UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LEANDRO REMPEL MEDEIROS

DESENVOLVIMENTO DE UM JOGO PARA ANDROID UTILIZANDO RECURSOS DE ACELERÔMETRO E TELA SENSÍVEL AO TOQUE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO – PR
2013

LEANDRO REMPEL MEDEIROS

DESENVOLVIMENTO DE UM JOGO PARA ANDROID UTILIZANDO RECURSOS DE ACELERÔMETRO E TELA SENSÍVEL AO TOQUE

Monografia apresentada como requisito parcial para obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná, Campus Pato Branco.

Orientador: Prof. Robison Cris Brito

PATO BRANCO - PR

2013

ATA Nº: **204**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **LEANDRO REMPEL MEDEIROS**.

Às 09:05 hrs do dia 16 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Robison Cris Brito (Orientador), Gilda Aparecida de Assis (Convidada) e Luís Cassiano Goularte Rista (Convidado), para avaliar o Trabalho de Diplomação do aluno Leandro Rempel Medeiros, matrícula 844144, sob o título **Desenvolvimento de um Jogo para Android Utilizando Recursos de Acelerômetro e Tela Sensível a Toque**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 10:15 hrs foi encerrada a sessão.

Prof. Robison Cris Brito, M.Sc.
Orientador

Profa. Gilda Aparecida de Assis, Dr.
Convidada

Prof. Luís Cassiano Goularte Rista, M.Sc.
Convidado

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

Dedico este trabalho a Ivana Conceição Marques e por todo incentivo e ajuda para que isso fosse possível.

AGRADECIMENTOS

Ao professor orientador Robison Cris Brito, pela perseverança, confiança e paciência na conclusão deste trabalho, sempre incentivando e motivando, desde o trabalho de estágio até este momento, sempre disponível para ajudar e orientar em tudo o que foi possível.

"Todos os homens sonham: mas não do mesmo jeito. Aqueles que sonham de noite nos recessos empoeirados de suas mentes acordam no dia seguinte para descobrir que seus sonhos eram vaidades. Mas aqueles que sonham de dia são homens perigosos, pois podem atuar em seus sonhos com os olhos abertos para torná-los realidade."

(T. E. Lawrence)

RESUMO

MEDEIROS, Leandro Rempel. Desenvolvimento de um jogo para Android utilizando recursos de acelerômetro e tela sensível ao toque. 2013. 102 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2013.

Com a evolução tecnológica, hoje *tablets* e *smartphones* possuem um poder de processamento equivalente a muitos computadores, e desta forma, tornam-se uma plataforma atrativa para o desenvolvimento de jogos. O mercado para jogos *mobile* é crescente e favorável para desenvolvedores independentes, seja pelas ferramentas gratuitas para desenvolvimento, seja pelas facilidades para venda e distribuição de aplicativos usando ferramentas *online* como o Google Play. Assim, este trabalho apresenta o desenvolvimento de um jogo para a plataforma Android controlado por uma tela sensível a toque e também pelo acelerômetro do aparelho. Para este desenvolvimento foi realizado um estudo sobre as ferramentas para o desenvolvimento e as tecnologias para controle do jogo (tela sensível a toque e acelerômetro). Para facilitar o desenvolvimento, um *framework* foi proposto e codificado. O resultado foi um jogo didático, onde a complexidade do código foi diminuída consideravelmente pelo uso do *framework*. Ao final do desenvolvimento, o aplicativo foi disponibilizado na plataforma Google Play.

Palavras-Chave: Aplicativo Móvel, Android, Jogo, Tela sensível a toque, Acelerômetro.

ABSTRACT

MEDEIROS, Leandro Rempel. Game Development using Android with Acelerometer and Touch-Screen Resources. 2013. 102 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2013.

Nowadays, with technological developments, tablets and smartphones, we have a power processing equivalent to many computers, and thus become one exciting platform for game development. The market for mobile games is increasing and favorable for independent developers, whether by tools free for development, whether the facilities for sale and distribution applications using online tools like Google Play. Thus, this work presents the development of a game for the Android platform controlled by one touch screen and also the accelerometer of the device. for this developing a study about the tools for development and the technologies to control the game (touchscreen and accelerometer). To facilitate the development, a framework was proposed and encoded. The result was a didactic game, where the complexity of the code was decreased considerably by the use of the framework. At the end of the development, application was made available on the platform Google Play.

Keywords: Mobile Application, Android, Game, Touchscreen, Accelerometer.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

2D	Espaço Bidimensional
3G	Terceira Geração de padrões e tecnologias de telefonia móvel
API	Interface de Programação de Aplicativos (Application Programming Interface)
APK	Arquivo pacote do Android (Android Package)
AVD	Dispositivo Virtual Android (Android Virtual Device)
BD	Banco de Dados
CEO	Presidente ou Diretor Geral (Chief Executive Officer)
CNPJ	Cadastro Nacional de Pessoal Jurídica
DAO	Objeto de Acesso a Dados (Data Access Object)
GPS	Sistema de Posicionamento Global (*Global Positioning System*)
IDE	Ambiente Integrado de Desenvolvimento (Integrated Development Environment)
JAR	Arquivo compactado usado para distribuir um conjunto de classes Java
OO	Orientação a Objeto
SDK	Kit de Desenvolvimento de Software (*Software Development Kit*)
SQL	Linguagem de Consulta Estruturada (Structured Query Language)
URL	Localizador-Padrão de Recursos (Uniform Resource Locator)
USB	Universal Serial Bus
WEB	Rede que conecta computadores por todo mundo
XML	Linguagem de Marcação Extensível (eXtensible Markup Language)
ZIP	Formato de compactação de arquivos

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão apresentadas as considerações iniciais do trabalho, seus objetivos, a justificativa, e finalizando, a estrutura do trabalho.

1.1 CONSIDERAÇÕES INICIAIS

O desenvolvimento de jogos eletrônicos teve início há muitos anos atrás, em 1958, quando William Higinbotham criou um jogo com um osciloscópio e um computador analógico, que segundo HUNTER (2000), foi o primeiro jogo eletrônico do mundo. Alguns anos depois, em 1971, foi criado o “SpaceWar”, o primeiro jogo comercializado.

No início, os jogos eram desenvolvidos para os tradicionais *video-games*, onde as imagens eram exibidas em televisores, o controle era feito através de *joysticks* limitados, e os jogos, simples na época, não prendiam a atenção do usuário por mais do que poucas horas.

Até que, na década de 90, os jogos foram introduzidos no mercado de computadores, dispositivos estes com recursos não tão limitados quanto os tradicionais *video-games*. Assim, foi possível o desenvolvimento de jogos com gráficos mais sofisticados e com uma melhor jogabilidade.

Com a evolução tecnológica, hoje *tablets* e *smartphones* possuem um poder de processamento equivalente a muitos computadores, e desta forma, tornam-se uma plataforma atrativa para o desenvolvimento de jogos. Assim, a tendência é que usuários migrem para os dispositivos portáteis na busca de entretenimento, que na sua maioria é a execução de jogos em plataformas mais avançados, como Blackberry, Android ou iOS.

Dentro das tecnologias disponíveis para o desenvolvimento de aplicativos para *smartphones* recebe destaque o Android, por ser uma tecnologia *open-source*, gratuita, permitindo que o desenvolvedor utilize todos os recursos disponíveis nos aparelhos, algo que não acontece nas outras tecnologias, como Java ME ou Brew. Isto é possível, pois o ambiente de desenvolvimento Android é executado sobre o sistema operacional Android, permitindo uma ampla integração.

Neste contexto, o presente trabalho apresenta o desenvolvimento de um jogo para a plataforma Android. Um jogo relativamente simples onde o controle é feito usando uma tela sensível ao toque (*touch-screen*), assim como com o movimento do celular (usando acelerômetro).

Para poder definir como o jogo pode ser feito será realizado uma pesquisa sobre tecnologias que podem ajudar na formação da jogabilidade e controle do jogo, como tela sensível ao toque e acelerômetro.

Para o desenvolvimento deste jogo será necessário desenvolver um *framework* de classes que ficará responsável por toda a parte de desenho de objetos e controle deles em tela.

Após, são apresentados os procedimentos para disponibilizar este jogo na plataforma de vendas da Google – o Google Play.

1.2 OBJETIVOS

A seguir serão apresentados os objetivos geral e específico do trabalho.

1.2.1 Geral

Desenvolver um jogo para Android controlado por uma tela sensível a toque e pelo acelerômetro. Sua distribuição é realizada pelo Google Play.

1.2.2 Específicos

Dentre os objetivos específicos do trabalho, destacam-se:

- Estudar o uso do Acelerômetro e interação através do toque em tela para Android;
- Realizar o desenvolvimento de um framework de classes para desenvolvimento de jogos;
- Realizar o desenvolvimento do jogo para Android;
- Disponibilizar o jogo no Google Play.

1.3 JUSTIFICATIVA

De acordo com ABI RESEARCH, no ano de 2010, foram feitos cerca de 9 bilhões de *downloads* de aplicativos para dispositivos móveis, e em 2011 foram 29 bilhões, o que mostra um crescimento de 322%. Assim a estimativa para 2014, segundo IDC (2010) é que sejam feitos 79,6 bilhões de *downloads* de aplicativos.

Atualmente, uma das tendências do mercado *mobile* é o desenvolvimento de jogos que de acordo com GEEKAPHONE (2011) a previsão é de que a indústria de jogos móveis movimentem US$ 8 bilhões até o final do ano de 2013 e US$ 11.4 bilhões em 2014.

Apesar deste mercado fértil, com grande crescimento e ótimas estimativas, ainda, no mercado de jogos móveis, não se tem muitas ferramentas para criação de jogos e a literatura sobre o assunto ainda não é rica.

Desta forma, este trabalho apresenta o desenvolvimento de um jogo para plataforma Android, que é uma plataforma que dispõe de ambiente de desenvolvimento gratuito, assim facilitando a entrada de desenvolvedores independentes. Para facilitar o desenvolvimento do jogo, um *framework* de classes será desenvolvido de forma didática para auxiliar o desenvolvimento de trabalhos futuros.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em cinco capítulos, dos quais este é o primeiro e apresenta um contexto sobre o tema, incluindo os objetivos e a justificativa.

O Capítulo 2 contém o referencial teórico que apresenta jogos para *smartphones*, recursos do Android bem como o modelo de negócio utilizado pelo Google – Google Play.

No Capítulo 3 são apresentadas as ferramentas e metodologias de desenvolvimento utilizadas no desenvolvimento do jogo, fruto deste trabalho. Neste, é feito um resumo de cada ferramenta e apresentada a análise de requisitos realizada.

O Capítulo 4 apresenta o jogo que foi desenvolvido, com suas características e parte de seu código fonte.

No Capítulo 5 é apresentada a conclusão do trabalho juntamente com as considerações finais e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO

O presente capítulo apresenta os conceitos fundamentais de jogos eletrônicos para *smartphones*, os recursos utilizados do Android (tela sensível ao toque e acelerômetro) bem como o modelo de negócio utilizado pelo Google Play.

2.1 JOGOS

De acordo com MICHAELIS (2009) uma definição de jogo é o divertimento ou exercício em que pessoas fazem prova da sua habilidade, destreza ou astúcia. Que em resumo, nada mais é que uma competição.

Um jogo pode ter um jogador, dois ou mais, jogando competitivamente e/ou cooperativamente, cada qual com seu objetivo. Existem jogos para diversos públicos alvos: crianças, jovens e adultos. Também existem jogos com intuito educativo, utilizados no aprendizado das crianças e jogos com intuito lógico, para desenvolvimento de adolescentes e adultos, dentre outros.

Neste contexto tem-se os jogos eletrônicos. O primeiro jogo eletrônico foi criado em 1958 por William Higinbotham com um osciloscópio e um computador analógico, que segundo HUNTER (2000), foi o primeiro jogo eletrônico do mundo. Alguns anos depois, em 1971, foi criado o “SpaceWar”, o primeiro jogo comercializado.

No início, os jogos eram desenvolvidos para os tradicionais *video-games*, simples na época, não prendiam a atenção do usuário por mais do que poucas horas.

Até que, na década de 90, os jogos foram introduzidos no mercado de computadores, dispositivos estes com recursos não tão limitados quanto os tradicionais *video-games*. Assim, foi possível o desenvolvimento de jogos com gráficos mais sofisticados e com uma melhor jogabilidade.

Jogabilidade é toda a interação, e a forma como é feita esta interação, do jogador com o jogo.

2.2 JOGOS ELETRÔNICOS PARA SMARTPHONES

Com o desenvolvimento do primeiro jogo comercializado, em 1971, este chamado de “SpaceWar”, nasceu o grande mercado dos jogos eletrônicos que hoje supera até mesmo o mercado do cinema dentro da área de entretenimento, como pode-se verificar no infográfico feito por LANDIM (2011) - Figura 1, o qual compara o mercado de cinema e de jogos com seus investimentos.

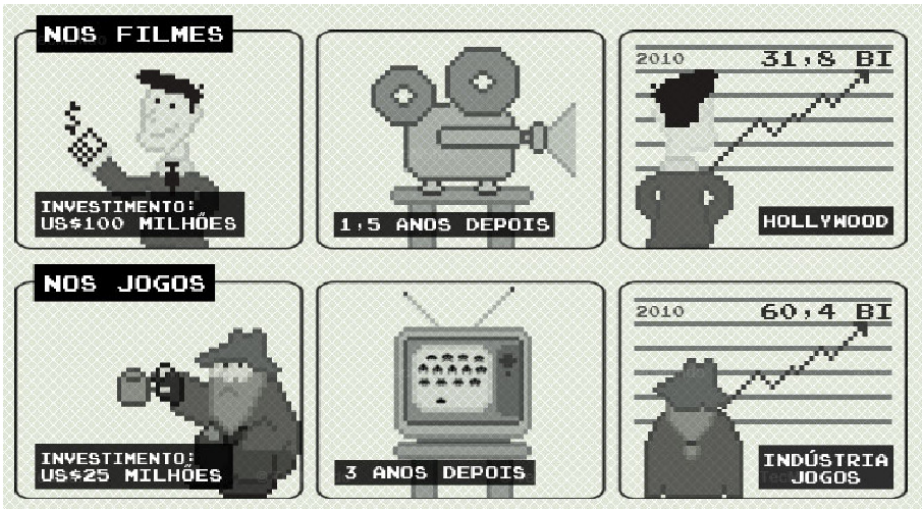

Figura 1 Infográfico O Tamanho da Indústria dos Video Games
Fonte: LANDIM (2011)

Uma das grandes novidades no mundo dos jogos é os dispositivos móveis, como *tablets*, *smarthphones* e celulares, que vem crescendo muito nos últimos anos. O crescimento é de tal forma que, segundo PEREZ (2012), o número de dispositivos móveis ativos no mundo já ultrapassou a população mundial no ano de 2012. Esses dispositivos móveis possuem vários tamanhos e diferentes sistemas operacionais, mas segundo NASCIMENTO (2010), a tendência é que usuários migrem para os *smartphones*, que na sua maioria são aparelhos com os sistemas operacionais Blackberry, Android ou iOS.

Dentro das tecnologias disponíveis para o *smartphones*, uma que recebe destaque é o Android, por ser uma tecnologia *open-source*, gratuita, permitindo que o desenvolvedor utilize todos os recursos disponíveis nos aparelhos, que SILVA (2011) cita como exemplos a câmera do dispositivo, GPS, Google Maps, 3G, tela sensível ao toque, banco de dados nativo, entre outros.

No desenvolvimento de jogos, pode-se utilizar todos estes recursos para enriquecer o jogo e deixa-lo mais atrativo. Um exemplo é o jogo FRUIT NINJA (2013) que utiliza a tela sensível ao toque para simular uma espada cortando as frutas lançadas, conforme Figura 2.

Figura 2 Jogo Fruit Ninja
Fonte: FRUIT NINJA (2013)

Outro exemplo é o NEED FOR SPEED SHIFT (2013) que utiliza o acelerômetro para simular um volante, utilizando para isso o *smartphone*, conforme Figura 3.

Figura 3 Jogo Need for Speed Shift
Fonte: NEED FOR SPEED SHIFT (2013)

Os jogos citados anteriormente são disponibilizados na Google Play, uma plataforma de venda de aplicativos para dispositivos Android, mantida pela Google, esta apresentada com detalhes na seção 2.5.

2.3 ACELERÔMETRO

De acordo com CASASANTA (2012) o acelerômetro é o sensor que permite a identificação da posição física do dispositivo, ou seja, através dele consegue-se saber quando um dispositivo está inclinado para a esquerda ou direita, para frente ou para traz, pra baixo ou para cima conforme ilustrado na Figura 4.

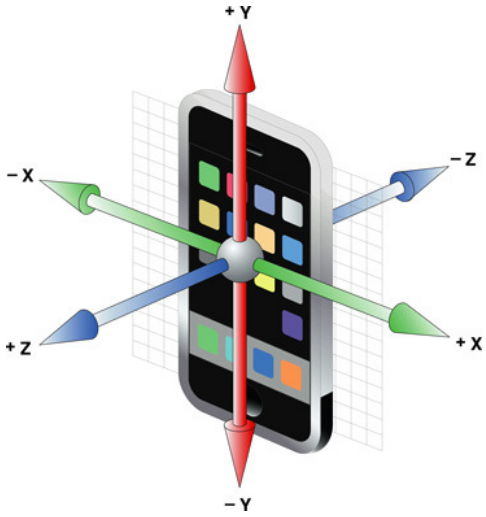

Figura 4 Ilustração do acelerômetro
Fonte: NASCIMENTO (2012)

De acordo com VECTORNAV (2013) um acelerômetro pode ser pensado como um peso suspenso em dois lados opostos com molas. O peso é conhecido como a “massa de prova” e a direção em que a massa é deixada mover-se é conhecido como “eixo de sensibilidade”. Assim pode-se medir a inclinação pela força exercida neste peso ou distância percorrida, como se pode notar na Figura 5.

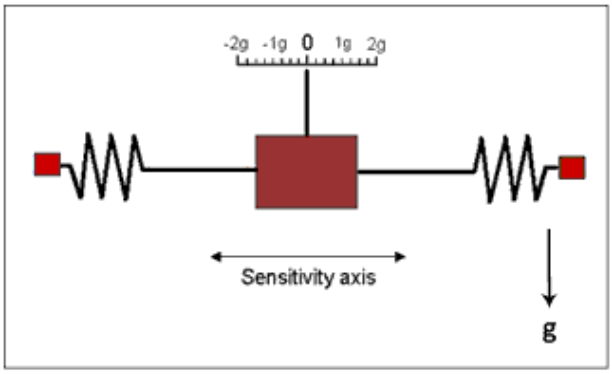

Figura 5 Exemplo de Acelerômetro
Fonte: VECTORNAV (2013)

2.4 TELA SENSÍVEL AO TOQUE

De acordo com HAMMERSCHMIDT (2008) a tela sensível ao toque é um *display* eletrônico visual que pode detectar a presença e localização de um toque dentro da área de exibição, por meio de pressão. O termo refere-se geralmente ao toque no visor do dispositivo com o dedo ou a mão, que também podem reconhecer objetos, como uma caneta. Este formato de captação de toque é chamado de resistivo.

Outro tipo de tela sensível ao toque é o sistema capacitivo, uma camada que armazena a carga elétrica é colocada no painel de vidro do monitor. Quando alguém toca no monitor com seu dedo, parte da carga é transferida para essa pessoa, de modo que a carga na camada capacitiva diminui. Esta diminuição é medida nos circuitos localizados em cada canto do monitor.

Pode-se verificar as duas formas de tela sensível ao toque na Figura 6.

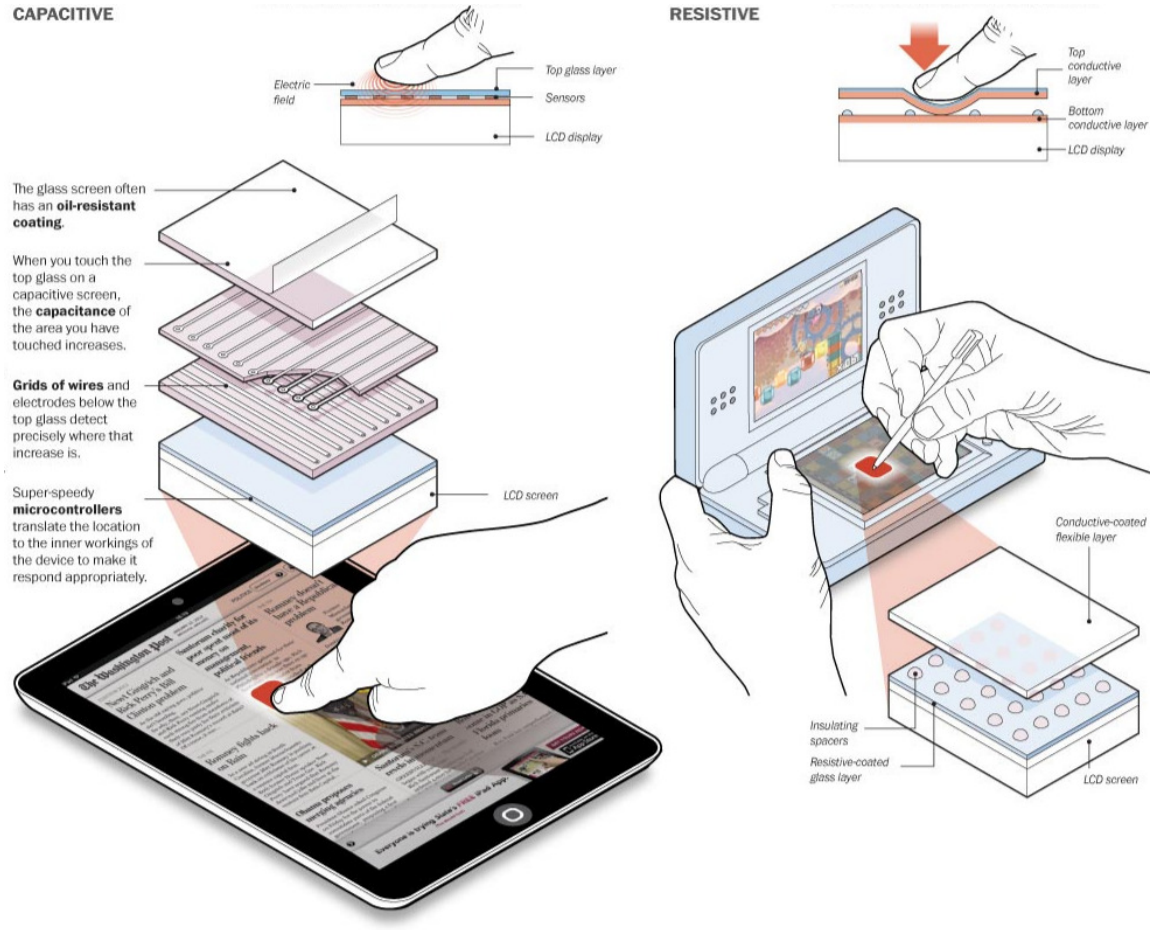

Figura 6 Tela Capacitiva e Resistiva
Fonte: WASHINGTONPOST (2012)

2.5 GOOGLE PLAY

O Google Play é uma plataforma para distribuição de conteúdo para dispositivos Android, tais como livros, filmes, aplicativos e jogos. A página principal do Google Play é apresentada na Figura 7.

