==false){

                                        }
                                        if(i!=0){
                                                openNextActivity(v, "Materias", arrayIDMateria);

                                        }
                                        else{
                                                Toast.makeText(getBaseContext(),        "Acessando
localmente", Toast.LENGTH_SHORT).show();
                                                openNextActivity(v, "Materias", null);
                                        }
                                }
                                else{
                                        openNextActivity(v, "Materias", null);

                                }
                        }
                });

                /* Abre a activity de informacoes */
                buttonInfo.setOnClickListener(new View.OnClickListener() {
                        int i=0;
                        @Override
                        public void onClick(View v) {
                                if(conexaoInternet == true){
                                        client   =   new   clientSocket("@solicitacao_informacoes",
usuario_rgm, usuario_senha, db, new clientSocket.OnMessageReceived(){

                                                @Override
                                                public void messageReceived(Object message) {
                                                        // TODO Auto-generated method stub
                                                        arrayInformacoes   =   (ArrayList<String>)
message;
                                                        i=1;
                                                }
                                        });
                                        client.start();
                                        while(i==0 && client.getFalha_conexao()==false){

                                        }
                                        if(i!=0){
                                                openNextActivity(v,                "Informacoes",
arrayInformacoes);
                                        }
                                        else{
                                                Toast.makeText(getBaseContext(),        "Acessando
localmente", Toast.LENGTH_SHORT).show();
                                                openNextActivity(v, "Informacoes", null);
```

```
                                        }
                                }
                                else{
                                        openNextActivity(v, "Informacoes", null);
                                }
                        }
                });

                /* Fecha a tela */
                buttonMainSair.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                finish();
                                android.os.Process.killProcess(android.os.Process.myPid());
                                onDestroy();
                        }
                });
        }

        public boolean isOnline() {
            ConnectivityManager             cm              =              (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo netInfo = cm.getActiveNetworkInfo();

            if (netInfo != null && netInfo.isConnectedOrConnecting()) {
                return true;
            }
            return false;
        }

        @Override
        protected void onResume() {
                // TODO Auto-generated method stub
                conexaoInternet = isOnline();
                super.onResume();
        }

        /* Inicializador das activities */
        public void openNextActivity(View view, String tipo, ArrayList<String> array) {
                Intent intent = null;
                Bundle params = new Bundle();

                params.putString("field_rgm", usuario_rgm);
                params.putString("field_senha", usuario_senha);
                params.putBoolean("conexaoInternet", conexaoInternet);
                if(array==null){
                        params.putBoolean("conexaoServidor", false);
                }
                else{
                        params.putBoolean("conexaoServidor", true);
                }
                if(tipo.equals("Materias")){
                        intent = new Intent(this, MateriaActivity.class);
                        if(conexaoInternet==true && array!=null){
                                params.putStringArrayList("arrayIDMat", array);
                                params.putStringArrayList("arrayNomeMateria", arrayNomeMateria);
                        }
                }
                else if(tipo.equals("Informacoes")){
                                intent = new Intent(this, InfoActivity.class);
                                if(conexaoInternet==true && array!=null){
                                        params.putStringArrayList("arrayInfo", array);

                                }
                        }
                intent.putExtras(params);
            startActivity(intent);
```

```
192         }
193
194 }
```

E.6 Classe MateriaActivity

```
1   package com.example.clientsocketmain;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   import android.net.ConnectivityManager;
7   import android.net.NetworkInfo;
8   import android.os.Bundle;
9   import android.app.Activity;
10  import android.content.Context;
11  import android.content.Intent;
12  import android.database.Cursor;
13  import android.database.sqlite.SQLiteDatabase;
14  import android.util.Log;
15  import android.view.Menu;
16  import android.view.View;
17  import android.widget.AdapterView;
18  import android.widget.AdapterView.OnItemClickListener;
19  import android.widget.ArrayAdapter;
20  import android.widget.ListView;
21  import android.widget.Toast;
22
23  public class MateriaActivity extends Activity {
24
25          DatabaseConnection db;
26          String usuario_rgm, usuario_senha;
27          ListView listViewMateria;
28          List<String> listaNomeMateria, listaID_Materia;
29          boolean conexaoServidor, conexaoInternet;
30          ArrayList<String> arrayNomeMat, arrayIDMat;
31          ArrayList<String> arrayBim, arrayNotas, arrayFaltas;
32          clientSocket cliente;
33
34
35          @Override
36          protected void onCreate(Bundle savedInstanceState) {
37                  super.onCreate(savedInstanceState);
38                  setContentView(R.layout.activity_materia);
39
40                  arrayIDMat = new ArrayList<String>();
41                  arrayNomeMat = new ArrayList<String>();
42
43                  Intent intent = getIntent();
44                  Bundle params = intent.getExtras();
45                  if(params!=null){
46                          usuario_rgm=params.getString("field_rgm");
47                          usuario_senha=params.getString("field_senha");
48                          conexaoServidor=params.getBoolean("conexaoServidor");
49                          if(conexaoServidor==true){
50                                  arrayIDMat = params.getStringArrayList("arrayIDMat");
51                                  arrayNomeMat = params.getStringArrayList("arrayNomeMateria");
52                          }
53                          else{
54                                  arrayIDMat = null;
```

```
                                arrayNomeMat = null;
                        }
                }
                listViewMateria = (ListView)findViewById(R.id.listViewMateria);
                db = new DatabaseConnection(new DatabaseCreateUpdate(this));
                db.openConnection();
                if(conexaoServidor==true && arrayNomeMat!=null){
                        listaNomeMateria = arrayNomeMat;
                        listaID_Materia = new ArrayList<String>();
                        listaID_Materia = arrayIDMat;
                }
                else{
                        listaNomeMateria = buscaMaterias(db.getConnection());
                }

                ArrayAdapter<String>     lista_materia     =     new     ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, listaNomeMateria);
                listViewMateria.setAdapter(lista_materia);



        }

        public  void  openNextActivity(View  view,  String  idMateria,  String  nomeMateria,  boolean
conexao) {
                Intent intent = new Intent(getBaseContext(), NotasActivity.class);
                Bundle params = new Bundle();

                params.putString("field_rgm", usuario_rgm);
                params.putString("materia", nomeMateria);
                params.putString("materiaID", idMateria);
                params.putBoolean("conexao", conexao);
                params.putStringArrayList("arrayBim", arrayBim);
                params.putStringArrayList("arrayNotas", arrayNotas);
                params.putStringArrayList("arrayFaltas", arrayFaltas);
                Log.e("MsgRecevi", arrayBim.size()+"");
                Log.e("MsgReceviN", arrayNotas.size()+"");
                Log.e("MsgReceviFa", arrayFaltas.size()+"");

                intent.putExtras(params);
                startActivity(intent);
        }

        /**
         * Busca todas as materias do aluno
         * @param connection
         * @return lista com as materias do aluno
         */
        public List<String> buscaMaterias(SQLiteDatabase connection){

                List<String> lista = new ArrayList<String>();
                listaID_Materia = new ArrayList<String>();
                Cursor cursor = connection.rawQuery("select distinct notas.ID_MATERIA," +
                "  materia.NOME_MATERIA  from  notas,  materia"  +  "  where  notas.id_aluno=?  and
notas.ID_MATERIA=materia.ID_MATERIA;", new String[]{usuario_rgm});

                while(cursor.moveToNext()){

        listaID_Materia.add(cursor.getString(cursor.getColumnIndex("ID_MATERIA")));
                        lista.add(cursor.getString(cursor.getColumnIndex("NOME_MATERIA")));
                }
                cursor.close();

                return lista;
        }

        public boolean isOnline() {
```

```
            ConnectivityManager             cm              =              (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo netInfo = cm.getActiveNetworkInfo();

            if (netInfo != null && netInfo.isConnectedOrConnecting()) {
                return true;
            }
            return false;
        }

        @Override
        protected void onStart() {
                // TODO Auto-generated method stub
                listViewMateria.setOnItemClickListener(new OnItemClickListener() {
                        int i=0, count=0;
                        @Override
                        public void onItemClick(AdapterView<?> arg0, View view, int pos,
                                        long id) {
                                // TODO Auto-generated method stub

                                db.setNomeMateria(listaNomeMateria.get(pos));
                                db.setIdMateria(listaID_Materia.get(pos));
                                arrayBim = new ArrayList<String>();
                                arrayFaltas = new ArrayList<String>();
                                arrayNotas = new ArrayList<String>();

                                /*
                                arrayBim.clear();
                                arrayFaltas.clear();
                                arrayNotas.clear();
*/

                                if(conexaoInternet==true){
                                        cliente    =    new    clientSocket("@solicitacao_notas",
usuario_rgm, usuario_senha, db, new clientSocket.OnMessageReceived(){

                                                @Override
                                                public void messageReceived(Object message) {
                                                        // TODO Auto-generated method stub
                                                        if(count==0){
                                                                arrayBim   =   (ArrayList<String>)
message;
                                                        }
                                                        else{
                                                                if(count==1){
                                                                        arrayNotas               =
(ArrayList<String>) message;
                                                                }
                                                                else{
                                                                        arrayFaltas              =
(ArrayList<String>) message;
                                                                        i=1;
                                                                }
                                                        }
                                                        count++;
                                                }
                                        });
                                        cliente.start();
                                        while(i==0 && cliente.getFalha_conexao()==false){

                                        }
                                        if(i!=0){
                                                //Ate aqui tudo ok
                                                openNextActivity(view,          db.getIdMateria(),
db.getNomeMateria(), true);
                                        }
                                        else{
```

```
                                                Toast.makeText(getBaseContext(),        "Acessando
localmente", Toast.LENGTH_SHORT).show();
                                                openNextActivity(view,          db.getIdMateria(),
db.getNomeMateria(), false);
                                        }
                                }
                                else{
                                        openNextActivity(view,                  db.getIdMateria(),
db.getNomeMateria(), false);
                                }

                                //Toast.makeText(getBaseContext(),  materiaID  +  "  "  +  materia,
Toast.LENGTH_SHORT).show();

                        }

                });

                super.onStart();
        }

        @Override
        protected void onResume() {
                // TODO Auto-generated method stub
                conexaoInternet = isOnline();

                super.onResume();
        }
```

E.7 Classe NotasActivity

```
package com.example.clientsocketmain;

import java.util.ArrayList;
import java.util.List;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class NotasActivity extends Activity {

        DatabaseConnection db;
        String materia, materiaID;
        String usuario_rgm;
        List<String> listaNota, listaFalta, listaBimestre;
        Button buttonNotasVoltar;
        boolean conexao;
        ArrayList<String> arrayBim, arrayNotas, arrayFaltas;

        @Override
```

```
29          protected void onCreate(Bundle savedInstanceState) {
30                  super.onCreate(savedInstanceState);
31                  setContentView(R.layout.activity_notas);
32                  Intent intent = getIntent();
33                  Bundle params = intent.getExtras();
34
35                  if(params!=null){
36                          usuario_rgm=params.getString("field_rgm");
37                          materia=params.getString("materia");
38
39                          materiaID=params.getString("materiaID");
40
41                          conexao = params.getBoolean("conexao");
42
43                          if(conexao==true){
44                                  arrayBim=params.getStringArrayList("arrayBim");
45                                  arrayNotas=params.getStringArrayList("arrayNotas");
46                                  arrayFaltas=params.getStringArrayList("arrayFaltas");
47                          }
48                          else{
49                                  arrayBim=null;
50                                  arrayNotas=null;
51                                  arrayFaltas=null;
52                          }
53                  }
54
55                  db = new DatabaseConnection(new DatabaseCreateUpdate(this));
56                  db.openConnection();
57                  if(conexao==false){
58                          buscaNotas(db.getConnection(), materiaID, false);
59                  }
60                  else{
61                          buscaNotas(db.getConnection(), materiaID, true);
62                  }
63                  mostraResultados();
64                  db.closeConnection();
65
66                  buttonNotasVoltar = (Button)findViewById(R.id.buttonNotasVoltar);
67
68                  buttonNotasVoltar.setOnClickListener(new OnClickListener() {
69
70                          @Override
71                          public void onClick(View v) {
72
73                                  onBackPressed();
74                          }
75                  });
76          }
77
78          public void mostraResultados(){
79                  TextView tViewNomeMat = (TextView)findViewById(R.id.textViewNomeMateria);
80                  TextView bim1, bim2, bim3, bim4;
81                  TextView n1, n2, n3, n4;
82                  TextView tViewFaltas;
83
84                  tViewNomeMat.setText(materia);
85                  int faltas;
86                  faltas=0;
87
88                  while(listaBimestre.isEmpty()==false){
89                          faltas+=Integer.parseInt(listaFalta.get(0));
90                          if(listaBimestre.get(0).equals("1")){
91                                  bim1 = (TextView)findViewById(R.id.textViewBim1);
92                                  bim1.setVisibility(View.VISIBLE);
93
94                                  n1 = (TextView)findViewById(R.id.textViewN1);
95                                  n1.append(listaNota.get(0));
```