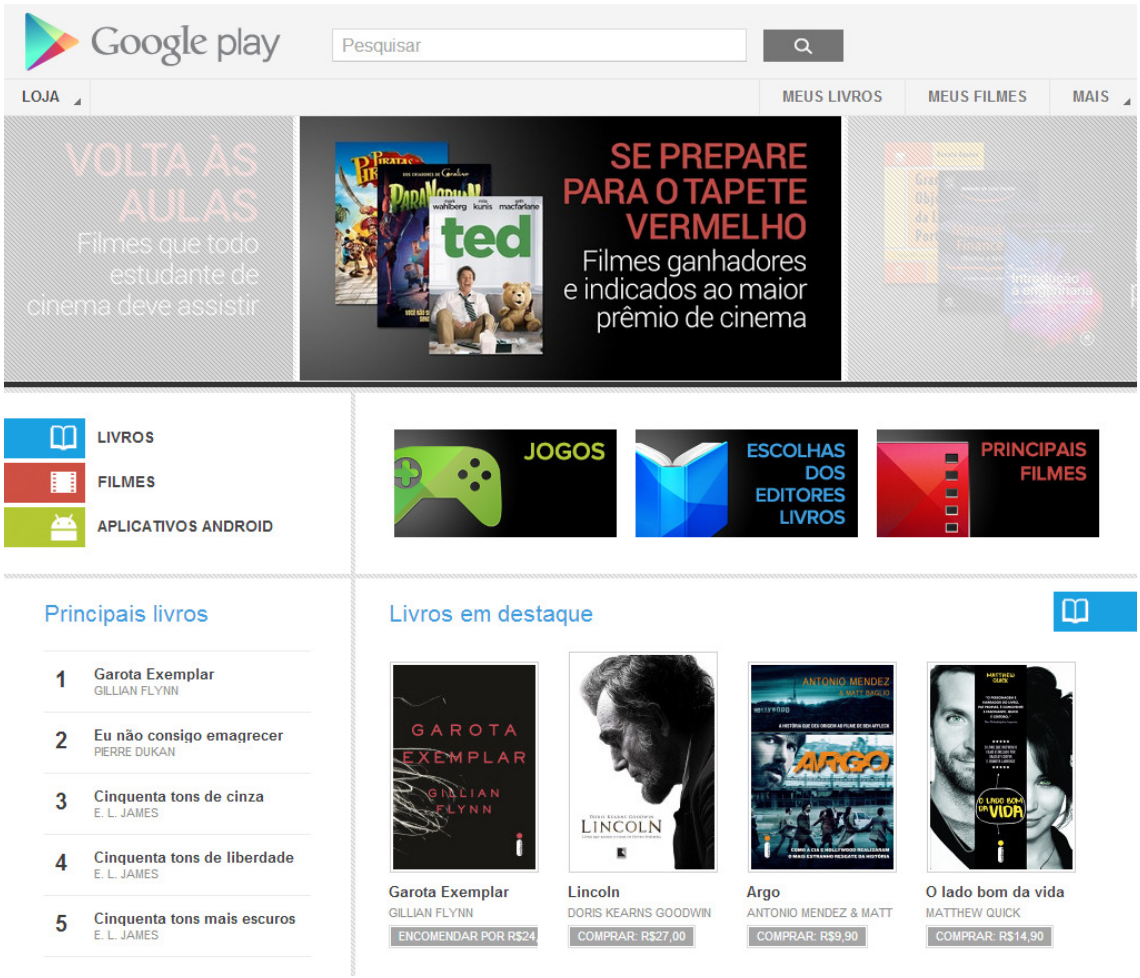

Figura 7 Página inicial da Plataforma Google Play
Fonte: GOOGLE PLAY (2013a)

Na página inicial, encontram-se uma lista com os principais conteúdos, bem como suas categorias, o que facilita a localização do aplicativo ou recurso desejado. Também existe um sistema de busca na parte superior do portal, o que facilita a localização de um aplicativo.

Para exemplificar a facilidade de venda oferecida pelo portal, o jogo NEED FOR SPEED MOST WANTED (2013), um game pago que de acordo com o GOOGLE PLAY (2013a), nos últimos dias foram vendidos mais de 100.000

unidades, pode ser adquirido por menos de R$ 10,00 a partir do portal, conforme apresentado na Figura 8.

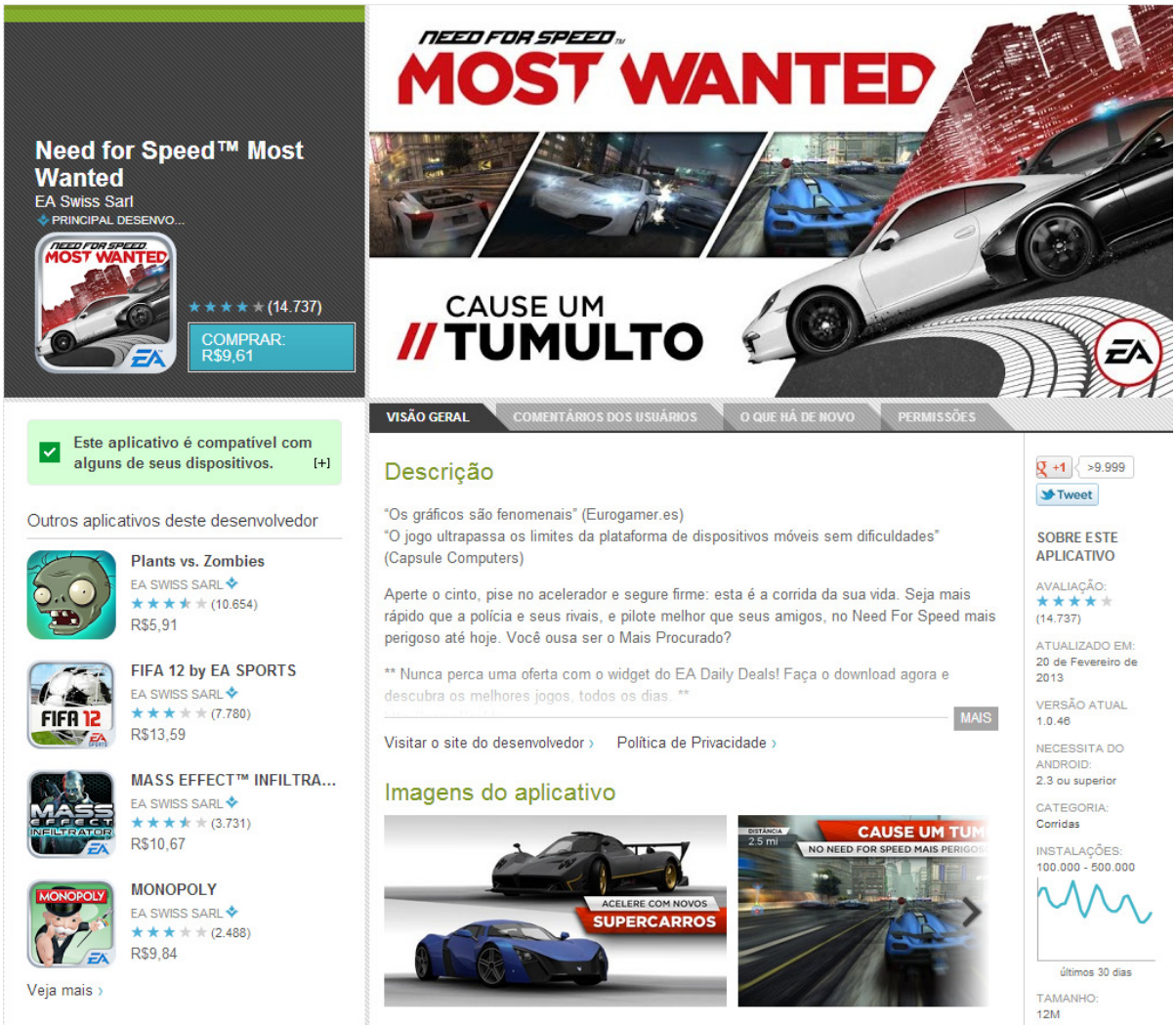

Figura 8 Jogo Need for Speed Most Wanted
Fonte: NEED FOR SPEED MOST WANTED (2013)

O portal também disponibiliza opções de jogos gratuitos, como FRUIT NINJA FREE (2013) que, de acordo com o GOOGLE PLAY (2013a), teve mais de 50.000.000 instalações nos últimos 30 dias conforme mostra Figura 9.

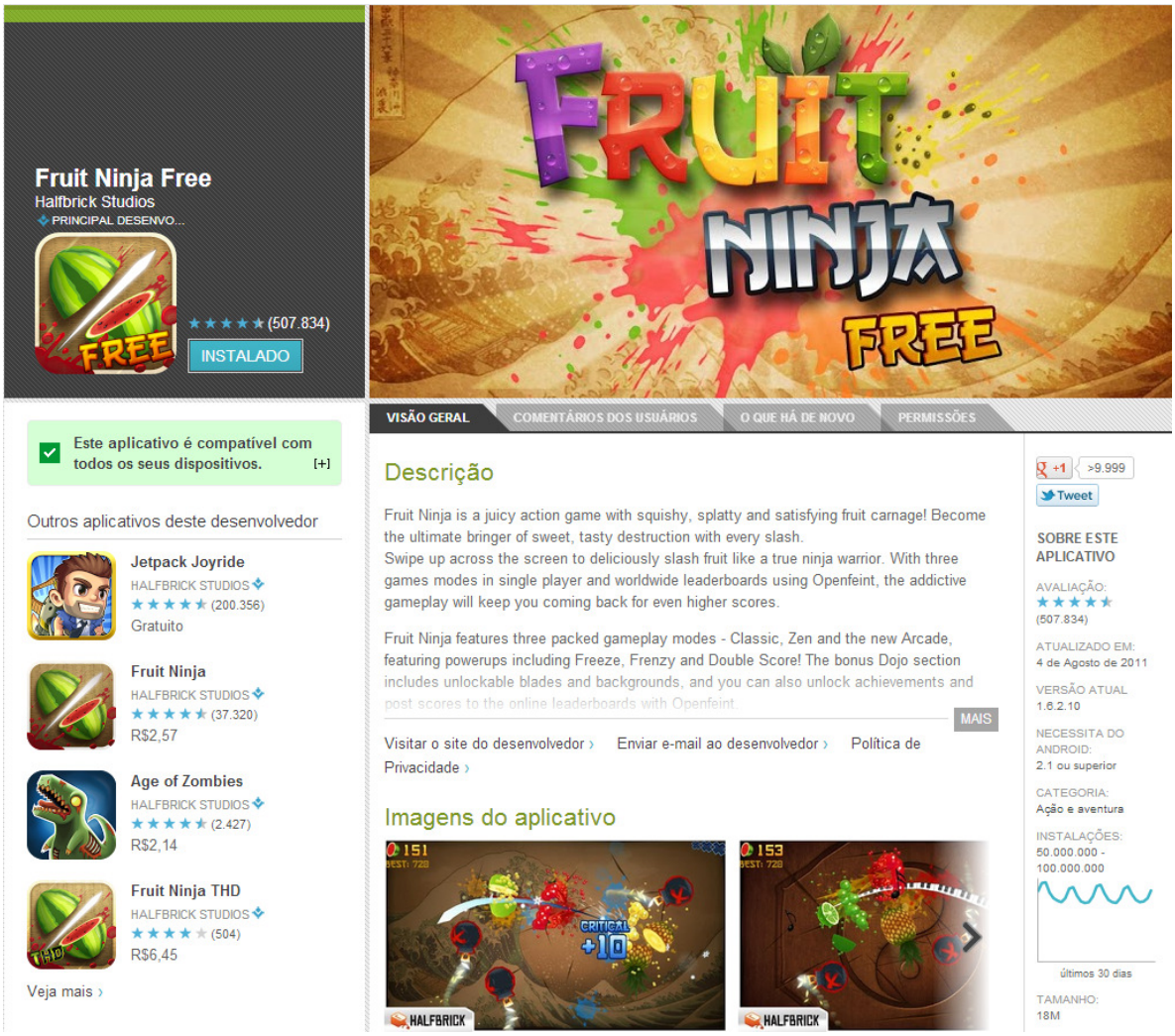

Figura 9 Jogo Fruit Ninja Free

Fonte: FRUIT NINJA FREE (2013)

De acordo com GOOGLE PLAY (2013b), qualquer pessoa ou empresa que tenha uma conta de desenvolvedor no Google Play pode distribuir aplicativos pelo mesmo. Para poder cobrar taxas por seus produtos, é necessário adquirir e manter uma conta de pagamento. Esta conta de pagamento pode ser uma conta Google Checkout, que de acordo com GOOGLE MERCHANT CENTER (2013), é um serviço para tornar a compra e venda na WEB mais rápida, conveniente e segura, ela serve tanto para o comprador como para o vendedor.

Para criar uma conta de desenvolvedor é cobrada uma taxa de $25, e de acordo com GOOGLE PLAY (2013c) será cobrada uma taxa por venda de 30% do valor do aplicativo vendido. Caso o aplicativo seja gratuito não terá taxa de disponibilização.

3 MATERIAIS E MÉTODOS

Neste capítulo serão apresentadas as tecnologias utilizadas para o desenvolvimento do jogo, sendo abordadas as metodologias adotadas para o desenvolvimento do jogo.

O jogo desenvolvido tem como objetivo controlar uma plataforma, como na Figura 10. Esta plataforma pode ser controlada pelo toque na tela ou com o movimento do dispositivo.

Esta plataforma deve evitar que o círculo que se movimenta no sentido vertical alcance a borda inferior da tela. Ao rebater na plataforma, o círculo sobe, e acaba colidindo com os destroços presentados na parte superior da tela. Com a colisão, os destroços são destruídos. Como se trata de um jogo com múltiplas fases, é necessário um controle de fases usando persistência de dados no banco de dados SQLite.

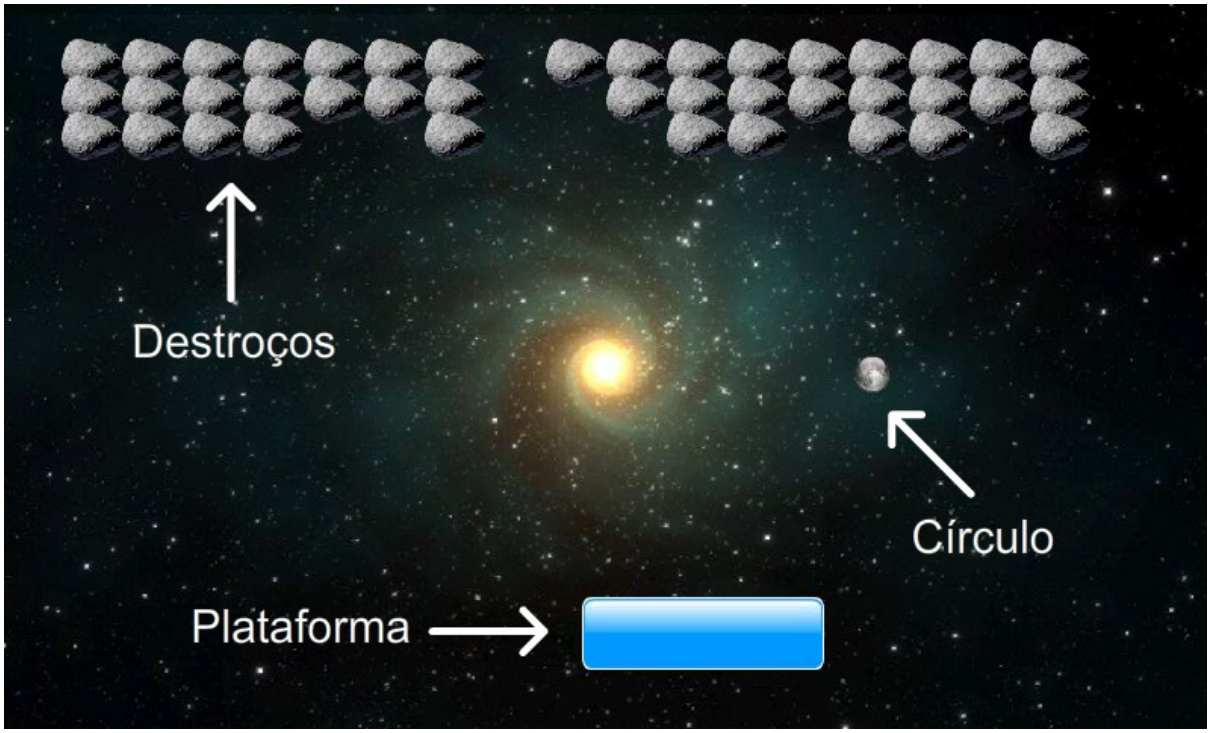

Figura 10 Tela exemplo Jogo

Fonte: Autoria Própria

3.1 MATERIAIS

Para o desenvolvimento do jogo é utilizada a IDE de desenvolvimento NetBeans versão 7.2.1. Para execução e depuração do aplicativo é utilizado o Android SDK versão 21.1. Como *plug-in* para integrar a IDE com o SDK, é utilizado o NBAndroid versão 1.7. Por fim, para a persistência dos dados é utilizado o SQLite versão 3.5.9.

Como é utilizado recursos para enriquecer a jogabilidade do aplicativo, também serão apresentados e aplicados recursos de acelerômetro e interação via tela sensível a toque no presente documento.

Por fim, para disponibilização do jogo para o download via *Internet*, é utilizado o Google Play, ambiente fornecido pela Google para a disponibilização e venda de aplicativos Android.

Na Figura 11 são mostradas as tecnologias aplicadas. Os círculos representam os ambientes de desenvolvimento e disponibilização do aplicativo, as elipses representam as ferramentas utilizadas e os retângulos representam os recursos utilizados.

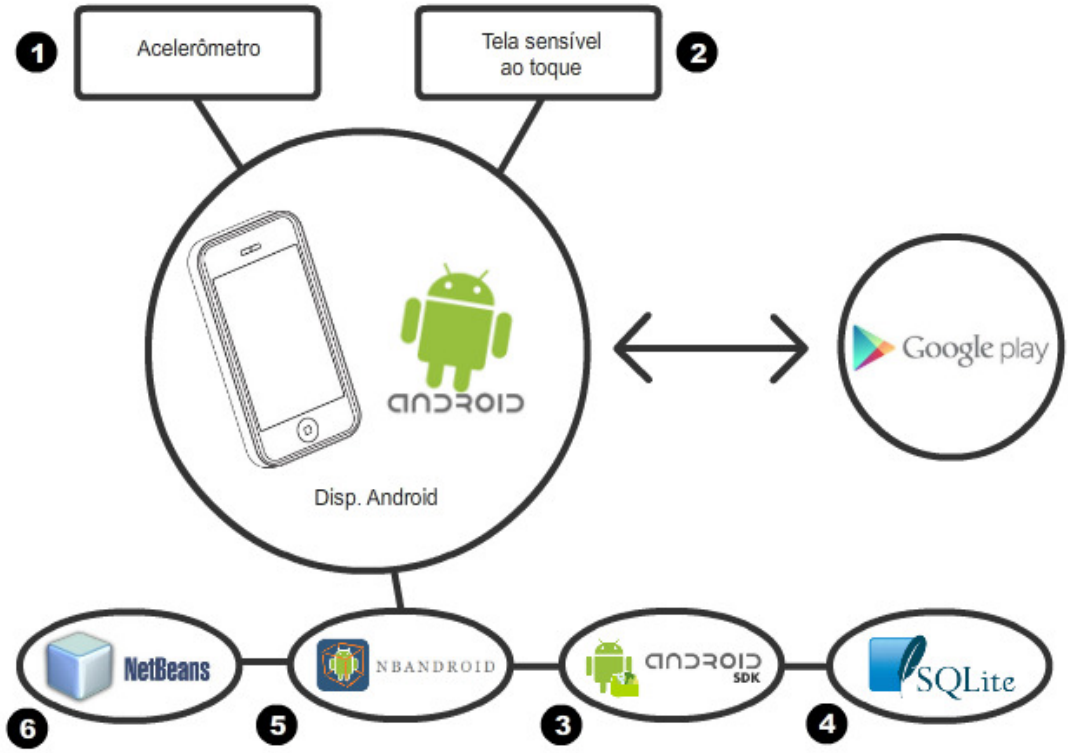

Figura 11 Materiais

Fonte: Autoria Própria.

Na sequência, uma breve descrição de cada recurso utilizado no desenvolvimento do aplicativo:

1. **Acelerômetro**: Recurso que permite ler a rotação do dispositivo em seu próprio eixo.

2. **Tela Sensível ao Toque**: Recurso que permite detectar o toque em na tela do dispositivo, recuperando a posição deste toque.

3. **Android SDK (API 7) versão 21.1**: O Android SDK provê bibliotecas e ferramentas de desenvolvimento necessárias para construir, testar e depurar aplicativos para Android. A ferramenta pode ser obtida através do site ANDROID SDK (2013). Já a API 7 diz respeito a versão do sistema operacional Android usado para testar o aplicativo no emulador, no caso, a API 7 representa o Sistema Operacional Android 2.1, também conhecido pelo codinome Eclair.

4. **SQLite versão 3.5.9**: O SQLite é um banco de dados transacional nativo do sistema Android que não depende de instalação de servidores e não possui necessidade de configurações. É um dos bancos de dados mais utilizados no mundo e que através do SQLITE (2013) é disponibilizado gratuitamente.

5. **NBAndroid versão 1.7:** NBAndroid é um conjunto de módulos do NetBeans que fornece suporte para o desenvolvimento de aplicações Android. É um software de código aberto distribuído sob licença Apache 2.0. O projeto é hospedado em KENAI (2013).

6. **NetBeans versão 7.2.1:** NetBeans IDE permite que você desenvolva em Java para desktop, móvel e aplicações web, enquanto também proporciona ótimas ferramentas para PHP e C / C + +. Ele é gratuito e de código aberto e tem uma grande comunidade de usuários e desenvolvedores de todo o mundo (NETBEANS, 2013).

3.1.1 Android

Em 2005, a Google comprou a Android, Inc, uma empresa que desenvolvia um pequeno sistema para celulares baseado em Linux. O Android, como plataforma e sistema operacional de dispositivos móveis, foi anunciado pela Google em 2007, e segundo MORIMOTO (2010), foi entregue a OHA (Open Handset Alliance) para o desenvolvimento dos componentes *open-source* da plataforma. Um ano depois, em 2008, finalmente foi lançada sua primeira versão, o Android versão 1.0.

O Android é uma plataforma de desenvolvimento que utiliza a linguagem Java. Os aplicativos desenvolvidos para Android são executados em dispositivos com o sistema operacional Android, garantindo uma total integração entre o sistema operacional e a linguagem de programação.

Anunciado em 2007 e com sua primeira versão lançada em 2008, o Android é uma plataforma baseada em Linux, mas segundo CARVALHO (2011), os aplicativos só são executados por usuários comuns via ícones na área de trabalho e estes ícones executam apenas aplicativos Java instalados através do sistema de pacotes APK. Por dentro o formato APK é um arquivo comprimido ZIP com um formato especial, mais especificamente uma modificação do formato JAR utilizado por programas Java.

3.1.2 NetBeans

NetBeans começou como um projeto de estudantes, (originalmente chamado Xelfi), sendo concebido na República Checa em 1996. O objetivo era escrever uma IDE (Integrated Development Environment) para Java como a IDE para Delphi. Xelfi foi a primeira IDE para Java escrito em Java, com seu primeiro pré-lançamento em 1997, conforme informado pela NETBEANS (2013).

O projeto atingiu tal magnitude que seus desenvolvedores, uma vez formados, decidiram comercializar o produto. Logo depois, eles foram contatados

por Roman Stanek[1]. Ele estava procurando uma boa ideia para investir, e descobriu Xelfi.

Em 1999, a antiga Sun Microsystems, hoje Oracle, queria melhores ferramentas de desenvolvimento Java, e tinha se interessado no NetBeans. Com a próxima versão do NetBeans em versão beta, um acordo foi firmado.

Então foi tomada a decisão que o NetBeans seria de código aberto. Em junho de 2000, o site inicial netbeans.org foi lançado. Os anos que se seguiram foram focados em melhorias contínuas de versão para versão, sendo hoje o NetBeans uma referência em desenvolvimento Java para Web, Desktop, Mobile, PHP e C++.

3.1.3 Android SDK

De acordo com o portal ANDROID SDK (2013), o Android SDK fornece as bibliotecas da API e ferramentas de desenvolvimento necessárias para construir, testar e depurar aplicativos para o Android. Além disso, segundo LECHETA (2010) ele possui um emulador com recursos de depuração em qualquer celular conectado em porta USB.

Para fazer o download do Android SDK basta acessar seu portal como na Figura 12.

Pode-se verificar a instalação do Android SDK no APÊNDICE A.

[1] Roman Stanek é um visionário de tecnologia que construiu empresas de classe mundial de tecnologia. Atualmente fundador e CEO da Good Data, que fornece análises de colaboração sob demanda, ele já foi co-fundador do NetBeans, STANEK (2009).

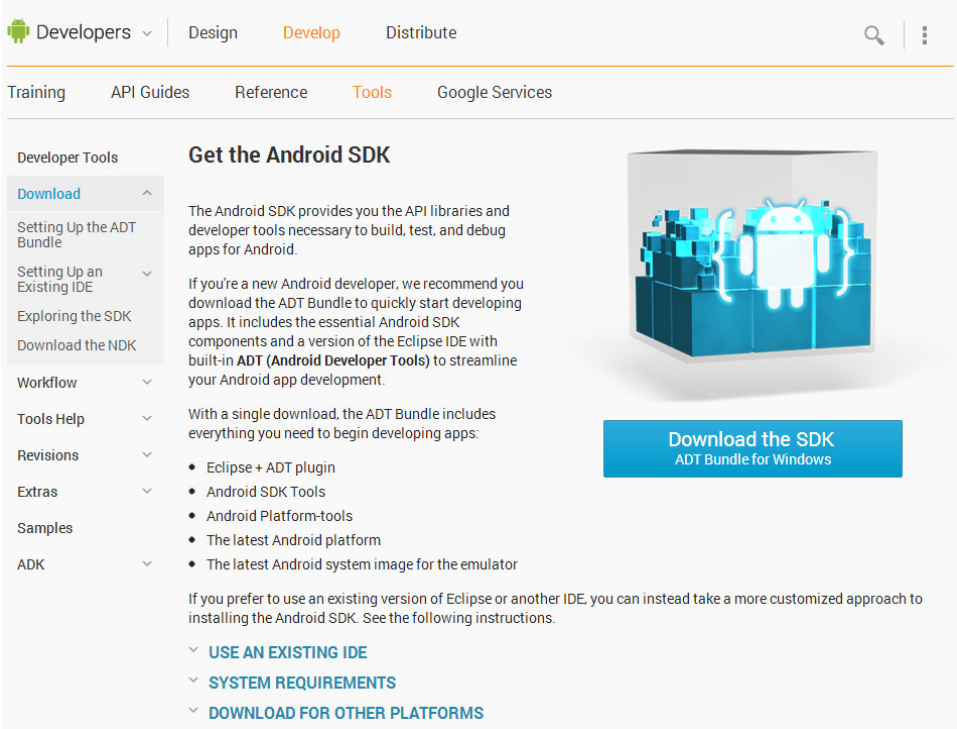

Figura 12 Android SDK

Fonte: ANDROID SDK (2013)

3.1.4 NBAndroid

Este *plug-in* para Netbeans IDE é projetado para fornecer um ambiente poderoso e integrado no desenvolvimento de aplicativos Android, conforme informado pelo NBANDROID (2013).

NBAndroid amplia os recursos do NetBeans para que possa-se criar novos projetos Android, criar interface gráfica de aplicativos, utilizar os pacotes do Android Framework API e depurar os aplicativos usando as ferramentas do Android SDK.

Ele também oferece editores XML personalizados que permitem editar arquivos específicos do Android em uma interface de usuário baseada em formulário. Este *plug-in* também fornece documentação integrada para as APIs do Android, podendo o programador acessar a documentação passando o mouse sobre classes, métodos ou variáveis.

Pode-se verificar a instalação do NBAndroid no APÊNDICE B.

3.1.5 Requisitos

De acordo com NETBEANS (2012) as configurações mínimas de *hardware* para utilizar NetBeans na versão 7.2, que é a versão utilizada para este trabalho, são:

Tabela 1 Requisitos de Sistema do NetBeans

SO	Processador	Memória	Espaço em Disco
Microsoft Windows XP Professional SP3/Vista SP1/Windows 7 Professional	Intel Pentium III 800MHz	512 MB	750 MB
Ubuntu 9.10	Intel Pentium III 800MHz	512 MB	650 MB
Solaris versão 11 Express (SPARC)	UltraSPARC II 450 MHz	512 MB	650 MB
Solaris versão 11 Express (Edição de Plataforma x86/x64)	AMD Opteron 1200 Series 1,8 GHz	512 MB	650 MB
Macintosh OS X 10.6 Intel	Intel Dual Core (32 ou 64 bits)	1 GB	650 MB

Para execução do jogo é necessário ter um dispositivo com Android 2.1 (Eclair) ou superior, esta versão mínima é definida no momento que é criado o projeto Android no NetBeans, pode-se notar esta situação na Figura 25.

3.2 MÉTODOS

O desenvolvimento do jogo foi realizado segundo os seguintes passos:

A) Estudo de Tecnologias: Realizado um estudo sobre as tecnologias que poderiam ser usadas com o Android para o desenvolvimento de um jogo, tais como acelerômetro, tela sensível ao toque e SQLite.

B) Definição do Jogo: Realizada a definição do jogo, bem como um esboço de sua jogabilidade, tendo como base na criação da jogabilidade os estudos sobre a tecnologia realizados anteriormente.

C) Análise: Criado o diagrama de tabelas e de fluxo de telas. Outros diagramas sugeridos pela UML não foram desenvolvidos, dada a simplicidade do jogo.

D) Desenvolvimento do *Framework*: Realizado desenvolvimento de um conjunto de classes em um *framework* para desenvolvimento de jogos, onde este será responsável por desenhar e controlar os objetos em tela.

E) Desenvolvimento do Jogo: Desenvolvimento do jogo usando as tecnologias listadas na seção 3.1 – MATERIAIS e o *framework* desenvolvido na seção 4.4 – DESENVOLVIMENTO DO FRAMEWORK.

F) Distribuição do Jogo: Disponibilização do jogo para *download* gratuito utilizando a plataforma Google Play.

G) Testes: Após disponibilizar o jogo para *download* é possível fazer os testes em vários dispositivos diferenciados.

4 RESULTADOS

O primeiro passo para o desenvolvimento do presente trabalho é o estudo das tecnologias que ajudariam a tornar o jogo mais atrativo, ou seja, acelerômetro e tela sensível a toque. Em seguida, utilizando os resultados do estudo, foi feito a definição da jogabilidade e elementos do jogo a ser desenvolvido neste trabalho. Com os elementos do jogo definidos foi possível fazer a análise de tabelas e o fluxo das telas.

Após, foi iniciado o desenvolvimento do *framework* para tornar mais fácil o desenvolvimento do jogo, além deste facilitar o desenvolvimento de futuros jogos. Por fim, o jogo foi desenvolvido e disponibilizado na plataforma Google Play.

4.1 ESTUDO DE TECNOLOGIAS

As tecnologias escolhidas para estudo foram a tela sensível ao toque e o acelerômetro para a interação com o aplicativo. O banco de dados SQLite também foi estudado para permitir a persistência dos dados referente as fases do jogo.

4.1.1 Tela sensível ao toque

O uso da tela sensível ao toque permite a movimentação dos objetos do jogo de acordo com o toque em tela. Dentro do pacote de classes Android, existe a classe android.view.View, que de acordo com ANDROID DEVELOPERS – VIEW (2013), esta classe representa o bloco de construção básico de componentes de interface do usuário. Uma View ocupa uma área retangular na tela e é o responsável pelo desenho e manipulação de eventos. Na View existe o método onTouchEvent(MotionEvent), sendo possível reescrever este método, recuperando o evento de toque na tela, conforme exemplo na Listagem 1.

Listagem 1 Código de Exemplo de uma View

```
1 import android.app.Activity;
2 import android.view.MotionEvent;
3 import android.view.View;
4
5 public class Tela extends View {
6
7     public float xTouch = 0;
8     public float yTouch = 0;
9
10    public Tela( Activity context ) {
11        super( context );
12    }
13
14    @Override
15    public boolean onTouchEvent( MotionEvent event ) {
16
17        this.xTouch = event.getX();
18        this.yTouch = event.getY();
19        return true;
20
21    }
22
23 }
```

O método onTouchEvent(MotionEvent) envia um objeto do tipo android.view.MotionEvent, que de acordo com ANDROID DEVELOPERS – MOTIONEVENT (2013), traz informações sobre o toque realizado na tela, como quantidade de toques, pressão e posição deste. A informação do MotionEvent que será utilizado no jogo é a posição do toque.

Esta posição é recuperada como referência de um plano cartesiano que representa a tela do dispositivo, dividido em x e y (a posição 0 e 0 para x e y é o canto superior esquerdo). Para recuperar as referências no plano cartesiano utiliza-se os métodos getX() e getY(). Também será necessário o tipo de ação do evento, que pode-se obter com o método getAction(). Os tipos de ação contidos no MotionEvent são mostradas na Listagem 2.

No jogo desenvolvido foram utilizadas apenas as ações ACTION_DOWN (ação disparada quando um toque é iniciado), ACTION_UP (ação disparada quando um toque é finalizado) e ACTION_OUTSIDE (ação disparada quando um toque é arrastado para fora da tela).

Listagem 2 Tipos de Ação de um MotionEvent

```
1 public static final int ACTION_MASK = 255;
2 public static final int ACTION_DOWN = 0;
3 public static final int ACTION_UP = 1;
4 public static final int ACTION_MOVE = 2;
5 public static final int ACTION_CANCEL = 3;
6 public static final int ACTION_OUTSIDE = 4;
7 public static final int ACTION_POINTER_DOWN = 5;
8 public static final int ACTION_POINTER_1_DOWN = 5;
9 public static final int ACTION_POINTER_2_DOWN = 261;
10 public static final int ACTION_POINTER_3_DOWN = 517;
11 public static final int ACTION_POINTER_UP = 6;
```

```
12 public static final int ACTION_POINTER_1_UP = 6;
13 public static final int ACTION_POINTER_2_UP = 262;
14 public static final int ACTION_POINTER_3_UP = 518;
15 public static final int ACTION_POINTER_ID_MASK = 65280;
16 public static final int ACTION_POINTER_ID_SHIFT = 8;
17 public static final int EDGE_TOP = 1;
18 public static final int EDGE_BOTTOM = 2;
19 public static final int EDGE_LEFT = 4;
20 public static final int EDGE_RIGHT = 8;
```

4.1.2 Acelerômetro

O acelerômetro foi estudado para dar alternativas de jogabilidade ao aplicativo, por exemplo, no lugar de fazer o usuário tocar na tela para movimentar seu personagem no jogo, ele pode simplesmente movimentar o *smartphone*.

Para usar o acelerômetro, a classe android.hardware.Sensor é necessária. De acordo com ANDROID DEVELOPERS – SENSOR (2013), para utilizar o sensor de aceleração, deve-se informar para a classe Sensor que se deseja utilizar o acelerômetro do dispositivo. Como podem existir vários sensores no dispositivo (aproximação, temperatura, giroscópio, entre outros), deve-se utilizar a classe android.hardware.SensorManager para a seleção do sensor desejado. De acordo com ANDROID DEVELOPERS – SENSORMANAGER (2013), esta classe permite que o programador acesse um sensor específico do dispositivo, conforme Listagem 3.

Listagem 3 Código exemplo de SensorManager e Sensor

```
 1 import android.hardware.Sensor;
 2 import android.hardware.SensorManager;
 3 import android.view.View;
 4 import android.app.Activity;
 5
 6 public class Tela extends View {
 7
 8     private Sensor acelerometro;
 9     private SensorManager aceleManger;
10
11     public Tela( Activity context ) {
12
13         super( context );
14
15         this.aceleManger = (SensorManager) context
16                 .getSystemService( SGScreenC.context.SENSOR_SERVICE );
17         this.acelerometro = this.aceleManger
18                 .getDefaultSensor( Sensor.TYPE_ACCELEROMETER );
19
20     }
21
22 }
```

Para utilizar um sensor, é necessário criar uma instância de SensorManager, Listagem 3 – linhas 15 e 16, sendo este um serviço oferecido pelo próprio sistema operacional Android, o qual possui uma referência a todos os sensores.

Seguindo, é informado o tipo de sensor que será utilizado, no exemplo, linhas 18 e 19, o sensor de acelerômetro. Para recuperar o momento exato de uma movimentação (evento gerado pelo acelerômetro), é usada a interface android.hardware.SensorEventListener, que de acordo com ANDROID DEVELOPERS – SENSOREVENTLISTENER (2013) ao ser utilizada, deve-se recuperar o evento a partir do método onSensorChanged(SensorEvent), este apresentado na Listagem 4, linhas 27 a 33.

Listagem 4 Código exemplo de implementação SensorEventListener

```
 1 import android.hardware.Sensor;
 2 import android.hardware.SensorEvent;
 3 import android.hardware.SensorEventListener;
 4 import android.hardware.SensorManager;
 5 import android.view.View;
 6 import android.app.Activity;
 7
 8 public class Tela extends View implements SensorEventListener {
 9
10     private Sensor acelerometro;
11     private SensorManager aceleManger;
12
13     public Tela( Activity context ) {
14
15         super( context );
16
17         this.aceleManger = (SensorManager) context
18                 .getSystemService( SGScreenC.context.SENSOR_SERVICE );
19         this.acelerometro = this.aceleManger
20                 .getDefaultSensor( Sensor.TYPE_ACCELEROMETER );
21         this.aceleManger.registerListener(
22                 this, this.acelerometro,
23                 SensorManager.SENSOR_DELAY_NORMAL );
24
25     }
26
27     public void onSensorChanged( SensorEvent event ) {
28         System.out.println( "<"
29                             +event.values[0]+","
30                             +event.values[1]+","
31                             +event.values[2]
32                             +">" );
33     }
34
35     public void onAccuracyChanged( Sensor arg0, int arg1 ) {
36
37     }
38
39 }
```

O método onSensorChanged(SensorEvent) recebe e envia um objeto do tipo android.hardware.SensorEvent, este objeto tem o atributo values. Como este método é genérico (pode ser usado por qualquer sensor), o valor retornado é um *array*. De acordo com ANDROID DEVELOPERS – SENSOREVENT (2013) no sensor do tipo

acelerômetro, este *array* possui três informações, sendo o valor da aceleração nos eixos x, y e z.

4.1.3 SQLite

O SQLite é utilizado no armazenamento dos dados do jogo, especificamente para gerenciar as fases. O SQLite é um pacote de classes que já vem com o Android SDK, sendo este um recurso nativo do sistema operacional.

Para criar um banco de dados, uma forma é usar a classe abstrata android.database.sqlite.SQLiteOpenHelper. De acordo com ANDROID DEVELOPERS – SQLITEOPENHELPER (2013) esta é uma classe auxiliar para gerenciar a criação de banco de dados e gerenciamento de versão. Codificando os métodos onCreate(SQLiteDatabase) e onUpgrade (SQLiteDatabase, int, int), esta classe se encarrega de abrir o banco de dados ou criá-lo se este não existe, e atualizá-lo, se necessário, como exemplificado na Listagem 5.

Listagem 5 Código exemplo de SQLiteOpenHelper

```
1  import android.database.sqlite.SQLiteDatabase;
2  import android.database.sqlite.SQLiteOpenHelper;
3
4  public class FaseDAO extends SQLiteOpenHelper {
5
6      public static final String TABELA = "Fase";
7      public static final int VERSION = 4;
8      public static final String[] COLUNAS = { "id", "grupo", "nome",
9                                               "nivel", "finalizado" };
10
11     public FaseDAO() {
12         super( SGScreenC.context, TABELA, null, VERSION );
13     }
14
15     @Override
16     public void onCreate( SQLiteDatabase db ) {
17         String sql =     "CREATE TABLE " + TABELA + " ( " +
18                          " id INTEGER PRIMARY KEY," +
19                          " grupo INTEGER, " +
20                          " nome TEXT, " +
21                          " nivel INTEGER, " +
22                          " finalizado INTEGER " +
23                          ");";
24         db.execSQL( sql );
25     }
26
27     @Override
28     public void onUpgrade( SQLiteDatabase db,
29                                 int oldVersion, int newVersion) {
30         if ( oldVersion!=newVersion ) {
31             db.execSQL( "DROP TABLE IF EXISTS " + TABELA );
32             this.onCreate( db );
33         }
34     }
35 }
```

O método onCreate(SQLiteDatabase), linhas 15 a 25, cria o banco se ele não existe. Com o método onUpgrade (SQLiteDatabase, int, int), linhas 27 a 34, pode-se verificar a versão do banco e alterá-lo. E se essa classe for configurada como um DAO[2] também será adicionado os comandos de *insert*, *update* e *delete*, como apresentada na Listagem 6.

Listagem 6 Código de Insert, Delete e Update

```
1     public void insert( Fase fase ) {
2
3         ContentValues valores = new ContentValues();
4         valores.put( "id", fase.getId() );
5         valores.put( "grupo", fase.getGrupo() );
6         valores.put( "nome", fase.getNome() );
7         valores.put( "nivel", fase.getNivel() );
8         valores.put( "finalizado", fase.getFinalizado() );
9
10        this.getWritableDatabase().insert(FaseDAO.TABELA,null, valores);
11        this.close();
12
13    }
14    public void update( Fase fase ) {
15
16        ContentValues valores = new ContentValues();
17        valores.put( "id", fase.getId() );
18        valores.put( "grupo", fase.getGrupo() );
19        valores.put( "nome", fase.getNome() );
20        valores.put( "nivel", fase.getNivel() );
21        valores.put( "finalizado", fase.getFinalizado() );
22
23        this.getWritableDatabase().update(
24            FaseDAO.TABELA, valores,
25            "id = " + fase.getId(), null
26        );
27        this.close();
28
29    }
30    public void delete( Fase fase ) {
31
32        this.getWritableDatabase().delete(
33            FaseDAO.TABELA,
34            "id = " + fase.getId(), null
35        );
36        this.close();
37
38    }
```

Para poder alterar o banco, uma alternativa é utilizar o método getWritableDatabase(), o qual cria ou abre um banco de dados que será usado para a leitura e escrita.

[2] Este padrão de projeto permite criar as classes de dados independentemente da fonte de dados. A fonte pode ser um BD relacional, um arquivo texto, um arquivo XML, um banco OO, entre outros. Para isso, ele encapsula os mecanismos de acesso a dados e cria uma interface de cliente genérica, para fazer o acesso aos dados permitindo que os mecanismos de acesso a dados sejam alterados independentemente do código que utiliza os dados. MACORATTI (2013)

Depois de aberto com sucesso, o banco de dados é armazenado em *cache*, assim pode-se chamar esse método cada vez que for necessário alterar o banco de dados.