```
                                n1.setVisibility(View.VISIBLE);

                        }else{
                                if(listaBimestre.get(0).equals("2")){
                                        bim2 = (TextView)findViewById(R.id.textViewBim2);
                                        bim2.setVisibility(View.VISIBLE);

                                        n2 = (TextView)findViewById(R.id.textViewN2);
                                        n2.append(listaNota.get(0));
                                        n2.setVisibility(View.VISIBLE);

                                }else{
                                        if(listaBimestre.get(0).equals("3")){
                                                bim3 = (TextView)findViewById(R.id.textViewBim3);
                                                bim3.setVisibility(View.VISIBLE);

                                                n3 = (TextView)findViewById(R.id.textViewN3);
                                                n3.append(listaNota.get(0));
                                                n3.setVisibility(View.VISIBLE);

                                        }
                                        else{
                                                bim4 = (TextView)findViewById(R.id.textViewBim4);
                                                bim4.setVisibility(View.VISIBLE);

                                                n4 = (TextView)findViewById(R.id.textViewN4);
                                                n4.append(listaNota.get(0));
                                                n4.setVisibility(View.VISIBLE);

                                        }
                                }
                        }
                        listaBimestre.remove(0);
                        listaNota.remove(0);
                        listaFalta.remove(0);
                }
                tViewFaltas = (TextView)findViewById(R.id.textViewFaltas);
                tViewFaltas.append(Integer.toString(faltas));
        }

        /**
         * Retorna as notas de uma materia
         * @param connection
         * @param materia
         * @return todas as notas da @param materia
         */
        public void buscaNotas(SQLiteDatabase connection, String materiaPos, boolean val){
                if(val==false){
                        listaNota = new ArrayList<String>();
                        listaFalta = new ArrayList<String>();
                        listaBimestre = new ArrayList<String>();
                        Cursor cursor = connection.rawQuery("select nota, faltas, bimestre" +
                                        " from notas where notas.id_aluno=? and notas.ID_MATERIA=?
limit 4", new String[]{usuario_rgm, materiaPos});

                        while(cursor.moveToNext()){
                                listaNota.add(cursor.getString(cursor.getColumnIndex("NOTA")));
                                listaFalta.add(cursor.getString(cursor.getColumnIndex("FALTAS")));

        listaBimestre.add(cursor.getString(cursor.getColumnIndex("BIMESTRE")));
                        }

                        cursor.close();
                }
                else{
                        listaNota = arrayNotas;
                        listaFalta = arrayFaltas;
```