4.2 DEFINIÇÃO DO JOGO

Uma visão geral do jogo foi apresentada no início do capítulo 3. Desta forma, a seguir serão apresentados os elementos do jogo, Plataforma, Círculo e Destroços.

Para o desenvolvimento, primeiro foi definido os objetos pertencentes a interação do jogo, sendo eles:

A) Plataforma: objeto de formato retangular posicionado na parte inferior da tela. Este objeto é controlado pelo usuário, seja via toque na tela ou com o movimento do dispositivo, conforme Figura 13.

Figura 13 Plataforma
Fonte: Autoria Própria

B) Círculo: objeto de formato circular que irá se movimentar pela tela de forma autônoma. Este objeto, ao colidir com a plataforma, ou com a borda superior da tela, ou ainda com algum destroço muda o sentido do seu movimento. O objetivo do jogo é fazer com que este círculo nunca alcance a borda inferior da tela – Figura 14.

Figura 14 Círculo
Fonte: Autoria Própria

C) Destroços: objetos que ficarão espalhados pela parte superior da tela, estes serão eliminados um a um pelo círculo – Figura 15.

Figura 15 Destroços
Fonte: Autoria Própria

Assim, este será um jogo 2D onde o usuário controla a plataforma, escolhendo previamente, nas opções do jogo, a forma de controle (toque em tela ou movimentando o dispositivo). A plataforma se movimentaria da esquerda para a direita até os limites da tela.

Os destroços ficam espalhados pela tela imóveis, e podem ser destruídos com contato com o círculo.

O Círculo se movimenta em linha reta (sentido vertical).

O objetivo do jogo é destruir todos os destroços sem deixar o círculo colidir com a borda inferior da tela.

4.3 ANÁLISE

Com a definição do jogo (seção 4.2) pronta, se tem as funcionalidades do jogo, desta forma foi possível realizar a análise.

Dada a simplicidade do jogo do ponto de vista de análise, foi desenvolvido apenas o diagrama de entidade e relacionamento das tabelas e o diagrama de fluxo de telas.

4.3.1 Diagrama de Entidade e Relacionamento das Tabelas

O jogo possui apenas três tabelas: GrupoFase, Fase e FaseItem, como pode-se verificar na Figura 16.

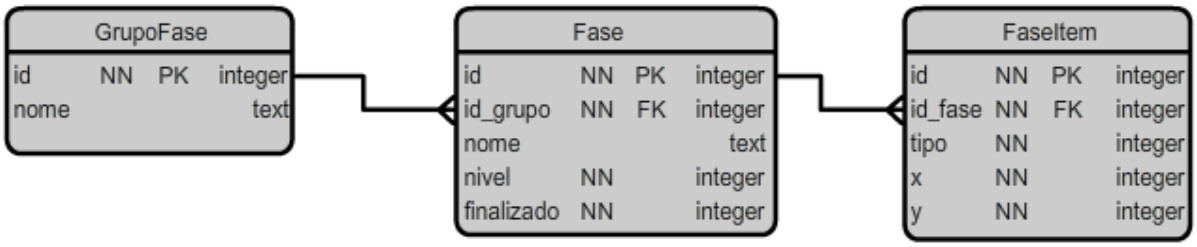

Figura 16 Diagrama de Tabelas

Fonte: Autoria Própria.

A função das tabelas serão:

A) GupoFase: Armazenar os grupos de fases. Um grupo de fase pode ter até nove Fases. O GrupoFase tem um código representado pelo id e descrição representado pelo nome da fase. Foi criado o grupo de fase para facilitar a separação das fases, assim pode-se mostrar as fases em grupos, sendo mais fácil para o jogador escolher qual jogar.

B) Fase: Armazena as fases do jogo. Tem código representado pelo id, descrição representado pelo nome, é vinculada ao GrupoFase pelo id_grupo, o campo nível é utilizado para saber qual sua posição na lista e também, quanto maior o nível mas difícil a fase, e o campo finalizado para gravar se a fase já foi completada.

C) FaseItem: Representa os itens da fase, como os destroços por exemplo, tem código representado pelo id, é vinculado a fase pelo id_fase, tem tipo para caso haja vários tipos de itens de fase (como destroços, círculo, plataforma), e as posições x e y na tela.

4.3.2 Diagrama de Fluxo de Telas

No diagrama de fluxo de telas é apresentada a navegação entre as telas do jogo, sendo a ordem que estas são apresentadas para o usuário. Este diagrama é apresentada na Figura 17.

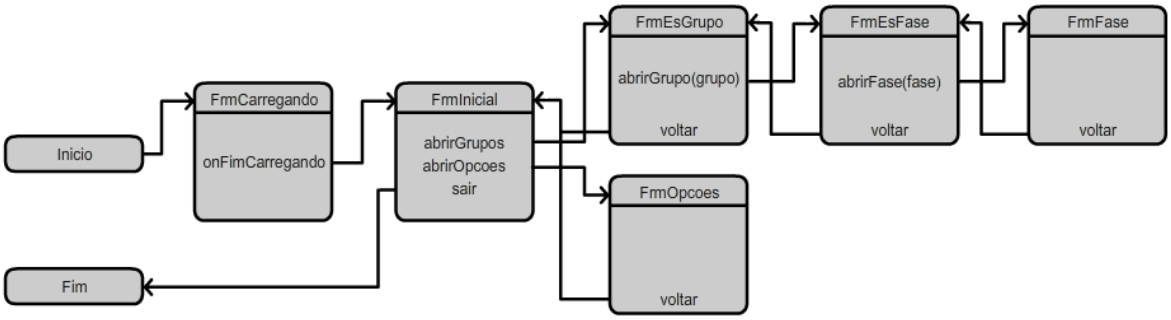

Figura 17 Diagrama de Fluxo de Telas

Fonte: Autoria Própria.

As telas do jogo são:

A) FrmCarregando: Esta tela é apresentada assim que o jogo é iniciado. Ela é apresentada enquanto são carregados alguns recursos necessários para o jogo, como por exemplo, imagens para os botões e a imagem para o plano de fundo de tela. Assim que o processo terminar ela chama a tela FrmInicial.

B) FrmInicial: Tela inicial do jogo, onde pode-se escolher entre chamar FrmOpcoes ou FrmEsGrupo.

C) FrmOpcoes: Nesta tela pode-se configurar a jogabilidade do jogo entre jogar com toque em tela ou movimentando o dispositivo.

D) FrmEsGrupo: Nesta tela é apresentada os grupos cadastrados na tabela GrupoFase. Onde escolhendo um grupo de fase são apresentadas as fases referentes a esta no FrmEsFase.

E) FrmEsFase: Nesta tela são apresentadas as fases cadastradas na tabela Fase para o grupo de fases escolhido na tela anterior. Ao escolher uma fase, o jogo é iniciado, apresentando os itens da fase na janela FrmFase.

F) FrmFase: Nesta tela é utilizado os itens cadastrados na tabela FaseItem para a fase escolhida. Esta é a tela do jogo, nela, o usuário pode interagir com os elementos.

4.4 DESENVOLVIMENTO FRAMEWORK

O desenvolvimento de um jogo pode ser decomposto em duas partes: uma área de desenho na tela do computador e um grande *looping* para atualizar os desenhos na tela, dando a sensação de movimento.

Como área de desenho citado no parágrafo anterior, foi utilizado o Canvas, este é um recurso disponível na maioria das linguagens de programação que permite desenhar em baixo nível na tela do computador (pixel a pixel).

4.4.1 Desenho com Canvas

Dentro do Android, a classe que permite desenhar na tela é a android.graphics.Canvas. De acordo com ANDROID DEVELOPERS – CANVAS (2013), esta classe possui o método "*draw*", que é responsável pelo desenho de elementos na tela. Para desenhar algo, precisa-se de quatro componentes básicos: um container para manter os pixels, um servidor para o Canvas executar suas chamadas (escrevendo no container), um desenho primitivo (por exemplo, Rect, Path, Text, Bitmap) e um Paint (para descrever as cores e estilos para o desenho).

Desta forma, o primeiro passo é achar o servidor para o Canvas. Para o jogo, o servidor usado foi o android.view.View que tem o método onDraw(Canvas), o qual implementa um container do tipo Canvas.

Tendo o Canvas, podemos desenhar em tela com ele, e para isso usa-se a classe android.graphics.Paint. De acordo com ANDROID DEVELOPERS – PAINT (2013) esta classe permite o desenho de formas geometrias, texto e bitmaps. Um exemplo de sua utilização está disponível na Listagem 7. O resultado da execução deste código é apresentado na Figura 18.

Listagem 7 Código exemplo de Canvas com onDraw na View

```
 1 import android.content.Context;
 2 import android.graphics.Canvas;
 3 import android.graphics.Color;
 4 import android.graphics.Paint;
 5 import android.view.View;
 6
 7 public class TesteTela extends View {
 8
 9     public TesteTela(Context context) {
10         super(context);
11     }
```

```
12
13      @Override
14      protected void onDraw(Canvas canvas) {
15
16          Paint paint = new Paint();
17          canvas.drawPaint( paint );
18
19          paint.setColor( Color.WHITE );
20          canvas.drawRect( 30, 30, 180, 180, paint );
21          paint.setColor( Color.BLUE );
22          canvas.drawRect( 110, 110, 260, 260, paint );
23
24          super.onDraw( canvas );
25
26      }
27
28  }
```

O código apresentado tem a função de desenhar dois quadrados, o primeiro da cor branca e o segundo da cor azul. Características como posição inicial e tamanho são passados no método de desenho, como por exemplo, o método drawRect().

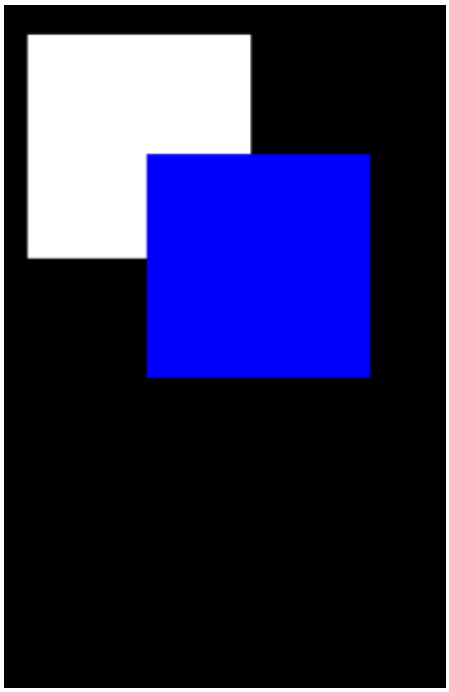

Figura 18 Resultado exemplo de Canvas com onDraw na View

Fonte: Autoria Própria

Com a ferramenta de desenho definida, falta o laço de repetição, um *looping* para atualizar os desenhos do jogo, definindo quantas vezes por segundo a tela do jogo será atualizada. Quanto maior essa taxa, maior a fluidez visual do jogo, porém, mais recurso de processamento e memória será necessário.

4.4.2 Atualização de tela com Thread e Runnable

Uma forma de programar a atualização da tela é utilizando uma instância da classe java.lang.Thread. De acordo com ANDROID DEVELOPERS – THREAD (2013), a Thread é uma unidade de execução simultânea. Ele tem sua própria pilha de chamadas de métodos e seus argumentos e variáveis locais. Juntamente com a Thread, usa-se a interface java.lang.Runnable que de acordo com ANDROID DEVELOPERS – RUNNABLE (2013) ela representa a sequência de código que será executado pela Thread.

Desta forma, a implementação da interface Runnable obriga a implementação do método run(), método este que possui o conjunto de código que será executado pela Thread. A classe Thread pode ser instanciada recebendo uma classe com Runnable implementada. Assim, dentro do método run() será adicionado dois códigos, postInvalidate() e Thread.sleep(long), conforme Listagem 8, linhas 35 e 36. O método Thread.sleep(long) faz com que a Thread espere o tempo determinado em milissegundos para continuar a execução, já o método postInvalidate() vai forçar a View a chamar novamente o método onDraw(Canvas), assim cria-se um laço de repetição para atualização da tela.

Listagem 8 Código exemplo de implementação de Runnable em View

```
 1 import android.content.Context;
 2 import android.graphics.Canvas;
 3 import android.graphics.Color;
 4 import android.graphics.Paint;
 5 import android.view.View;
 6
 7 public class SGScreen extends View implements Runnable {
 8
 9     private Boolean running = true;
10     private Thread thread;
11     private int posX = 0;
12
13     public SGScreen( Context context ) {
14         super( context );
15
16         this.thread = new Thread( this );
17         this.running = true;
18         this.thread.start();
19
20     }
21
22     @Override
23     protected void onDraw( Canvas canvas ) {
24         this.posX+=5;
25         Paint paint = new Paint();
26         canvas.drawPaint( paint );
27         paint.setColor( Color.WHITE );
28         canvas.drawRect( this.posX+30, 30, this.posX+180, 180, paint );
29         super.onDraw( canvas );
30
31     }
```

```
32      public void run() {
33          while ( this.running ) {
34              try {
35                  this.postInvalidate();
36                  Thread.sleep( 40 );
37              } catch ( Exception e ) {
38                  running = false;
39              }
40          }
41      }
42  }
```

A Listagem apresentada anteriormente é a estrutura básica da tela do jogo desenvolvido neste trabalho. Esta será chamada de SGScreen.

4.4.3 Configuração do Activity

Em um projeto Android, deve-se ter pelo menos uma instância da classe android.app.Activity. De acordo com ANDROID DEVELOPERS – ACTIVITY (2013), a Activity, de modo geral, é a classe que gerenciará a interface gráfica. É ela quem controla o ciclo de vida de uma aplicação Android e que gerencia a interação com o usuário.

O Activity é a primeira classe executada em um projeto Android. É de responsabilidade de Activity iniciar uma interface visual como, por exemplo, apresentar um componente View na tela do dispositivo.

Usando o NetBeans, ao criar um projeto é criada automaticamente uma Activity e uma interface visual, conforme Listagem 9.

Listagem 9 Código exemplo de Activity

```
 1 import android.app.Activity;
 2 import android.os.Bundle;
 3
 4 public class MainActivity extends Activity {
 5
 6     @Override
 7     public void onCreate(Bundle savedInstanceState) {
 8
 9         super.onCreate(savedInstanceState);
10         this.setContentView( R.layout.main );
11
12     }
13
14 }
```

O método onCreate(Bundle), linha 7, oferece um objeto da classe android.os.Bundle, que de acordo com ANDROID DEVELOPERS – BUNDLE (2013) a função é guardar o estado do aplicativo em sua última execução. Também dentro

do método onCreate(Bundle) foi utilizando o método setContetView(View), linha 10, este tem a função de abrir uma interface gráfica, esta criada a partir de um XML, apresentada na Listagem 10.

Listagem 10 Exemplo de View por XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >
7  <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="Hello World, MainActivity"
11     />
12 </LinearLayout>
```

Essa tela padrão, criada pelo NetBeans, tem a função de apresentar um texto estático na tela do usuário – “Hello World, MainActivity”, conforme apresentada na Figura 19.

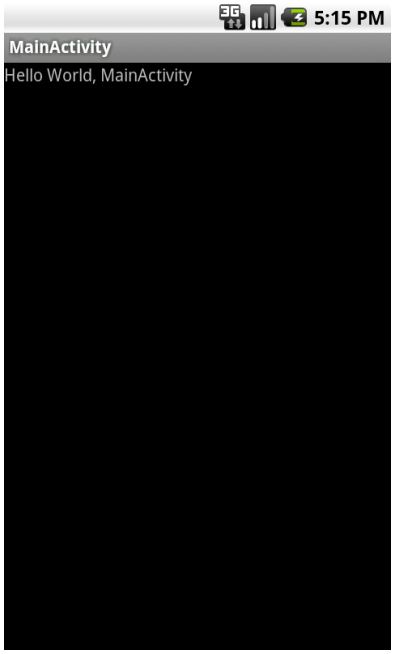

Figura 19 Resultado de View por XML

Fonte: Autoria Própria

Como se pode observar, é apresentada a *Barra de Status* padrão do Android no topo, logo abaixo o titulo da Activity atual, mas na maioria dos jogos isso não é necessário, já que eles são jogados em tela cheia (*full-screen*), ocupando toda a

área útil da tela. Para deixar o aplicativo em tela inteira, utiliza-se o código da Listagem 11, linhas 13 a 15.

Listagem 11 Exemplo de configuração de Activity

```
 1 import android.app.Activity;
 2 import android.os.Bundle;
 3 import android.view.Window;
 4 import android.view.WindowManager;
 5
 6 public class MainActivity extends Activity {
 7
 8     @Override
 9     public void onCreate(Bundle savedInstanceState) {
10
11         super.onCreate(savedInstanceState);
12
13         requestWindowFeature( Window.FEATURE_NO_TITLE );
14         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
15                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
16
17         this.setContentView( R.layout.main );
18
19     }
20
21 }
```

O método da linha 13 retira o título da Activity, e o método das linhas 14 e 15, faz com que o aplicativo fique em tela cheia, escondendo a *Barra de Status* do Android, conforme Figura 20.

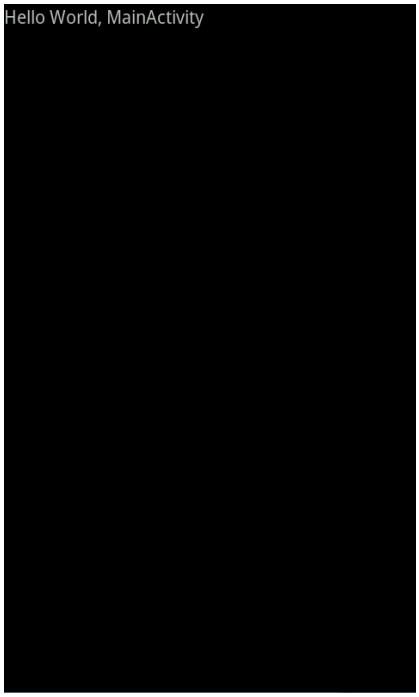

Figura 20 Resultado de configuração de Activity

Fonte: Autoria Própria

Assim, foi criada uma classe padrão do Activity no *framework*, esta chamada de SGActivity. Para facilitar a chamada de novas telas, foi codificado o método openScreen(SGScreen), como apresentado na Listagem 12, linhas 19 a 21, onde é executado o método setContetView(View).

Listagem 12 Exemplo de Activity com openScreen

```
1  import android.app.Activity;
2  import android.os.Bundle;
3  import android.view.Window;
4  import android.view.WindowManager;
5
6  public class MainActivity extends Activity {
7
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10
11         super.onCreate(savedInstanceState);
12
13         requestWindowFeature( Window.FEATURE_NO_TITLE );
14         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
15                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
16
17     }
18
19     public void openScreen( SGScreen screen ) {
20         this.setContentView( screen );
21     }
22
23 }
```

4.4.4 Criando um objeto genérico

Em termos de orientação a objeto, a criação de um objeto genérico que pudesse ser pai de todos os outros objetos de tela facilitaria do ponto de vista de reaproveitamento de código. Esta técnica interessante diminui a complexidade e a quantidade de código. Assim, para o presente *framework* foi criada uma classe genérica com os métodos para desenhar em tela, armazenar sua posição na tela, seu tamanho, entre outras características visuais.

Esse objeto, chamado de SGObject, será adicionado a uma SGScreen, assim, este ficará responsável em ler seus atributos e desenhá-lo na tela. O código inicial do SGObject é apresentado na Listagem 13.

Listagem 13 Início de SGObject

```
1 public class SGObject {
2
3     public float x = 0;
4     public float y = 0;
5     public float width = 0;
6     public float height = 0;
7
8     public SGObject() {
9         this.x = 0;
10        this.y = 0;
11        this.width = 0;
12        this.height = 0;
13    }
14
15 }
```

Como esse objeto será desenhado em tela, é necessário uma instância da classe android.graphics.Paint, este possuindo os métodos de desenho do android.graphics.Canvas, deixando as configurações de desenho dentro do objeto, assim, pode-se ter configurações diferentes para cada um deles, conforme exemplo da Listagem 14.

Listagem 14 Adição de configuração de desenho do SGObject

```
1 import android.graphics.Color;
2 import android.graphics.Paint;
3
4 public class SGObject {
5
6     public Paint paint = new Paint();
7
8     public float x = 0;
9     public float y = 0;
10    public float width = 0;
11    public float height = 0;
12
13    public SGObject() {
14        this.x = 0;
15        this.y = 0;
16        this.width = 0;
17        this.height = 0;
18        this.paint.setColor( Color.WHITE );
19    }
20
21 }
```

Também foi criado dois metodos para tratar a cor utilizada no desenho pelo objeto Paint. Esses métodos foram chamados de getColor() e setColor(int), como apresentado na Listagem 15.

Listagem 15 Métodos para mudar cor do objeto

```
1     public int getColor() {
2         return this.paint.getColor();
3     }
4     public void setColor(int color) {
5         this.paint.setColor( color );
6     }
```

Esse objeto genérico também deve ter a possibilidade de aceitar outros objetos dentro dele, como por exemplo, um botão terá imagens utilizadas para pintar o fundo, também um texto para apresentar sua função. Desta forma, será criado uma lista de SGObject, bem como seus métodos para adicionar e remover itens, como pode ser observado na Listagem 16. A lista ArrayList<SGObject> listSGObject, linha 14, e os dois métodos addSGObject(SGObject) e removeSGObject(SGObject), linhas 24 a 30, estão codificados nesta listagem.

Listagem 16 Adicionando lista de SGObject no próprio SGObject

```
 1 import android.graphics.Color;
 2 import android.graphics.Paint;
 3 import java.util.ArrayList;
 4
 5 public class SGObject {
 6
 7     public Paint paint = new Paint();
 8
 9     public float x = 0;
10     public float y = 0;
11     public float width = 0;
12     public float height = 0;
13
14     private ArrayList<SGObject> listSGObject =new ArrayList<SGObject>();
15
16     public SGObject() {
17         this.x = 0;
18         this.y = 0;
19         this.width = 0;
20         this.height = 0;
21         this.paint.setColor( Color.WHITE );
22     }
23
24     public boolean addSGObject( SGObject obj ) {
25         return this.listSGObject.add( obj );
26     }
27
28     public boolean removeSGObject( SGObject obj ) {
29         return this.listSGObject.remove( obj );
30     }
31
32 }
```

Com a inclusão deste ArrayList, cria-se um problema referente a posição do objeto, pois agora tem-se objetos dentro de objetos, e quando troca-se a posição de um objeto é necessário que se troque a posição de todos os objetos dentro dele. Para solucionar este problema, será criado atributos para trabalhar com a posição global do SGObject, ou seja, a posição em tela dele, e os atributos x e y serão relativos a posição dentro do objeto pai.

Pode-se observar na Listagem 17 que foi adicionado os atributos xGlobal, linha 16, e yGlobal, linha 17, para o controle da posição global do objeto, owner, linha 9, para o acesso ao pai deste objeto e na função addSGObject(SGObject) foi adicionado o código obj.owner = this;, linha 30, para atribuir o pai a esses objetos.

Listagem 17 Adicionado controle de posição relacionado a lista de SGObject

```
1  import android.graphics.Color;
2  import android.graphics.Paint;
3  import java.util.ArrayList;
4
5  public class SGObject {
6
7      public Paint paint = new Paint();
8
9      protected SGObject owner;
10
11     public float x = 0;
12     public float y = 0;
13     public float width = 0;
14     public float height = 0;
15
16     public float xGlobal = 0;
17     public float yGlobal = 0;
18
19     private ArrayList<SGObject> listSGObject =new ArrayList<SGObject>();
20
21     public SGObject() {
22         this.x = 0;
23         this.y = 0;
24         this.width = 0;
25         this.height = 0;
26         this.paint.setColor( Color.WHITE );
27     }
28
29     public boolean addSGObject( SGObject obj ) {
30         obj.owner = this;
31         return this.listSGObject.add( obj );
32     }
33
34     public boolean removeSGObject( SGObject obj ) {
35         return this.listSGObject.remove( obj );
36     }
37
38 }
```

Agora falta criar um método padrão que será chamado pela SGScreen para atualizar o objeto e desenhá-lo em tela. Pode-se observar na Listagem 18 que foram criados dois métodos, o método update(Canvas), linha 1, e onUpdate(Canvas), linha 10, desta forma o método update será chamado pela SGScreen e ele será responsável por atualizar a posição global do objeto, chamar o método update da lista objetos internos e chamar o método onUpdate que ficara responsável por desenhar o objeto em tela.

Listagem 18 Métodos para atualização e desenho do SGObject

```
1      public void update( Canvas canvas ) {
2
3          this.xGlobal = this.x;
4          this.yGlobal = this.y;
5          if ( this.owner!=null ) {
6              this.xGlobal += this.owner.xGlobal;
7              this.yGlobal += this.owner.yGlobal;
8          }
9
10         this.onUpdate( canvas );
11         for ( SGObject obj : this.listSGObject ) {
12             obj.update( canvas );
13         }
14
```

```
15      }
16
17      public void onUpdate( Canvas canvas ) {
18          canvas.drawRect(
19              this.xGlobal,
20              this.yGlobal,
21              this.xGlobal+this.width,
22              this.yGlobal+this.height,
23              this.paint
24          );
25      }
```

No exemplo da listagem apresentada, o método onUpdate desenha um retângulo em tela, usando o método drawRect() de Canvas, isso é usado apenas para não deixar o método vazio, pois na prática, quando será utilizado este método, ele será reescrito pela classe que será estendia a SGObject, exemplo na seção 4.4.6.

4.4.5 Alterando SGScreen para trabalhar com SGObject

Agora será criado os métodos para trabalhar com SGObject na SGScreen. Como se pode notar na Listagem 19, foi adicionado uma lista de SGObject chamada listSGObject, linha 12, e os métodos addSGObject(SGObject) e removeSGObject(SGObject), linhas 23 a 29, para controlar os objetos na lista, note que nesse caso não informa-se o objeto pai ao SGObject.

Listagem 19 Adicionando lista de SGObject na SGScreen

```
 1 import android.content.Context;
 2 import android.view.View;
 3 import br.com.lrmarkanjo.teste.SGObject;
 4 import java.util.ArrayList;
 5
 6 public class SGScreen extends View implements Runnable {
 7
 8     private Boolean running = true;
 9     private Thread thread;
10     private int posX = 0;
11
12     private ArrayList<SGObject> listSGObject =new ArrayList<SGObject>();
13
14     public SGScreen( Context context ) {
15         super( context );
16
17         this.thread = new Thread( this );
18         this.running = true;
19         this.thread.start();
20
21     }
22
23     public void addSGObject( SGObject obj ) {
24         this.listSGObject.add( obj );
25     }
26
27     public void removeSGObject( SGObject obj ) {
```

```
28          this.listSGObject.remove( obj );
29      }
```

Assim, como apresentado na Listagem 20, cria-se um método chamado onUpdate(), para que ao estender uma classe, a SGScreen possa reescreve-la para pegar sua atualização de tela, e o método onDraw() foi modificado para chamar o seu próprio onUpdate, linha 11, e o update da lista listSGObject, linhas 12 a 14, que contém os objetos da tela.

Listagem 20 Adicionado métodos de atualização de SGObject em SGScreen

```
1     public void onUpdate() {}
2
3     @Override
4     protected void onDraw( Canvas canvas ) {
5
6         Paint paint = new Paint();
7         canvas.drawPaint( paint );
8         paint.setColor( Color.WHITE );
9         super.onDraw( canvas );
10
11        this.onUpdate();
12        for ( SGObject obj : this.listSGObject ) {
13            obj.update( canvas );
14        }
15
16    }
```

4.4.6 Exemplo de criação Componente usando SGObject

Como exemplo, será apresentado a criação do componente SGLabel, que será o componente que nos possibilitará de adicionar textos a tela do jogo.

Para iniciar, será criada uma classe e estendida a SGObject, como mostrado na Listagem 21.

Listagem 21 Inicio da criação do SGLabel

```
1 import br.com.lrmarkanjo.SGEngine.base.SGObject;
2
3 public class SGLabel extends SGObject {
4 }
```

Em seguida, como na Listagem 22, adiciona-se um atributo do tipo String para guardar o texto do objeto, linha 5. No exemplo, este foi chamado de text, sendo criado os métodos *getters* e *setters* desse texto, linhas 7 a 13.

Listagem 22 Adicionando atributo texto ao SGLabel

```
1 import br.com.lrmarkanjo.SGEngine.base.SGObject;
2
3 public class SGLabel extends SGObject {
4
5     private String text;
6
7     public String getText() {
8         return text;
9     }
10
11    public void setText(String text) {
12        this.text = text;
13    }
14
15 }
```

Agora será adicionado o controle de tamanho da fonte do texto, com os métodos setTextSize(float) para informar o tamanho que será utilizado no objeto Paint, e o método getTextSize() para recuperar este tamanho, como mostrado na Listagem 23, linhas 15 a 21. Note que não é necessário fazer os métodos para cor, pois eles já existem no SGObject.

Listagem 23 Métodos de controle do tamanho da fonte do SGLabel

```
1 import br.com.lrmarkanjo.SGEngine.base.SGObject;
2
3 public class SGLabel extends SGObject {
4
5     private String text;
6
7     public String getText() {
8         return text;
9     }
10
11    public void setText(String text) {
12        this.text = text;
13    }
14
15    public float getTextSize() {
16        return this.paint.getTextSize();
17    }
18
19    public void setTextSize( float textSize ) {
20        this.paint.setTextSize( textSize );
21    }
22
23 }
```

Assim a ultima modificação que será feita na classe é reescrever o método onUpdate() para desenhar o texto. Como se pode observar na Listagem 24, primeiro verifica-se a existência do texto e depois é usado o método drawText do Canvas, linha 5, para desenhar o texto na tela. Neste método é informado o texto a desenhar, a posição em x e y e por fim, o Paint que será usado para o desenho.

Listagem 24 Reescrevendo o método onUpdate no SGLabel

```
1     @Override
2     protected void onUpdate(Canvas canvas) {
3         if ( this.getText()!=null ) {
4             this.paint.setTextSize( this.getTextSize() );
5             canvas.drawText(
6                 this.getText(),
7                 this.xGlobal,
8                 this.yGlobal+this.height,
9                 this.paint
10            );
11        }
12    }
```

4.4.7 Finalizando e Testando o Framework

As classes criadas para o *framework* foram codificadas no projeto SGEngine (acrônimo em inglês para Motor para Jogos Simples) e esse projeto foi criado do tipo *Library*.

Uma *Library* é um conjunto de classes para um determinado fim. Não será um aplicativo executável, e sim, parte do código de um aplicativo. No NetBeans, para que um projeto seja do tipo *Library,* basta acessar as propriedades do projeto, selecionar a guia *Libraries* e selecionar o item *Is Library,* conforme apresentado na Figura 21.

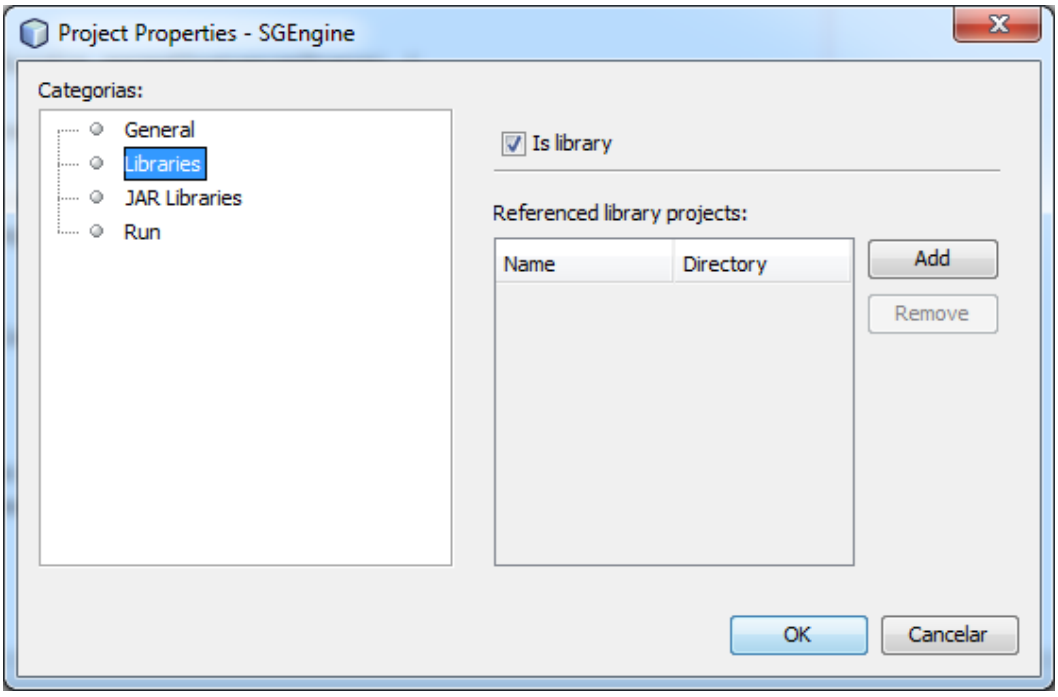

Figura 21 Projeto do tipo Library no NetBeans

Fonte: Autoria Própria.

Desta forma, o resultado do projeto é um arquivo do tipo *.jar,* podendo ser importado em outros projetos que tratam de criação de jogos. Um exemplo de *.jar* importado é apresentado na Figura 22.

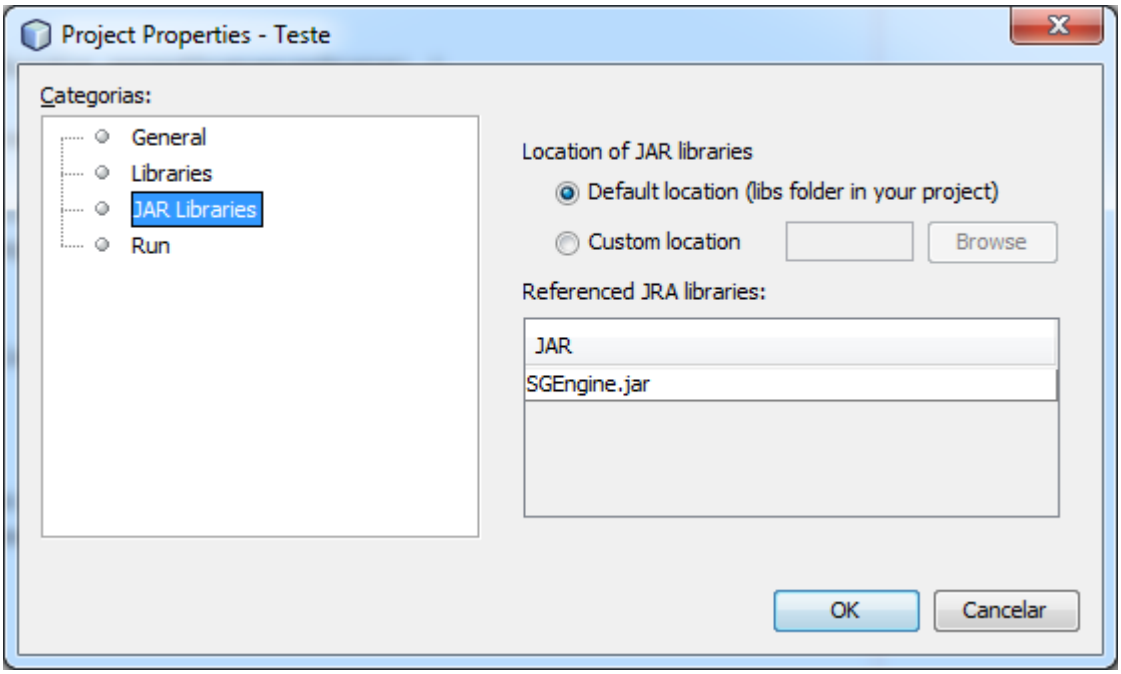

Figura 22 Exemplo de arquivo .jar importado em um projeto
Fonte: Autoria Própria.

Tendo o arquivo *.jar* do *framework* que foi nomeado de SGEngine.jar, pode-se criar outros projetos Android para utilizar este *.jar*. No NetBeans segue-se os seguintes passos:

Para iniciar a criação de um projeto basta acessar o *menu Arquivo* e selecionar a opção *Novo Projeto*... conforme Figura 23.

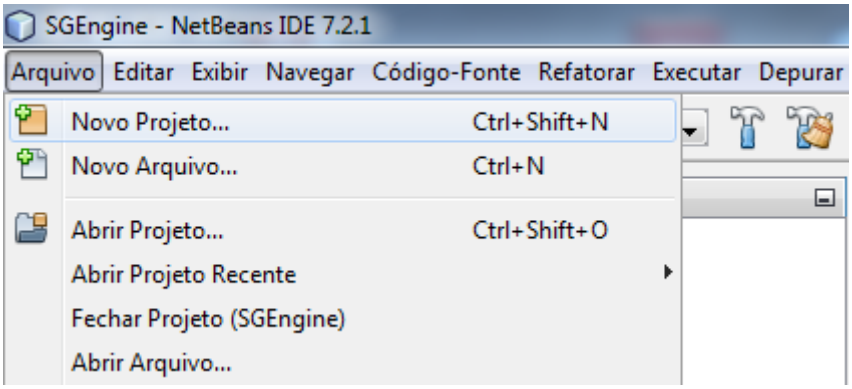

Figura 23 Caminho novo projeto NetBeans
Fonte: Autoria Própria.

Assim o NetBeans irá abrir a tela de Novo Projeto - Figura 24.

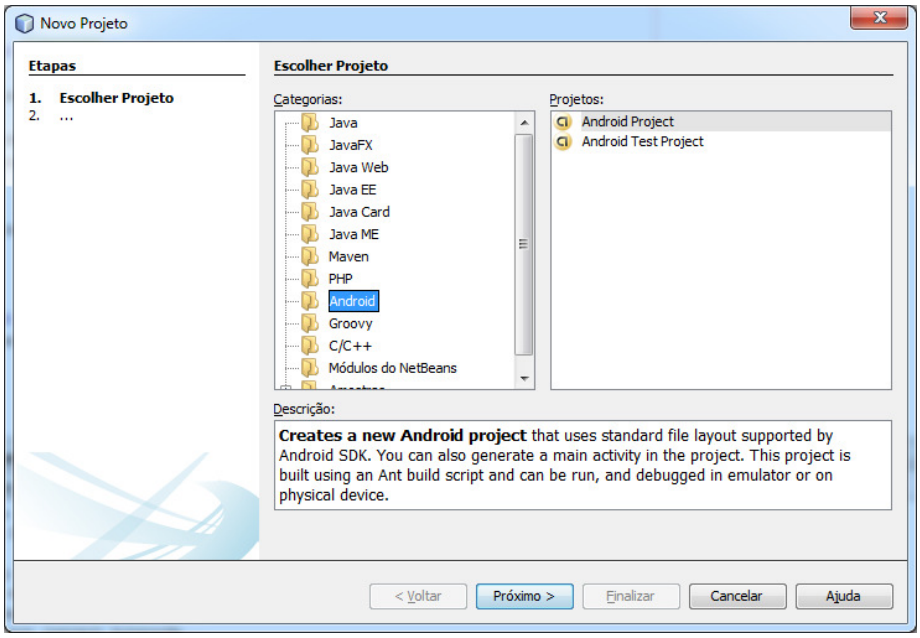

Figura 24 Tela de Novo Projeto no NetBeans

Fonte: Autoria Própria.

Nesta janela, na guia Categorias seleciona-se *Android*, na guia Projetos seleciona-se *Android Project* e acionando o botão *Próximo* o NetBens irá abrir a tela de *Novo Android Application*, conforme Figura 25.

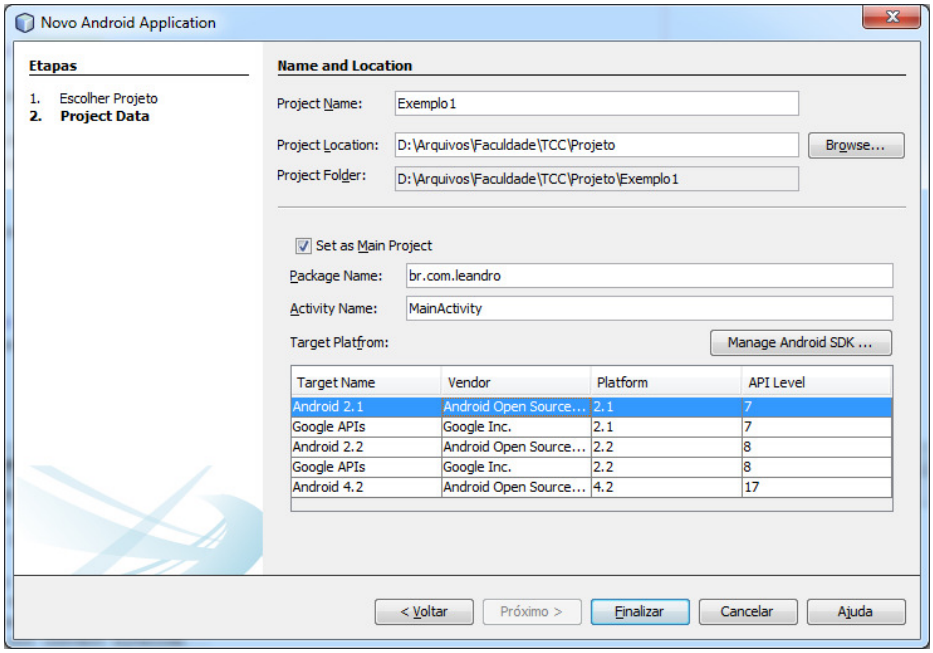

Figura 25 Tela Novo Android Application no Netbeans

Fonte: Autoria Própria.

Nesta janela apresentada, deve-se nomear o projeto, selecionar o caminho para o mesmo ser gravado, informar o nome do pacote da Activity, e por último seleciona-se a plataforma alvo de seu projeto. Clicando em *Finalizar*, o NetBeans irá criar o projeto Android como na Figura 26.

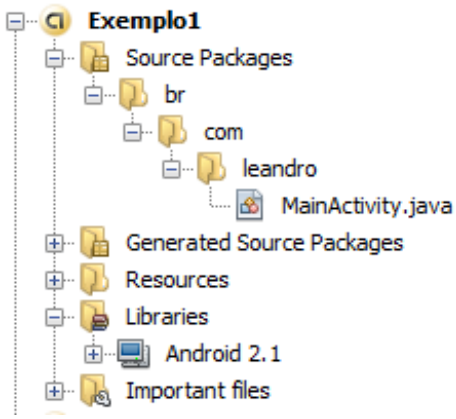

Figura 26 Estrutura de um projeto Android no Netbens
Fonte: Autoria Própria.

Para se adicionar o SGEngine.jar no projeto, basta adicionar o arquivo dentro da *Libraries* do projeto, conforme Figura 27.

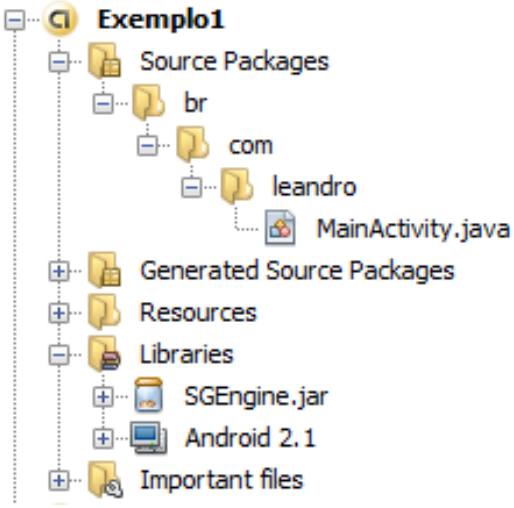

Figura 27 Projeto android com library SGEngine.jar adicionada
Fonte: Autoria Própria.

Pode-se verificar na Figura 27 que o NetBeans já cria automaticamente a Activity principal do projeto, de acordo com nome que é informado na criação do

projeto, neste caso como MainActivity.java. Um exemplo desta Activity foi criado na Listagem 9.

Neste MainActivity.java, Listagem 25, se deve apagar o método onCreate() e trocar o “extends Activity” para “extends SGActivity”, linha 5, neste momento será necessário implementar o método onInit(), linha 9, pois é um método abstrato na classe SGActivity.

Listagem 25 MainActivity com SGActivity