```
                    listaBimestre = arrayBim;
            }
    }
```

Apêndice F

Instalação do MySQL no Windows 7

1. Entre no site oficial do MySQL[1] e baixe a versão do MySQL para Windows.
2. Após a conclusão do download, execute o arquivo baixado
3. Ao abrir a janela do MySQL Installer, clique em "*Install MySQL Products"*.
4. Leia e aceite os termos de uso, clique em "*Next*".
5. Não marque a opção de pular a checagem de update. Clique em executar para verificar a conexão com a Internet.
6. Após a verificação, clique em "*Next*". Escolha a opção Custom e clique em "*Next*" novamente.
7. Após o passo 6, será aberta uma janela igual à da **Figura F.1**, selecione todos os itens de MySQL Server 5.6.14, e selecione apenas os itens MySQL Workbench e connector/J, em seguida, clique mais duas vezes em "*Next*".

[1]http://www.mysql.com/downloads/

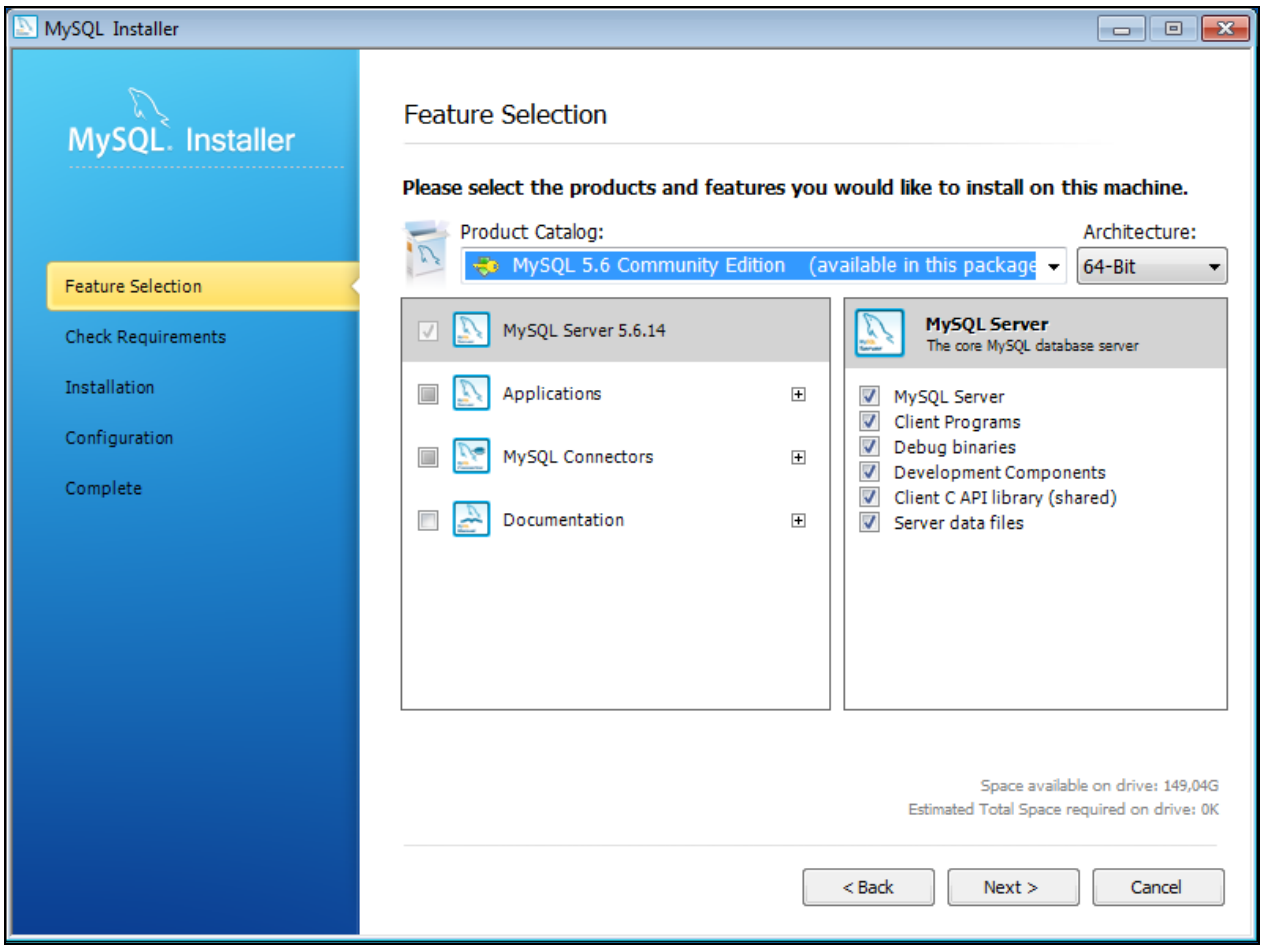


Figura F.1: Itens a instalar.