```
1  package br.com.leandro;
2
3  import br.com.lrmarkanjo.SGEngine.screen.SGActivity;
4
5  public class MainActivity extends SGActivity
6  {
7
8      @Override
9      protected void onInit() {
10         throw new UnsupportedOperationException("Not supported yet.");
11     }
12
13 }
```

Agora será criado uma SGScreen, clicando com o botão direito em um dos pacotes java do projeto, pode-se selecionar o menu *Novo* e então *Classe Java*. O Netbeans abrirá a janela *New Classe Java*, como pode ser visto na Figura 28.

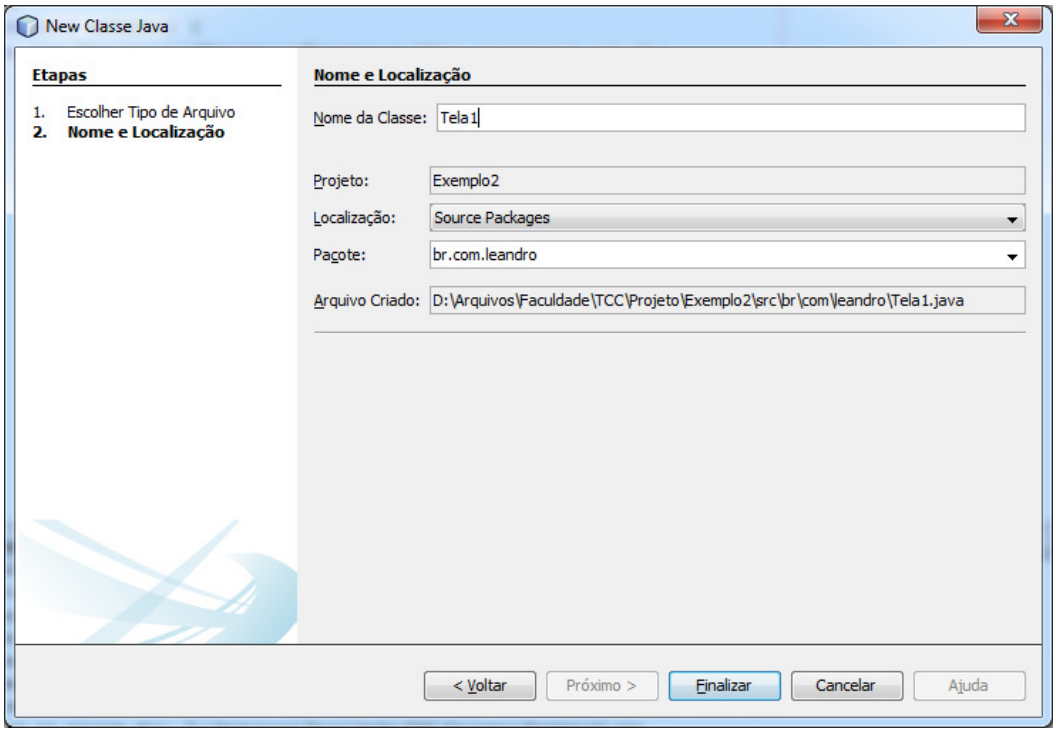

Figura 28 Janela New Classe java do Netbeans

Fonte: Autoria Própria.

Na janela *New Classe Java* da-se um nome a classe no campo *Nome da Classe* e pode-se alterar o pacote onde ela vai ficar no campo Pacote, clica-se em Finalizar para criar a classe.

Esta classe será estendida a SGScreen que automaticamente ira pedir para criar o método construtor informando o SGActivity como na Listagem 26.

Listagem 26 Classe exemplo SGScreen

```
1  package br.com.leandro;
2
3  import br.com.lrmarkanjo.SGEngine.screen.SGActivity;
4  import br.com.lrmarkanjo.SGEngine.screen.SGScreen;
5
6  public class Tela1 extends SGScreen {
7
8      public Tela1( SGActivity context ) {
9          super(context);
10     }
11
12 }
```

Com isso, pode-se adicionar uma SGLabel na tela, conforme Listagem 27. Foi criado um atributo do tipo SGLabel, linha 10, selecionando sua cor na linha 16, tamanho na linha 17, conteúdo em texto na linha 18, posição nas linhas 20 e 21 e adicionado a tela na linha 23.

Listagem 27 Exemplo de código para SGLabel

```
1  package br.com.leandro;
2
3  import android.graphics.Color;
4  import br.com.lrmarkanjo.SGEngine.object.SGLabel;
5  import br.com.lrmarkanjo.SGEngine.screen.SGActivity;
6  import br.com.lrmarkanjo.SGEngine.screen.SGScreen;
7
8  public class Tela1 extends SGScreen {
9
10     private SGLabel label;
11
12     public Tela1( SGActivity context ) {
13         super(context);
14
15         this.label = new SGLabel();
16         this.label.setColor( Color.BLACK );
17         this.label.setTextSize( 20 );
18         this.label.setText( "Teste de SGLabel" );
19
20         this.label.x = 50;
21         this.label.y = 50;
22
23         this.addSGObject( this.label );
24
25     }
26
27 }
```

Para finalizar, basta abrir a tela criada na SGActivity, com o método openScreen(SGScreen), conforme Listagem 28, linha 11, e executar o projeto que o resultado será a tela apresenta na Figura 29.

Listagem 28 Abrindo exemplo de SGCreen no SGActivity

```
1  package br.com.leandro;
2
3  import br.com.lrmarkanjo.SGEngine.screen.SGActivity;
4
5  public class MainActivity extends SGActivity
6  {
7
8      @Override
9      protected void onInit() {
10
11         this.openScreen( new Tela1( this ) );
12
13     }
14
15 }
```

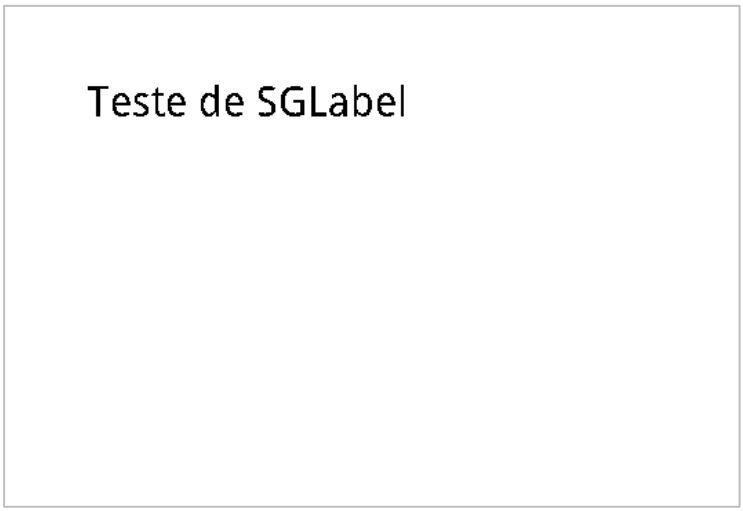

Figura 29 Exemplo de SGLabel em tela

Fonte: Autoria Própria.

4.4.8 Gerar documentação do *framework*

Para gerar a documentação do *framework* foi utilizado o JavaDoc, de acordo com ORACLE (2013) o JavaDoc é uma ferramenta que analisa as declarações e comentários de documentação em um conjunto de arquivos de origem e produz um conjunto de páginas HTML, descrevendo as classes, interfaces, construtores, métodos e campos.

Para que o JavaDoc reconheça a documentação basta criar um comentário acima dos métodos desejados usando “/**” para iniciar o texto e “*/” para finalizar o texto para aquele método, pode-se notar esse procedimento no exemplo da Listagem 29.

Listagem 29 Exemplo de comentário para JavaDoc

```
1  /**
2   * Esta classe é responsável por carregar e desenhar
3   * os objetos em tela e controlar o looping da aplicação.
4   * @author Leandro
5   */
6  public class SGScreen extends View implements Runnable {
7  }
```

Após a finalização dos comentários para os métodos, utilizando o NetBeans, basta acessar o menu *Executar* e selecionar o opção *Gerar Javadoc* que será criado a documentação do projeto.

Desta forma foi criado a documentação do *framework* como pode-se verificar na Figura 30.

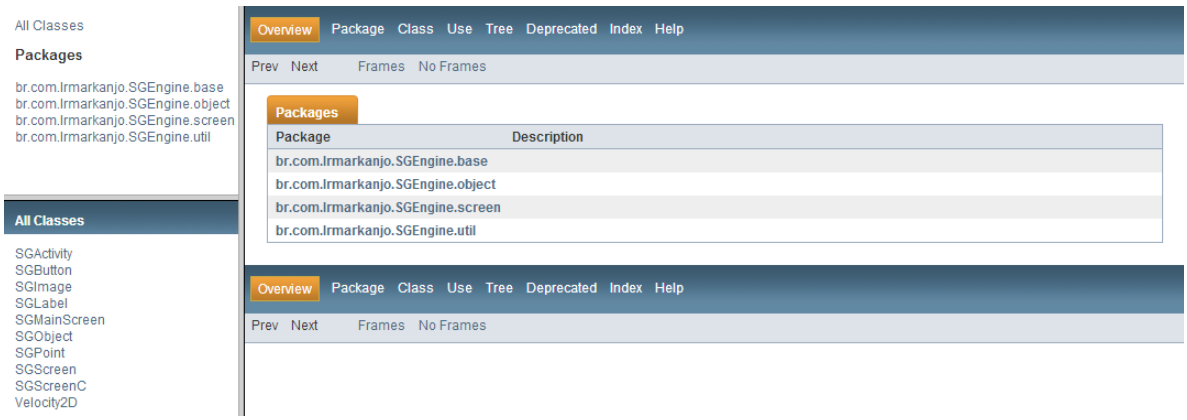

Figura 30 Documentação do *framework*

Fonte: Autoria Própria.

4.5 DESENVOLVIMENTO DO JOGO

Um problema no desenvolvimento de jogos para Android é que seus dispositivos podem ter vários tamanhos de telas, assim, deve-se trabalhar com tamanhos relativos ao tamanho da tela do dispositivo. Para ajudar na padronização dos componentes de tela, pode-se criar uma tela para carregar informações e dados

principais para o jogo, essa tela em geral é simples, como na Figura 31, pois não tem outra finalidade além de carregar dados gerais para o jogo.

Figura 31 Tela FrmCarregando

Fonte: Autoria Própria.

O tamanho da tela pode ser acessado com os métodos getWidth() e getHeight() do Canvas, como apresentado na Listagem 30. Estes dados podem ser guardados em uma classe estática para que possa ser acessada de outras telas.

Listagem 30 Acessando tamanho da tela.

```
1     @Override
2     protected void onUpdate() {
3
4         SGScreenC.width = this.canvas.getWidth();
5         SGScreenC.height = this.canvas.getHeight();
6
7     }
```

Nesta tela, também pode-se carregar algumas imagens gerais para o jogo, como por exemplo a imagem de fundo das telas. Para carregar essas imagens, foi utilizado a classe SGImage, criado no próprio *framework*, como na Listagem 31.

.

Listagem 31 Carregar imagem com SGImage

```
1 this.imgCarregar = new SGImage(getResources(),R.drawable.espaco_inicial);
2 this.imgCarregar.scale( SGScreenC.width, SGScreenC.height );
3 Consts.imgFundoPrincipal = this.imgCarregar.getBitmap();
```

Esta classe possui o métodos construtores public SGImage(Resources, int), o qual carrega uma imagem recebendo um android.content.res.Resources, que de

acordo com ANDROID DEVELOPERS – RESOURCES (2013) é responsável pelo acesso a arquivos e recursos do aplicativo, (esses arquivos ficam na pasta *Resourses*, como mostrado na Figura 32) e também o seu indice. No método scale(int, int), é escalonada a imagem ao tamanho desejado, em largura e altura respectivamente.

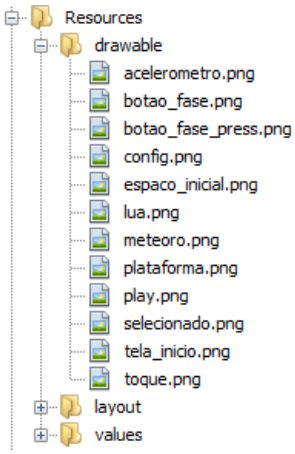

Figura 32 Pasta Resourses no Netbeans

Fonte: Autoria Própria.

Após carregar informações e imagens necessárias, é carregada a tela inicial do jogo, conforme Figura 33.

Figura 33 Tela FrmInicial

Fonte: Autoria Própria.

Na Listagem 32 pode-se observar como os objetos criados no jogo não possum posição fixa, mas sim uma posição que é referente ao tamanho da tela. Os

metodos setLeft(float) e setRight(float), linhas 6 e 10, posicionam o objeto afastado da borda esquerda ou direita da tela, respectivamente, de acordo com a porcentagem informada (a porcentagem é um valor entre 0 e 1, sendo 0.3 equivalente a 30% da tela).

Listagem 32 Objetos posicionados em relação ao tamanho da tela.

```
1     this.imgLogojogo.x = SGScreenC.width/4;
2     this.imgLogojogo.y = (SGScreenC.height/10);
3
4     this.btnIniciar.y = SGScreenC.height/1.5f;
5     this.btnIniciar.width = this.btnIniciar.height = SGScreenC.height/4;
6     this.btnIniciar.setLeft( 0.3f );
7
8     this.btnConfig.y = SGScreenC.height/1.5f;
9     this.btnConfig.width = this.btnConfig.height = SGScreenC.height/4;
10    this.btnConfig.setRight( 0.3f );
```

Na tela inicial, se acionado o botão da engrenagem, o jogo irá abrir a tela de configurações, como apresentado na Figura 34, onde pode-se escolher a forma de controlar o jogo (a esquerda, toque na tela, a direita usando o movimento do *smartphone*).

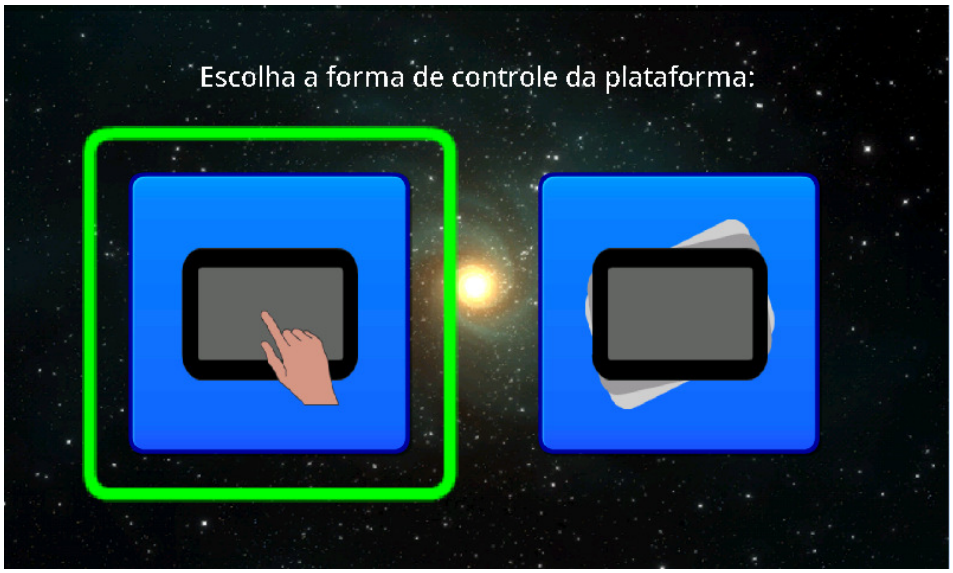

Figura 34 Tela FrmOpcoes
Fonte: Autoria Própria.

Ainda na Tela Inicial, acionando o botão a esquerda, o jogo irá abrir a tela de escolha de grupo de fases, estes persistidos no SQLite, conforme Figura 35. Os grupos de fases estão na tabela GrupoFase, que foi criada com a classe GrupoFaseDAO, conforme Listagem 32.

Figura 35 Tela FrmEsGrupo

Fonte: Autoria Própria.

Listagem 33 Classe GrupoFaseDAO

```
1  import android.database.sqlite.SQLiteDatabase;
2  import android.database.sqlite.SQLiteOpenHelper;
3  import br.com.lrmarkanjo.SGEngine.screen.SGScreenC;
4
5  public class GrupoFaseDAO extends SQLiteOpenHelper {
6
7      public static final String TABELA = "GrupoFase";
8      public static final int VERSION = 1;
9      public static final String[] COLUNAS = { "id", "nome" };
10
11     public GrupoFaseDAO() {
12         super( SGScreenC.context, TABELA, null, VERSION );
13     }
14
15     @Override
16     public void onCreate( SQLiteDatabase db ) {
17         String sql = "CREATE TABLE " + TABELA + "( "
18                 + "id INTEGER PRIMARY KEY, "
19                 + "nome TEXT "
20                 + ");";
21         db.execSQL( sql );
22     }
23
24     @Override
25     public void onUpgrade( SQLiteDatabase db,
26                         int oldVersion, int newVersion ) {
27         if ( oldVersion!=newVersion ) {
28             db.execSQL( "DROP TABLE IF EXISTS " + TABELA );
29             this.onCreate( db );
30         }
31     }
32
33 }
```

Um grupo de fase é composto por nove fases, cada grupo de fase terá um tema ou carateristica diferente, como por exemplo a imagem de fundo será diferente, ou um novo elemento que muda a jogabilidade sera apresentado neste grupo. Esses grupos estarão informados na tabela GrupoFase.

Com a classe GrupoFaseDAO criada, pode-se acessar os dados da tabela GrupoFase, como na Listagem 34, utilizando o cursor para recuperar as informações. De acordo com ANDROID DEVELOPERS – CURSOR (2013), o cursor fornece o acesso de leitura e escrita para um conjunto de resultados de uma consulta de banco de dados, com ele pode-se percorrer os dados da consulta a tabela e criar uma lista de itens.

Listagem 34 Código para buscar lista de GrupoFase