8. Em seguida, clique em execute para instalar os itens selecionados, conforme mostra a Figura F.2.

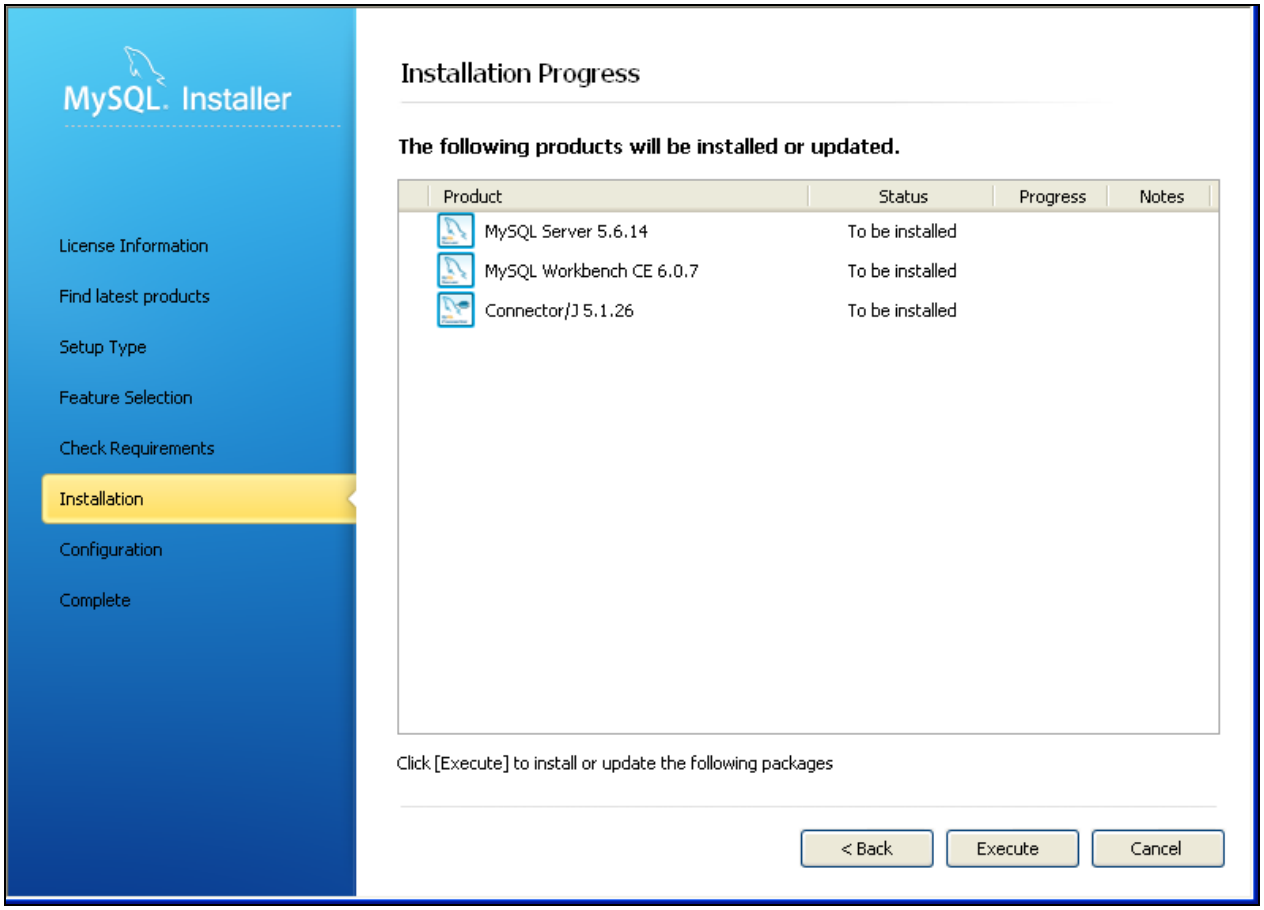


Figura F.2: Itens a serem instalados

9. Após a conclusão da instalação, clique em “Next”.

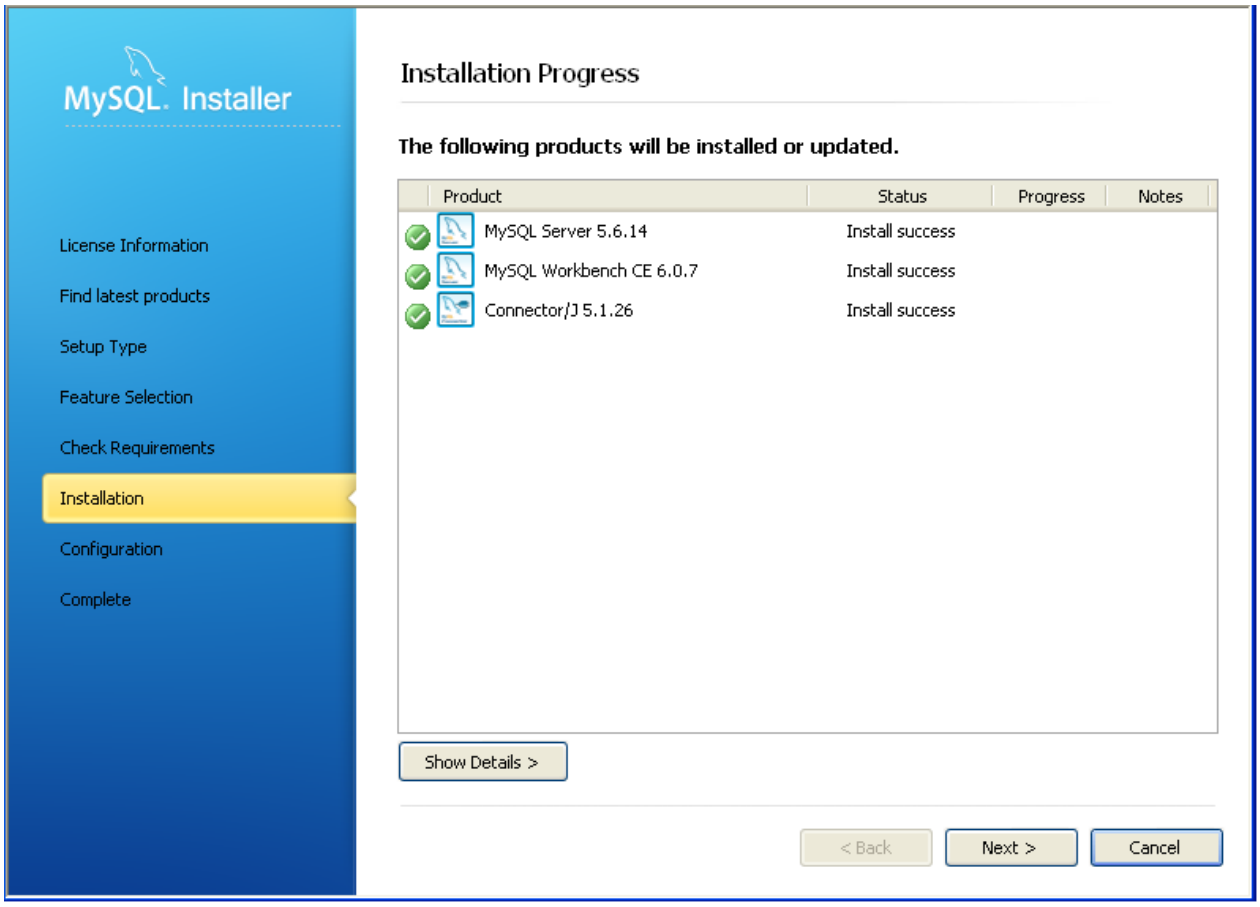


Figura F.3: Término da instalação dos itens selecionados.

10. Em seguida será perguntado se deseja fazer a configuração do MySQL Server, clique em "Next". Escolha o tipo de configuração do servidor, a porta do MySQL, certifique-se de que a opção de abrir porta do firewall está marcada e selecione a opção de mostrar configurações avançadas.

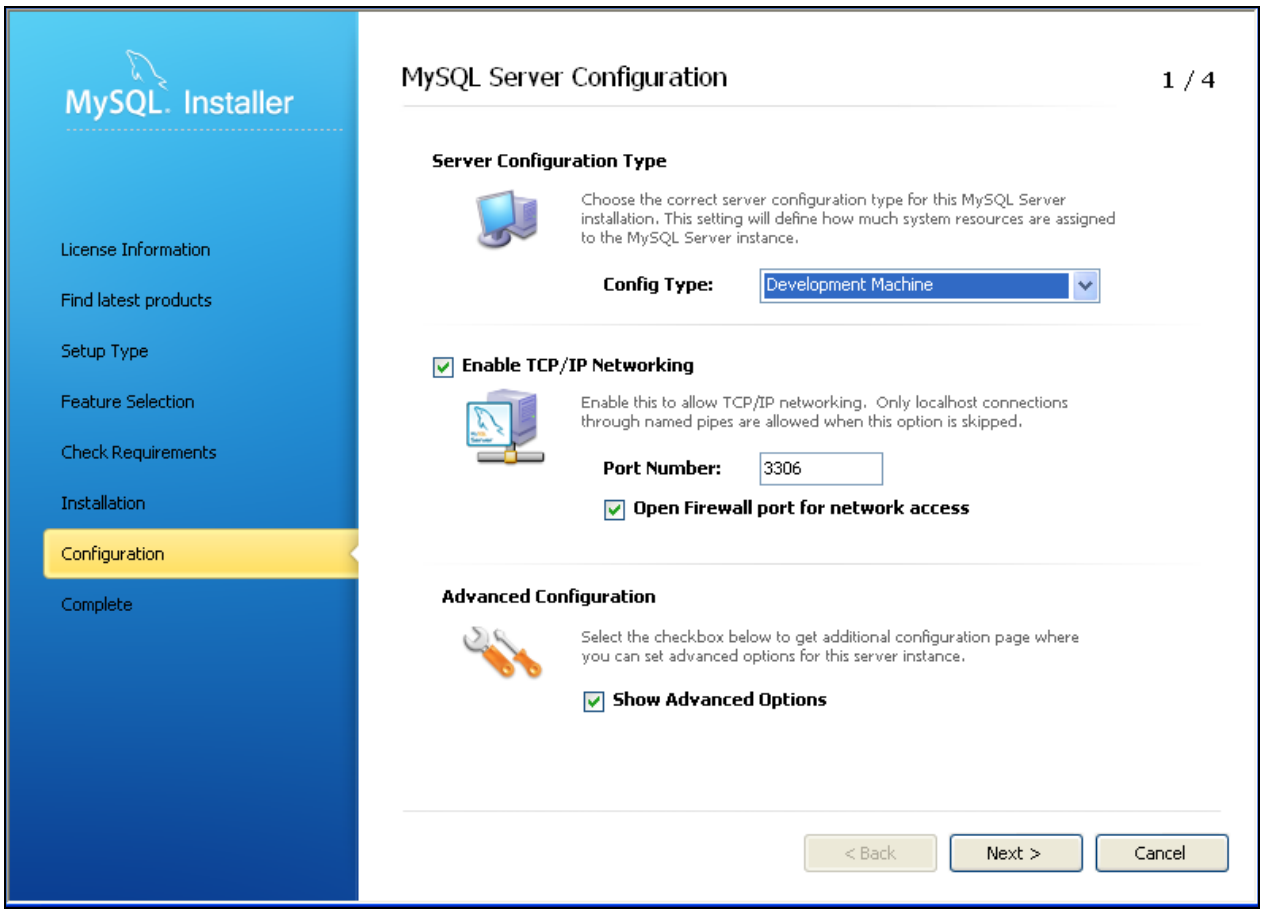


Figura F.4: Opções de configuração do MySQL Server.

11. Em seguida, escolha uma senha root e repita ela no campo seguinte, e clique em "Next". Escolha o nome do serviço, o padrão é MySQL56. Marque a opção de abrir iniciar o MySQL Server quando o Windows iniciar.
12. Em seguida, clique em "Next" duas vezes e aguarde o fim da configuração. Clique em "*Next*" novamente e por fim, em "*Finish*", para terminar.

Para adicionar o connector/J ao *library patch* do seu projeto, faça os seguintes passos:

1. Baixe o connector/J selecionando a plataforma como plataforma independente no site oficial[1];
2. Descompacte o arquivo;
3. Abra seu projeto com Netbeans ou Eclipse e entre nas propriedades do projeto, procure as opções "*Bibliotecas*" ou "*Java Build Patch*";

[1]http://dev.mysql.com/downloads/connector/j/

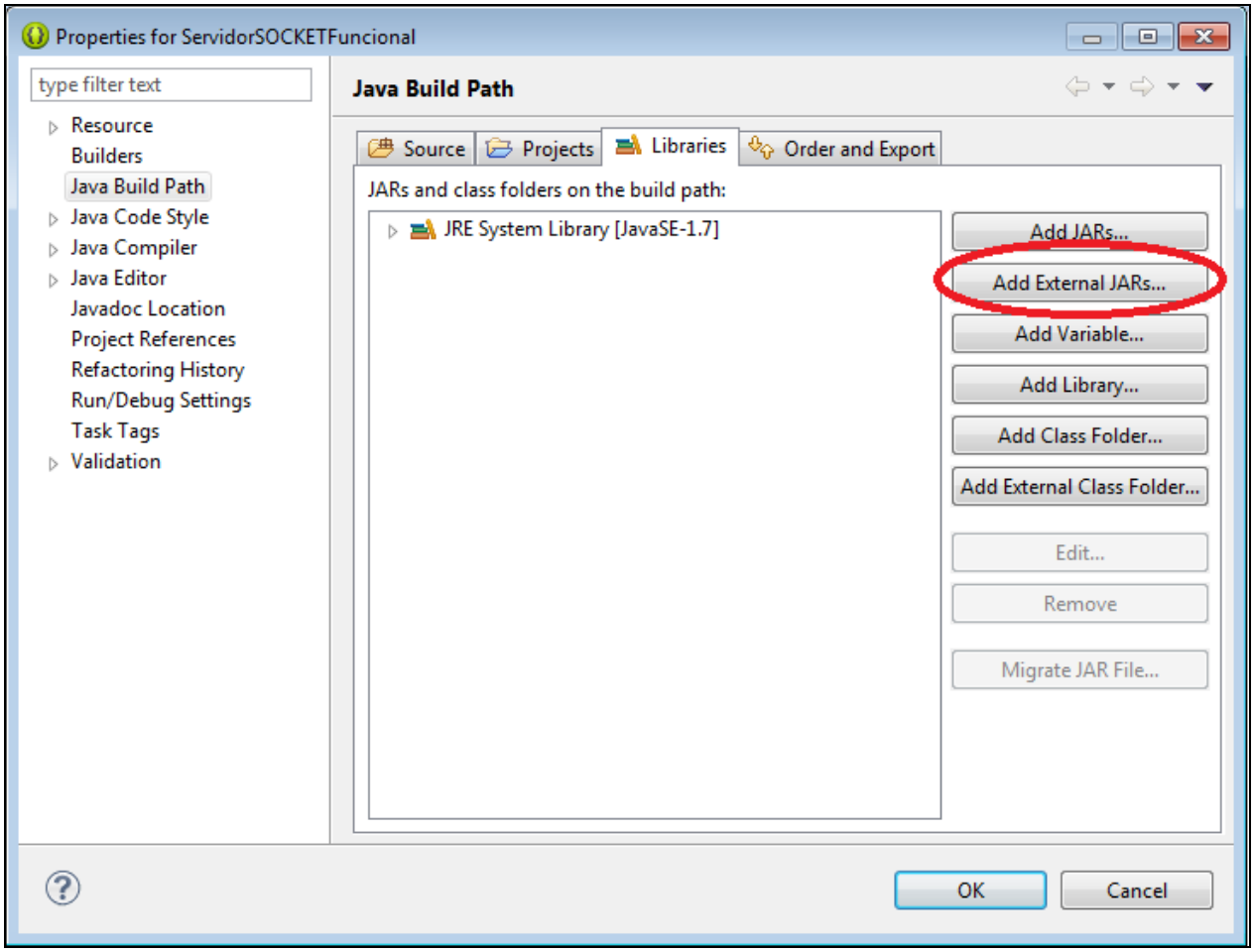


Figura F.5: Propriedades do projeto.

4. Selecione a aba *Libraries*, clique em *Add External Jar*, e selecione o arquivo mysql-connector-java-5.1.XX-bin.jar, onde XX é a versão do connector/J;
5. Clique em adicionar e clique em "OK" para terminar.

Apêndice G

Códigos utilizados na criação do banco de dados

Após iniciar o MySQL, entre com o nome de usuário e senha.

G.1 Crie um banco de dados chamado universidade. Use o banco de dados criado.