```
1     this.dao = new GrupoFaseDAO();
2
3     Cursor c = this.dao.getReadableDatabase()
4                     .query( GrupoFaseDAO.TABELA,
5                             GrupoFaseDAO.COLUNAS,
6                             null, null, null, null, "id, nome" );
7     List<GrupoFase> lista = new ArrayList<GrupoFase>();
8     while( c.moveToNext() ) {
9
10        GrupoFase a = new GrupoFase();
11        a.setId( c.getInt(0) );
12        a.setNome( c.getString(1) );
13
14        lista.add( a );
15
16    }
17    c.close();
18    this.dao.close();
19    return lista;
```

Na tela de escolha de grupos, após escolher um grupo de fases, o jogo apresentará a tela de Escolha de Fase, de acordo com o grupo escolhido como na Figura 36.

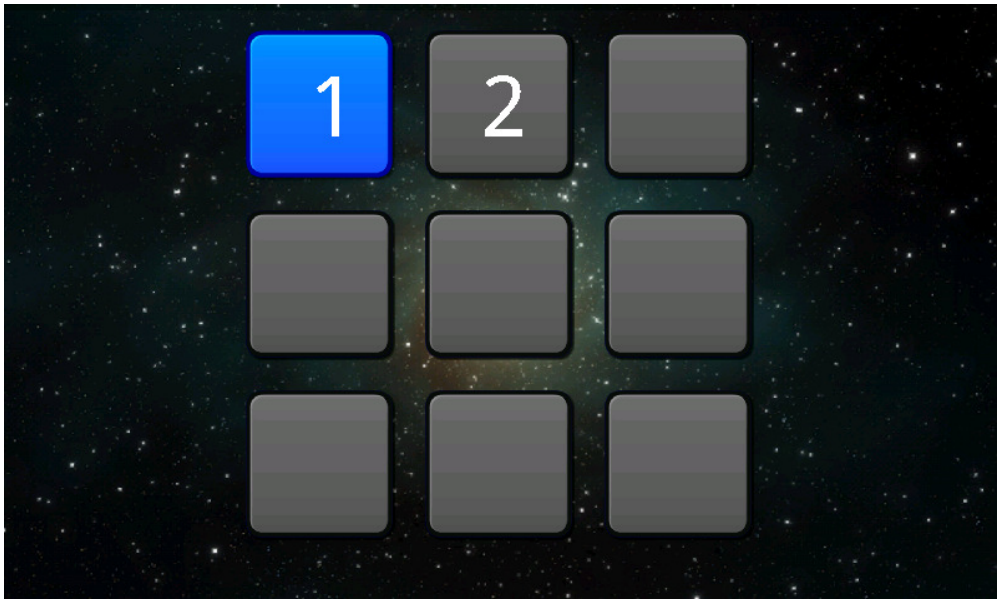

Figura 36 Tela FrmEsFase

Fonte: Autoria Própria.

Na tela de Escolha de Fase, se nenhuma fase foi concluída apenas a primeira fase estará ativa para jogar, após cada fase concluída, a proxíma fase é ativada para jogar.

Selecionando uma fase, o jogo ira abrir a Tela da Fase, como na Figura 37 onde o jogador irá controlar a Plataforma para não deixar cair o Círculo e tentar destruir todos os Destroços da tela.

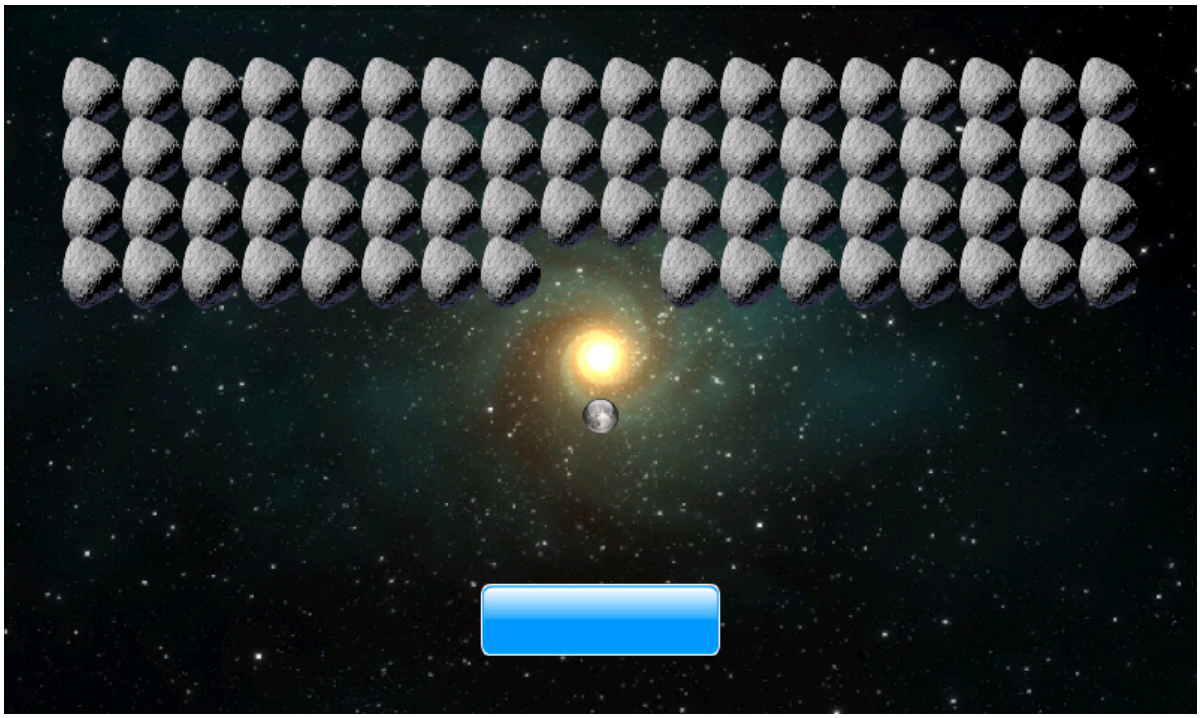

Figura 37 Tela FrmFase
Fonte: Autoria Própria.

4.6 PUBLICANDO JOGO DO GOOGLE PLAY

Após o aplicativo codificado e testado, o próximo passo é publicá-lo no Google Play, mas antes é necessário gerar uma versão final do aplicativo no formato *.apk*, sendo este o arquivo disponibilizado na plataforma.

4.6.1 Gerar versão final de um Aplicativo Android

No Netbeans, para gerar uma versão final do jogo basta acessar o menu *Ferramentas* na barra de superior e clicar na opção *Export Signed Android Package* como mostrado na Figura 38.

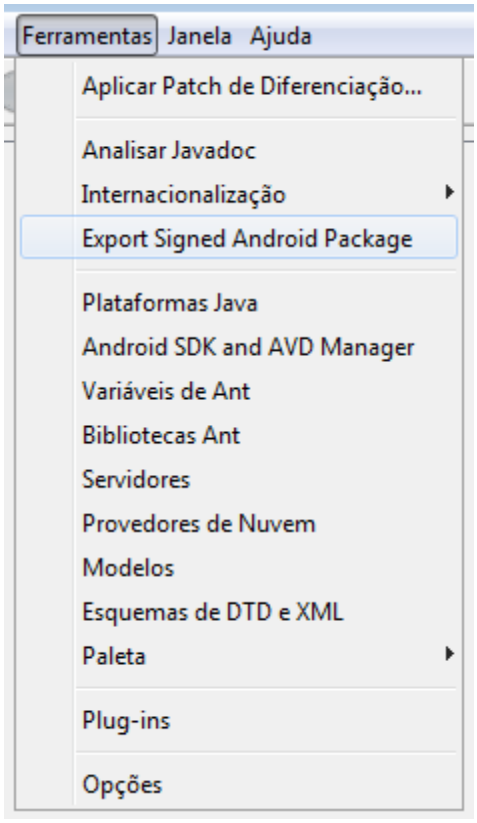

Figura 38 Menu Ferramentas do Netbeans
Fonte: Autoria Própria.

Com isso, a janela da Figura 39 será aberta. Nessa janela pode-se usar ou criar uma *keystore*, onde serão guardadas as chaves utilizadas nas aplicações Android. Essa chave serve como segurança para a publicação no Google Play, uma vez publicado com uma chave só se pode atualizar a aplicação com um pacote *.apk* se ele tiver a mesma chave. Para criá-la basta selecionar um caminho com nome do arquivo e digitar uma senha para acesso a esse arquivo.

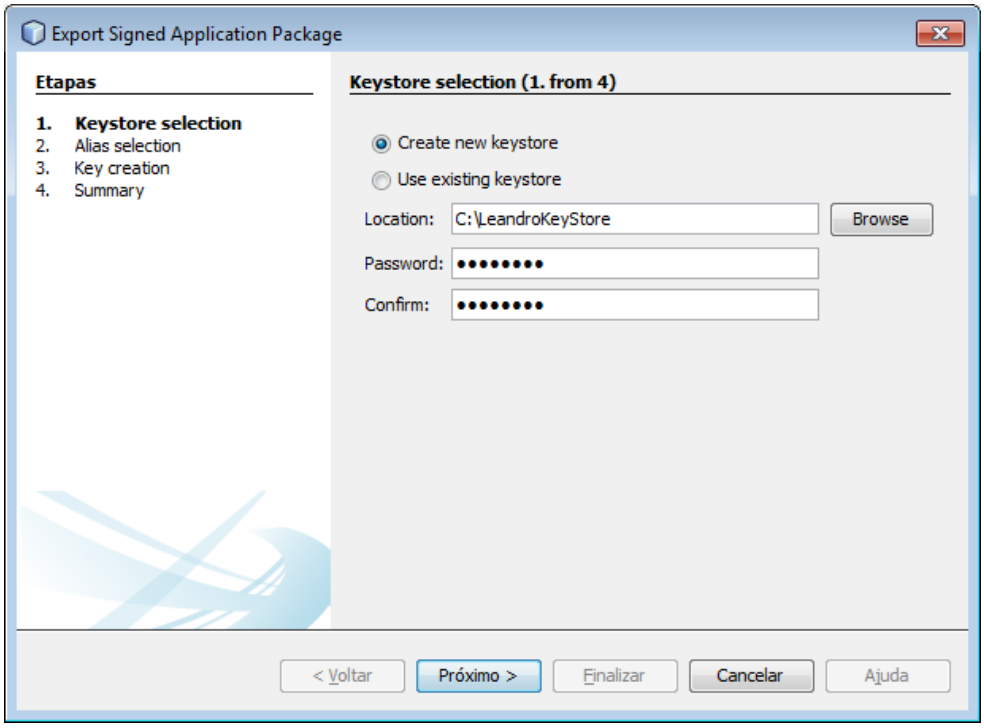

Figura 39 Export Signed Application Package passo 1

Fonte: Autoria Própria

Clicando em *Próximo,* a janela da Figura 40 será aberta, onde se seleciona a chave ou cria-se uma nova chave para utilizar na exportação.

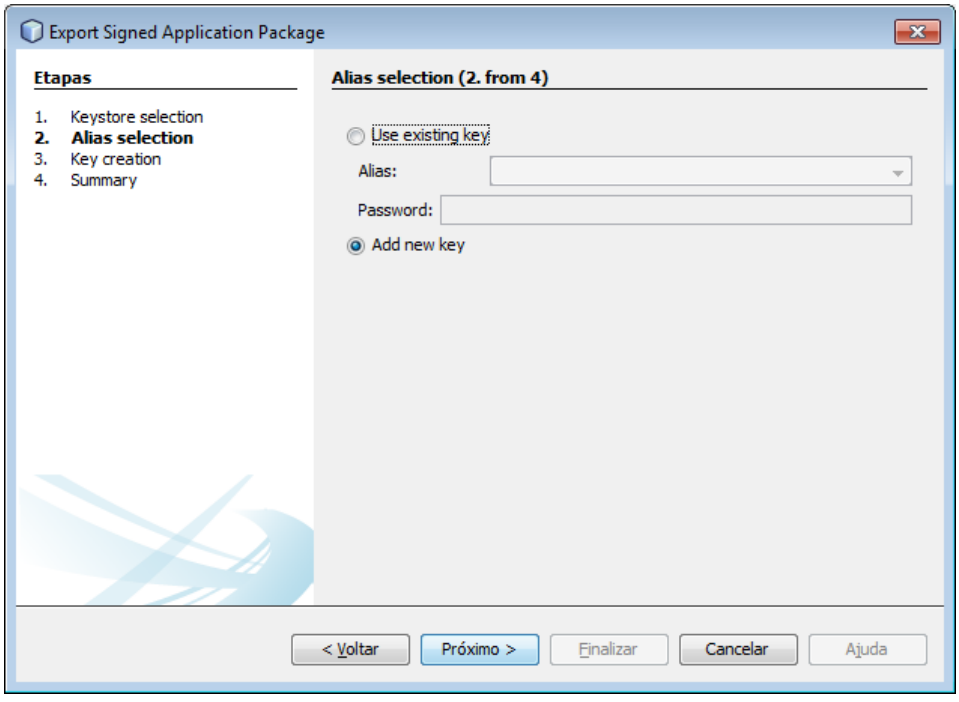

Figura 40 Export Signed Application Package passo 2

Fonte: Autoria Própria

Se a opção escolhida for *Use existing key* ao clicar em *Próximo* a janela da Figura 42 será aberta, agora se a opção foi *Add new key* a janela da Figura 41 é que será aberta.

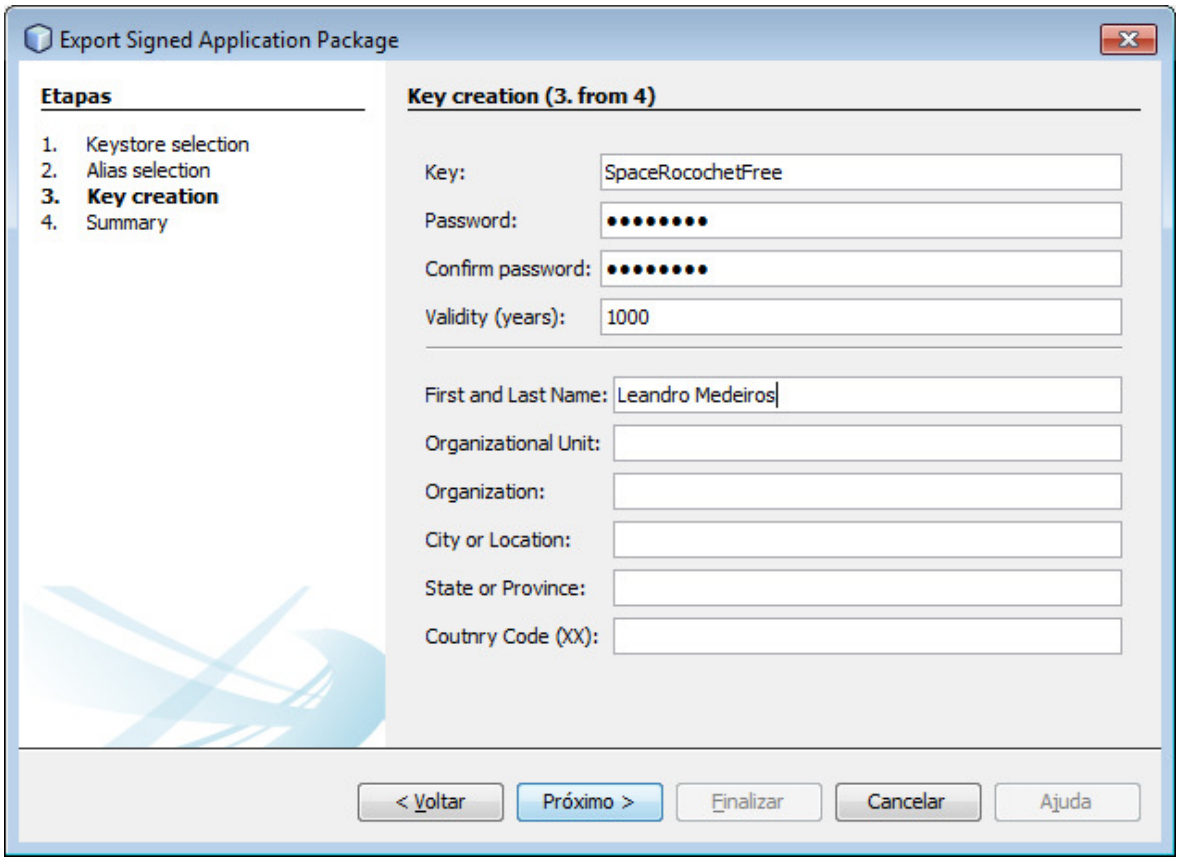

Figura 41 Export Signed Application Package passo 3

Fonte: Autoria Própria

Nesta janela informa-se um nome para a chave, senha, validade em anos e nome do usuário/desenvolvedor. A validade determina quanto tempo o aplicativo poderá ser atualizado pelos seus desenvolvedores.

Clicando em *Próximo, a* janela da Figura 42 será aberta onde apresentará o caminho onde o arquivo *.apk* da aplicação será exportado e o tempo da chave escolhida, clicando em Finalizar o arquivo será gerado.

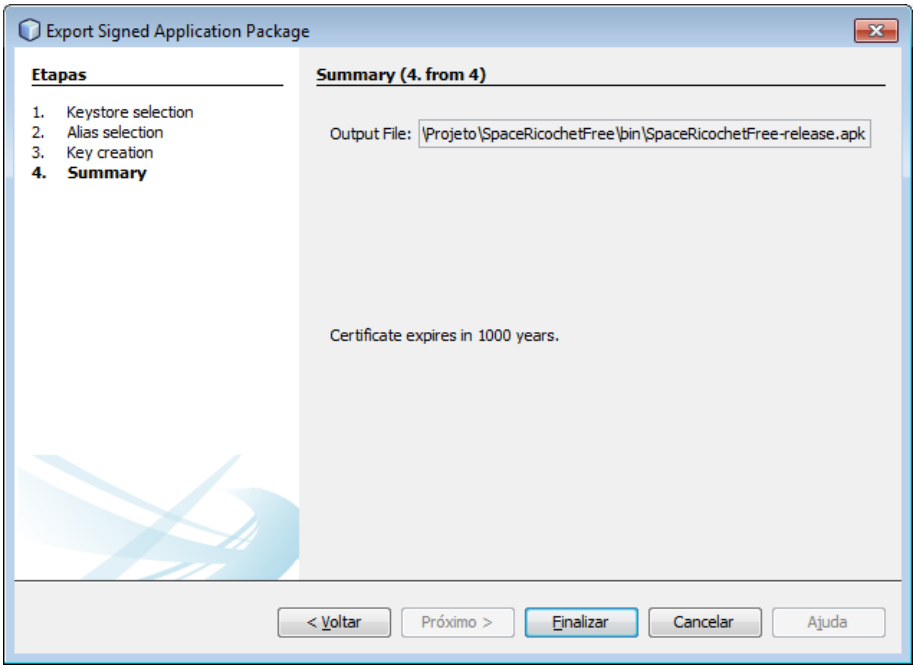

Figura 42 Export Signed Application Package passo 4

Fonte: Autoria Própria

4.6.2 Postando aplicativo no Google Play

Para postar um aplicativo no Google Play, deve-se acessar o portal GOOGLE PLAY DEVELOPER CONSOLE (2013), caso não tenha uma conta no Google, basta seguir o guia para criação de conta, como na Figura 43.

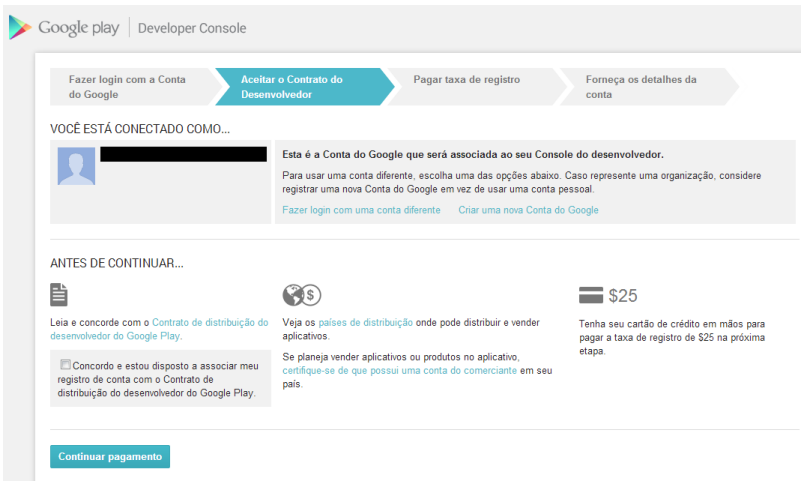

Figura 43 Página para criar conta no Google Play

Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Tendo uma conta, ao acessá-la, a página da Figura 44 será exibida. Esta é a pagina inicial, onde serão apresentados todos os aplicativos da conta.

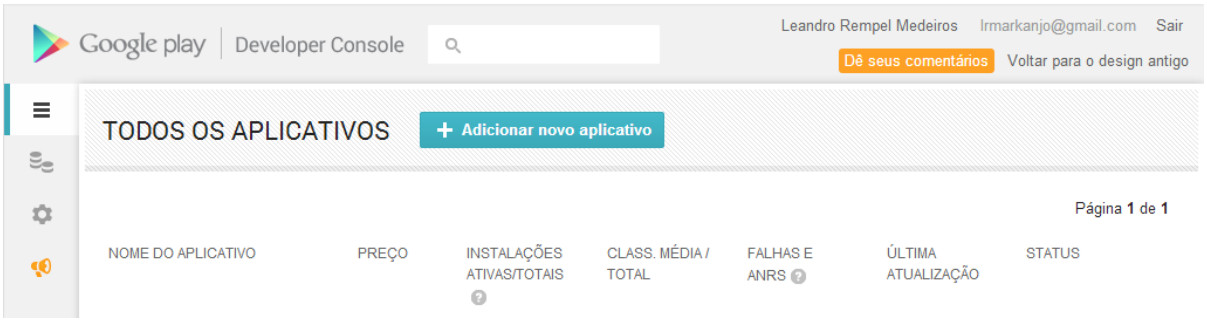

Figura 44 Página inicial do Google Play Developer Console
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Para adicionar um aplicativo, nesta página, clica-se em *Adicionar novo aplicativo* e a página da Figura 45 será exibida.

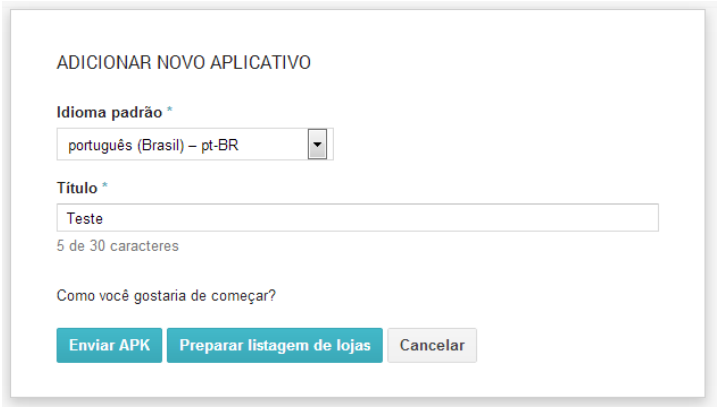

Figura 45 Google Play Adiconar novo Aplicativo
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Nesta página, informe o idioma e o título do aplicativo. Na página existem duas opções: *Preparar a listagem de lojas* que abrirá a página da Figura 49 ou enviar o arquivo APK que abrirá a página da Figura 46. Ambas as opções terão que ser realizada. Em *Preparar listagem de lojas* será informado dados gerais do aplicativo, como título, descrição, imagens de demonstração. Em *Enviar APK* será a opção para enviar o arquivo *.apk* criado na seção anterior.

Uma vez tendo o arquivo *.apk* pode-se escolher a opção *Enviar APK*.

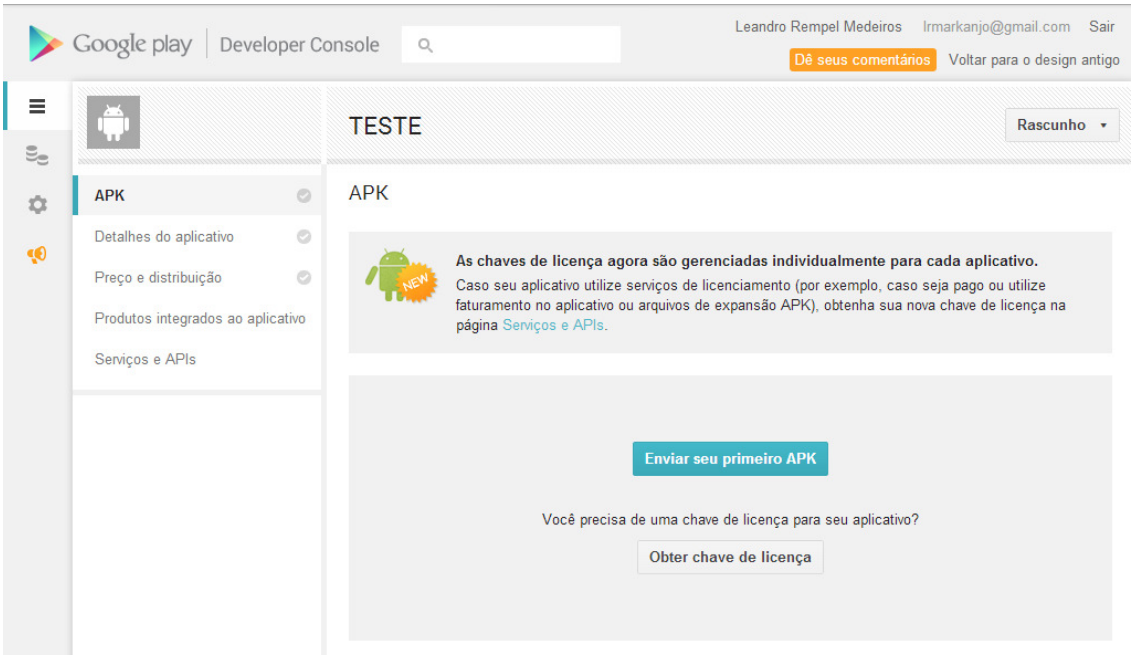

Figura 46 Google Play Página do Aplicativo
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Nesta página, para enviar o arquivo *.apk,* seleciona a opção *Enviar seu primeiro APK* e a página da Figura 47 sera exibida.

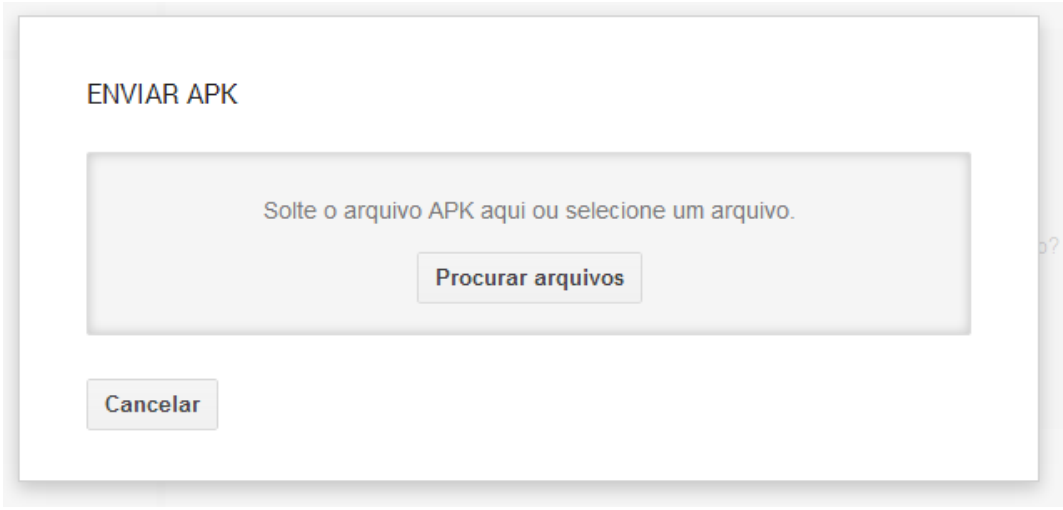

Figura 47 Google Play Enviar APK
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Nesta página, basta clicar em *Procurar arquivos* e selecionar o arquivo *.apk* que foi gerados na seção 4.6.1. O arquivo será importado e a página da Figura 48 será exibida.

Figura 48 Google Play Informações do APK enviado
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Nesta página é apresentado algumas informações sobre o arquivo importado como nome do pacote, versão, data de importação, etc.

No menu a esquerda da página tem mais dois passos obrigatórios para que a seja possível publicar o aplicativo: *Detalhes do aplicativo* e *Preço e distribuição*.

Em Detalhes do aplicativo (Figura 49) será informado/alterado título, descrição, imagens de demonstração, ícone de apresentação, etc. Após terminar deve-se clicar em Salvar.

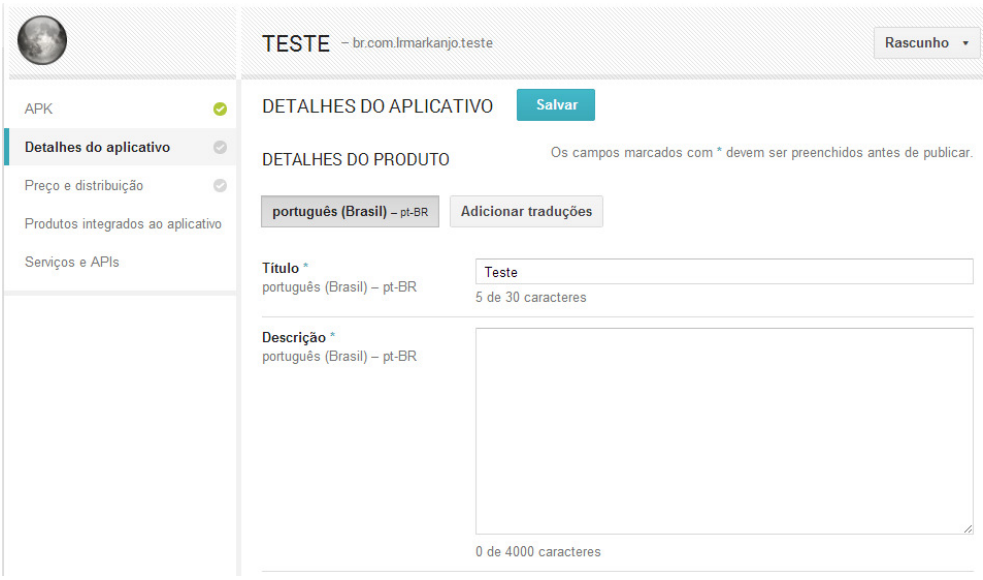

Figura 49 Google Play Detalhes do aplicativo
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Em Preço e distribuição (Figura 50), será informado o preço e países de distribuição e outras autorizações. Após terminar deve-se clicar em Salvar.

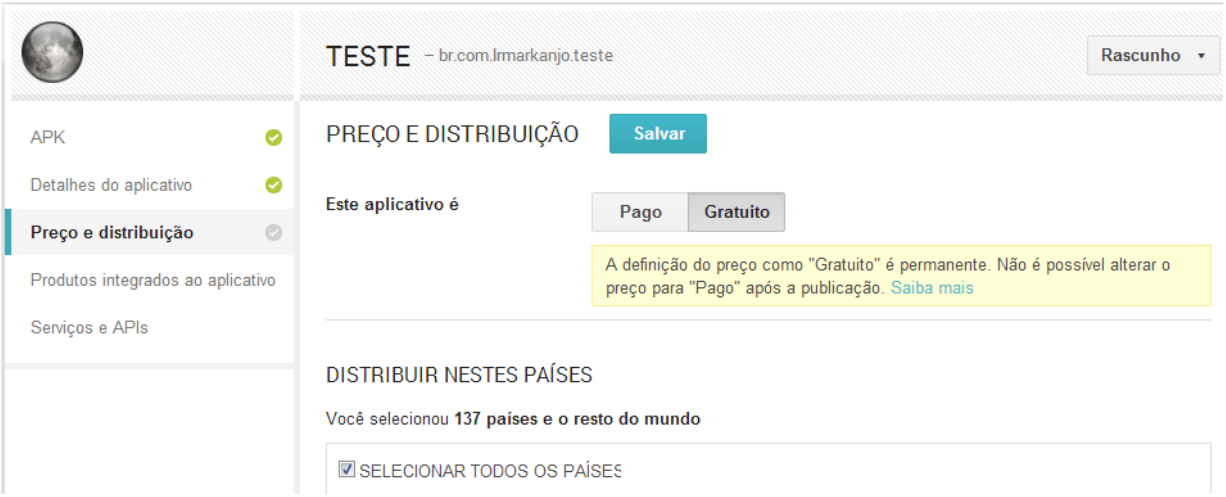

Figura 50 Google Play Preço e Distribuição
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Após informar todos os dados a opção de publicar ficará ativa no topo superior direito da página, como na Figura 51.

Figura 51 Google Play Opção pronto para publicar
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Caso esta opção não fique ativa, basta clicar em *Rascunho* e selecionar a opção *Por que não posso publicar?* – Figura 52. Uma página com as informações que faltam para habilitar a publicação será exibida como na Figura 53.

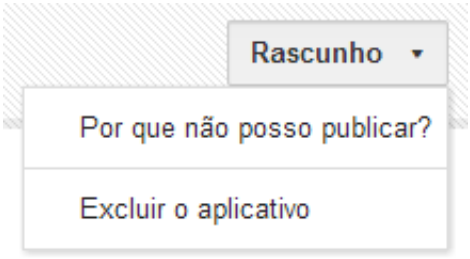

Figura 52 Google Play Opção pronto para publicar desativada
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Figura 53 Google Play Porque não posso publicar?
Fonte: GOOGLE PLAY DEVELOPER CONSOLE (2013)

Após publicar o aplicativo, ele irá ser apresentado no Google Play para que possa ser comprado/baixado, como na Figura 54.

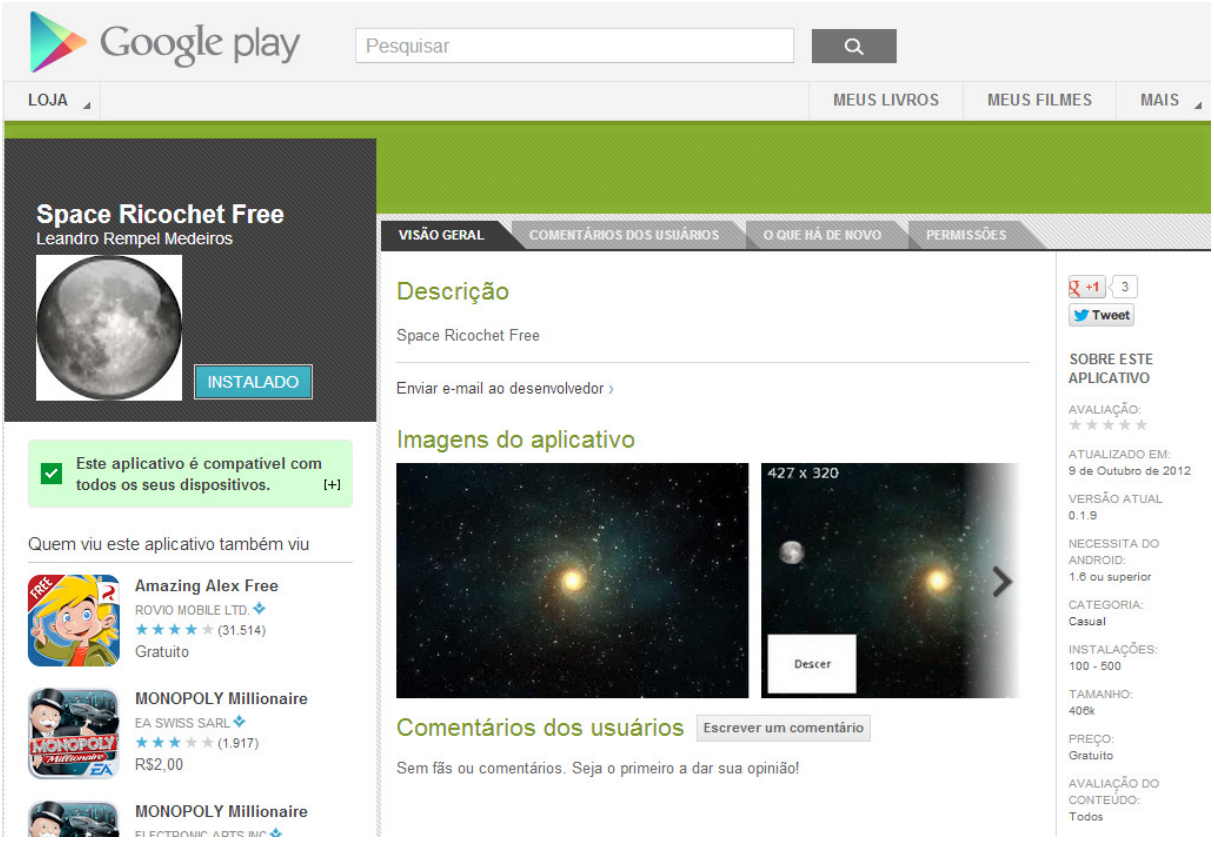

Figura 54 Jogo Space Ricochet publicado no Google Play
Fonte: GOOGLE PLAY (2013)

4.7 Testes

Após publicação, o jogo foi sujeito a alguns testes tanto em dispositivos pequenos como em *tablets*, onde foi constatado que o jogo funcionou perfeitamente em todos os casos. Nas figuras 54 e 55 pode-se verificar o jogo em dispositivos Android diferentes.

Figura 55 Jogo em um Sony Ericsson Xperia Neo V

Fonte: Autoria Própria

Figura 56 Jogo em um Sansung Galaxy 5

Fonte: Autoria Própria

5 CONCLUSÃO

Este trabalho teve como objetivo desenvolver um jogo para Android utilizando tela sensível ao toque e acelerômetro, sendo também desenvolvido um *framework* para ajudar no desenvolvimento do jogo. Após codificado, o jogo foi publicado no Google Play para distribuição.

O estudo desenvolvido mostra que a utilização da tela sensível ao toque e do recurso de acelerômetro é relativamente simples, não ocorrendo erros ou problemas na sua utilização. A utilização das ferramentas de desenvolvimento Netbeans, NBAndroid e Android SDK atenderam as expectativas, pois, mesmo com ferramentas livres (sem custo), foi possível montar um ambiente de desenvolvimento que possibilitou o desenvolvimento do jogo.

O desenvolvimento do *framework* proporcionou a separação em dois projetos (*framework* e jogo), assim facilitando o desenvolvimento e a futura manutenção do jogo, pois, as classes de estrutura e componentes de tela ficaram no *framework* e as classes responsáveis pelo controle do jogo ficaram no projeto do jogo, e estando em um projeto separado o *framework* pode ser reutilizado para o desenvolvimento de outros jogos.

O desenvolvimento do jogo acabou sendo simples, pois pontos mais complexos como controle de objetos em tela e métodos de desenho ficaram no desenvolvimento *framework*. Com a análise de tabelas e fluxo de telas realizada, ficou simples criar as telas e desenvolver cada funcionalidade, pois já estava tudo planejado. O SQLite que foi utilizado controla toda a parte de criação e alteração de tabelas, não sendo necessário desenvolver um processo para isso.