```
1   CREATE DATABASE  IF NOT EXISTS `universidade`;
2   USE `universidade`;
```

G.2 Crie uma tabela chamada aluno.

```
1   CREATE TABLE `aluno` (
2     `RGM` int(11) NOT NULL AUTO_INCREMENT,
3     `NOME` varchar(45) DEFAULT NULL,
4     `ENDERECO` varchar(45) DEFAULT NULL,
5     `SENHA` varchar(45) NOT NULL,
6     `EMAIL` varchar(45) DEFAULT NULL,
7     PRIMARY KEY (`RGM`)
8   );
```

G.3 Crie uma tabela chamada professor.

```
CREATE TABLE `professor` (
  `ID_PROFESSOR` int(11) NOT NULL AUTO_INCREMENT,
  `NOME` varchar(45) DEFAULT NULL,
  `SENHA` varchar(45) NOT NULL,
  PRIMARY KEY (`ID_PROFESSOR`)
);
```

G.4 Crie uma tabela chamada materia.

```
CREATE TABLE `materia` (
  `ID_MATERIA` int(11) NOT NULL AUTO_INCREMENT,
  `NOME_MATERIA` varchar(45) DEFAULT NULL,
  `ID_PROFESSOR` int(11) NOT NULL,
  PRIMARY KEY (`ID_MATERIA`,`ID_PROFESSOR`),
  KEY `ID_PROFESSOR_idx` (`ID_PROFESSOR`),
  CONSTRAINT `ID_PROFESSOR` FOREIGN KEY (`ID_PROFESSOR`) REFERENCES `professor`
(`ID_PROFESSOR`) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

G.5 Crie uma tabela chamada notas.

```
CREATE TABLE `notas` (
  `BIMESTRE` int(11) NOT NULL,
  `ID_MATERIA` int(11) NOT NULL,
  `ID_ALUNO` int(11) NOT NULL,
  `NOTA` float DEFAULT NULL,
  `FALTAS` int(11) DEFAULT NULL,
  PRIMARY KEY (`BIMESTRE`,`ID_MATERIA`,`ID_ALUNO`),
  KEY `ID_MATERIA_idx` (`ID_MATERIA`),
  KEY `ID_ALUNO_idx` (`ID_ALUNO`),
  CONSTRAINT `ID_ALUNO` FOREIGN KEY (`ID_ALUNO`) REFERENCES `aluno` (`RGM`) ON
DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `ID_MATERIA` FOREIGN KEY (`ID_MATERIA`) REFERENCES `materia`
(`ID_MATERIA`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Referências Bibliográficas

AND-DEV-FUN. Android Developer Application Fundamentals. http://developer.android.com/guide/components/fundamentals.html, 2013. Acessado em: 24 jul. 2013.

AND-DEV-MEDIA. Android Developer Media Playback. http://developer.android.com/guide/topics/media/mediaplayer.html, 2013. Acessado em: 20 jul. 2013.

AND-DEV-OPENGL. Android Developer OpenGL ES. http://developer.android.com/guide/topics/graphics/opengl.html, 2013. Acessado em: 20 jul. 2013.

AND-DEV-SEC. Android Developer Security with HTTPS and SSL. http://developer.android.com/training/articles/security-ssl.html, 2013. Acessado em: 20 jul. 2013.

AND-DEV-TOOLS. Android Developer Tools Help. http://developer.android.com/tools/help/index.html, 2013. Acessado em: 30 jul. 2013.

AND-DEV-VIS. Android Developer Visibility for Your Apps. http://developer.android.com/distribute/googleplay/about/visibility.html, 2013. Acessado em: 10 ago. 2013.

AND-GIT-REP. Android Git Repositories. https://android.googlesource.com, 2013. Acessado em: 20 jul. 2013.

AND-SRC-CODE. The Android Source Code. http://source.android.com/source/index.html, 2013. Acessado em: 27 ago. 2013.

AND-SRC-BUILD. Android Source Codenames, Tags and Build Numbers. http://source.android.com/source/build-numbers.html, 2013. Acessado em: 29 ago. 2013.

AND-SRC-GRA. Android Source Graphics. http://source.android.com/devices/graphics.html, 2013. Acessado em: 20 jul. 2013.

AQUINO, J. F. S. Plataformas de Desenvolvimento Para Dispositivos Móveis. 2007. 14 f. Monografia (Pós Graduação em Informática). Programa de Pós Graduação em Informática, Pontifícia Universidade Católica do Rio de Janeiro – Rio de Janeiro.

BARROS, T. Cinco anos de Android: relembre a história e todas as versões do sistema. http://www.techtudo.com.br/noticias/noticia/2013/09/cinco-anos-de-android-relembre-historia-e-todas-versoes-do-sistema.html, 2013. Acessado em: 29 sep. 2013.

BEAVIS, G. A complete history of Android. http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327, 2008. Acessado em: 15 jul. 2013.

BERNSTEIN, P. A; HADZILACOS, V; GOODMAN, N (1987). Concurrency control and recovery in database systems. 226p-240p. Addison Wesley Publishing Company. ISBN 0-201-10715-5.

BORDIN, M. V. Introdução a Arquitetura Android. http://sites.setrem.com.br/stin/2012/anais/Maycon.pdf, 2012. Acessado em: 19 jul. 2013.

BURETTA, M. Data Replication: tools and techniques for managing distributed information. Estados Unidos da América: John Wiley e Sons, Inc, 1997. 157p.

CRUZ, M. Computação Móvel – Evolução e Estado Atual. Disponível em < http://www.practicalsw-pt.blogspot.com.br/2012/08/computacao-movel-evolucao-e-estado-atual.html>. Acessado em: 25 set. 2013

EHRINGER, D. The Dalvik Virtual Machine Architecture. Disponível em: <http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf>. Acessado em: 20 nov. 2013.

FERREIRA, S. D. A. et al. Bluetooth – aedb. In: II Simpósio de Excelência em Gestão e Tecnologia. p.01-08, 2005. Disponivel em <http://www.aedb.br/seget/artigos05/16_artigo_Bluetooth.pdf>. Acessado em: 14 ago. 2013.

FIGUEIREDO, M. S.; NAKAMURA, E. Computação Móvel: Novas Oportunidades e Novos Desafios. T&C Amazônia, Junho de 2003.

FREETYPE. http://www.freetype.org/, 2013. Acesso em: 20 jul. 2013.

GARCIA, L. G. U. Redes Locais Sem Fio Que Atendem ao Padrão IEEE 802.11. Disponível em < http://www.gta.ufrj.br/grad/01_2/802-mac/> Acessado em 26 set. 2013.

GARGENTA, M. Learning Android. O'Reilly Media, 2011.

GOOGLEBLOG. Introducing Google Play: All your entertainment, anywhere you go. http://googleblog.blogspot.com.br/2012/03/introducing-google-play-all-your.html, 2012. Acessado em: 26 jul. 2013.

GOOGLEPLAY. Google Play Apps. http://play.google.com/about/apps/index.html, 2013. Acessado em: 29 jul. 2013.

HTC-G1. HTC-About. http://www.htc.com/www/about/?id=66338&lang=1033, 2012. Acessado em: 23 jul. 2012.

KELION, L. Android KitKat unveiled in Google surprise move. http://www.bbc.co.uk/news/technology-23926938, 2013. Acessado em: 29 sep. 2013.

KOVACH, S. How Android Grew To Be More Popular Than The iPhone. http://www.businessinsider.com/history-of-android-2013-8?op=1, 2013. Acessado em: 13 ago. 2013.

MILK. Dalvik Libraries. http://milk.com/kodebase/dalvik-docs-mirror/docs/libraries.html, 2013. Acessado em: 25 jul. 2013.

OHA-AND-REV. Android Review. http://www.openhandsetalliance.com/android_overview.html, 2007. Acessado em: 12 jul. 2013.

OHA-ANN-ANDROID. Industry Leaders Announce Open Plataform for Mobile Devices. http://www.openhandsetalliance.com/press_110507.html, 2007. Acessado em: 15 jul. 2013.

ÖZSU M.; VALDURIEZ P. Principles of Distributed Database Systems. Nova Jersey: Prentice-Hall, 1999. 185p.

PEREIRA, L. C. O.; SILVA, M. L. D. Android para desenvolvedores. 2009. Editora Brasport. Disponivel em: <http://books.google.com.br/books?id=8u9wJowXfdUC&lpg=PA1&ots=LSjp04-ln3&dq=android&lr=lang_pt&hl=pt-BR&pg=PP1#v=onepage&q&f=false>. Acessado em: 20 jul. 2013.

PESSOA, A. C. Android: Aplicações do Futuro. Revista Asper, V.02, p.01-03, 2013.

RICKER, T. HTC Dream FCC approved, Android clear for launch?. http://www.engadget.com/2008/08/18/htc-dream-fcc-approved-android-clear-for-launch/, 2008. Acessado em: 12 jul. 2013.

SAITO, Y.; SHAPIRO, M. Optimistic replication. ACM Computing Surveys, 2005.

SILVA, L. A. D. Apostila de Android, 4ª ed. http://www.agenciadream.com/uploads/download/download_bf2f5d7803d78d0de48d133a0aa450dd.pdf, 2012. Acesso em: 11 jul. 2013.

SQLITE. http://www.sqlite.org/. Acessado em: 20 jul. 2013.

STEIN, P. Bluetooth - Introdução. Disponível em <http://www.gta.ufrj.br/seminarios/semin2003_1/stein/> Acessado em 29 set. 2013

STRICKLAND, J. How the Google Phone Works. http://electronics.howstuffworks.com/google-phone.htm, 2007. Acessado em: 20 jul. 2013.

SYMLAB. Surface Manager Overview. http://www.symlab.org/main/documentation/reference/s3/pdk/GUID-55EF3CEB-AF3E-5A32-96F3-F146D1A06C8F.html, 2013. Acessado em: 29 jul. 2013.

T-MOBILE. T-Mobile G1 - The First Phone Powered by Android. http://www.t-mobile.com/company/PressReleases_Article.aspx?assetName=Prs_Prs_20080923&title=T-Mobile%20Unveils%20the%20T-Mobile%20G1%20%E2%80%93%20the%20First%20Phone%20Powered%20by%20Android, 2008. Acesso em: 13 jul. 2013.

WEBKIT. http://www.webkit.org/. Acessado em: 20 jul. 2013.