A distribuição pelo Google Play é facilitada para desenvolvedores independentes pela não necessidade de ter um CNPJ para distribuir seus aplicativos. Assim, qualquer desenvolvedor pode desenvolver seus aplicativos e distribuir pela plataforma, basta criar sua conta no Google Play, e não terá mais nenhum custo pelo serviço, e a própria plataforma gerencia a venda/download do aplicativo.

Ao final deste trabalho conclui-se que o desenvolvimento de jogos para Android com o auxílio de um *framework* torna-se fácil porque em geral os jogos para esta plataforma são simples, exatamente para passar como um passa tempo. Mas o

desenvolvimento do próprio *framework* não é simples porque acaba sendo um caminho trabalhoso, um exemplo desta questão é o trabalho com imagens. Uma propriedade muito comum utilizada para criar *skins* de botões é o *9-slice-scaling* que, em resumo, é dividir uma imagem em nove pedaços, onde os pedaços referentes aos quatro cantos de uma imagem não são alterados quando esta imagem é redimensionada. Esta propriedade da imagem foi desenvolvida no *framework*, mas foi de longe, a que levou maior tempo de desenvolvimento.

A utilização de recursos do Android, como tela sensível ao toque e acelerômetro, tornam o jogo mais atrativo, uma comprovação disto é que o jogo já teve mais de cem *downloads*. Mas, apesar da documentação sobre as classes referentes ao recurso do acelerômetro, a utilização da rotação do dispositivo no eixo z não é possível utilizando apenas o sensor de acelerômetro, por isso, foi resolvido no trabalho utilizar apenas a rotação no eixo y.

Como trabalho futuro registra-se a necessidade de incrementar o *framework* para tratar sons. Também poderá ser melhorado o *design* das telas e objeto do jogo, adicionado mais grupos de fases e consequentemente mais fases, adicionar pontuação nas fases com intuito de motivação ao refazer as fases e por fim adicionada uma forma do usuário criar suas próprias fases e disponibilizar para outras pessoas jogarem.

REFERÊNCIAS BIBLIOGRAFICAS

ABI RESEARCH. 2011**. Android Overtakes Apple with 44% Worldwide Share of Mobile App Downloads.** Artigo publicado no portal ABIresearch, disponível em: <http://www.abiresearch.com/press/android-overtakes-apple-with-44-worldwide-share-of>. Acessado em 20 de abril de 2013.

ANDROID DEVELOPERS – ACTIVITY. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/app/Activity.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – BUNDLE. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/os/Bundle.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – CANVAS. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/graphics/Canvas.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – CURSOR. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/database/Cursor.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – MOTIONEVENT. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/view/MotionEvent.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – PAINT. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/graphics/Paint.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – RESOURCES. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/content/res/Resources.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – RUNNABLE. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/java/lang/Runnable.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – SENSOR. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/hardware/Sensor.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – SENSOREVENT. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/hardware/SensorEvent.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – SENSOREVENTLISTENER. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/hardware/SensorEventListener.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – SENSORMANAGER. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/hardware/SensorManager.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – SQLITEOPENHELPER. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – THREAD. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/java/lang/Thread.html>. Acessado em: 12 de Março de 2013.

ANDROID DEVELOPERS – VIEW. 2013. Portal: Android Developers disponível em: <http://developer.android.com/reference/android/view/View.html>. Acessado em: 12 de Março de 2013.

ANDROID SDK. 2013**.** Portal: Developers Android disponível em: <http://developer.android.com/sdk/index.html>. Acessado em: 25 de Fevereiro de 2013.

CARVALHO, FELIPE MONTEIRO DE. 2011. **Programação Android**. Revista ClubeDelphi edição 128

CASASANTA, LEONARDO VIANA. 2012, **Acelerômetro no Android.** Portal: Android on Board disponível em: <http://androiddevbr.wordpress.com/2012/11/25/ acelerometro-no-android/>. Acessado em: 22 de Janeiro de 2013.

FRUIT NINJA. 2013**.** Portal: Google Play disponível em: <https://play.google.com/store/apps/details?id=com.halfbrick.fruitninja>. Acessado em: 25 de Fevereiro de 2013.

FRUIT NINJA FREE. 2013**.** Portal: Google Play disponível em: <https://play.google.com/store/apps/details?id=com.halfbrick.fruitninjafree>. Acessado em: 25 de Fevereiro de 2013.

GEEKAPHONE. 2011**. Mobile Gaming is Dominating the Gaming Industry.** Artigo publicado no portal geekaphone, disponível em: <http://geekaphone.com/blog/mobile-games-by-the-numbers/>. Acessado em 20 de abril de 2013.

GOOGLE MERCHANT CENTER. 2013. **Google Checkout**. Disponível em: <http://support.google.com/merchants/bin/answer.py?hl=pt-BR&answer=188480>. Acessado em: 25 de Fevereiro de 2013.

GOOGLE PLAY . 2013a. Portal: Google disponível em: <https://play.google.com>. Acessado em: 25 de Fevereiro de 2013.

GOOGLE PLAY. 2013b. **Contrato de serviços ao desenvolvedor**. Disponível em: <https://play.google.com/intl/ALL_br/about/developer-distribution-agreement.html>. Acessado em: 25 de Fevereiro de 2013.

GOOGLE PLAY. 2013c. **Taxa de transações**. Disponível em: <http://support.google.com/googleplay/android-developer/answer/112622?hl=pt-BR>. Acessado em: 25 de Fevereiro de 2013.

GOOGLE PLAY DEVELOPER CONSOLE. 2013. Portal: Google Play disponível em: <https://play.google.com/apps/publish/v2/signup/>. Acessado em: 25 de Fevereiro de 2013.

HAMMERSCHMIDT, ROBERTO. 2008, **O que é Touch Screen?** Portal: TecTudo disponível em: <http://www.tecmundo.com.br/multitouch/177-o-que-e-touch-screen-.htm>. Acessado em: 22 de Janeiro de 2013.

HUNTER, William. 2000, **The History of Video Games, from 'pong' to 'pac-man'.** Designboom.com. disponível em: <http://www.designboom.com/eng/education/pong.html>. Acessado em: 18 de julho de 2012.

IDC. 2010**. IDC Forecasts Worldwide Mobile Applications Revenues to Experience More Than 60% Compound Annual Growth Through 2014.** Artigo publicado no portal IDC, disponível em: <http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22617910&secti>. Acessado em 20 de abril de 2013.

KENAI. 2013**.** Portal: Project Kenai disponível em: <http://kenai.com/projects/nbandroid>. Acessado em: 25 de Fevereiro de 2013.

LANDIM, WIKERSON. 2011, **O tamanho da indústria dos vídeo games[infografo].** Portal: TecMundo disponível em: <http://www.tecmundo.com.br/infografico/9708-o-tamanho-da-industria-dos-video-games-infografico-.htm>. Acessado em: 22 de Janeiro de 2013.

LECHETA, Ricardo R. **Google Android : aprenda a criar aplicações para dispositivos móveis com o Android SDK**. – 2. ed. ver. e ampl. – São Paulo : Novatec Editora, 2010.

MACORATTI, JOSÉ CARLOS. 2011**, Apresentando o padrão DAO - Data Access Object.** Macoratti.net Disponível em: <http://www.macoratti.net/11/10/pp_dao1.htm>. Acessado em: 05 de Abril de 2013.

MICHAELIS. **Dicionário prático da língua portuguesa**. 1. ed. São Paulo: Editora Melhoramentos, 2009. 502 p.

MORIMOTO, CARLOS E. 2010, **Entendendo o Android.** Hardware.com.br Disponível em: <http://www.hardware.com.br/tutoriais/android/>. Acessado em: 23 de julho de 2012.

NASCIMENTO, ALEXANDRE. 2012, **O que é acelerômetro? Como funciona?** Disponível em: <http://www.appleboy.com.br/blog/o-que-e-acelerometro-como-funciona>. Acessado em: 21 de abril de 2013.

NASCIMENTO, R. M. 2010, **5 tendências que irão revolucionar o mundo mobile.** Disponível em: http://www.mobilepedia.com.br/noticias/5-tendencias-que-irao-revolucionar-o-mundo-mobile. Acessado em: 27 de maio de 2013.

NBANDROID. 2013**.** Portal: Project NBAndroid disponível em: <http://www.nbandroid.org/p/about.html>. Acessado em: 25 de Fevereiro de 2013.

NEED FOR SPEED MOST WANTED. 2013**.** Portal: Google Play disponível em: <https://play.google.com/store/apps/details?id=com.ea.games.nfs13_row>. Acessado em: 25 de Fevereiro de 2013.

NEED FOR SPEED SHIFT. 2013**.** Portal: Google Play disponível em: <https://play.google.com/store/apps/details?id=com.eamobile.nfsshift_row_wf>. Acessado em: 25 de Fevereiro de 2013.

NETBEANS. 2012. **Notas da Release do NetBeans IDE 7.2.** Artigo publicado no portal NetBeans, disponível em: <https://netbeans.org/community/releases/72/relnotes_pt_BR.html>. Acessado em 20 de abril de 2013.

NETBEANS. 2013. **A Brief History of NetBeans.** Artigo publicado no portal NetBeans, disponível em: <http://netbeans.org/about/history.html>. Acessado em 14 de janeiro de 2013.

ORACLE. 2013. **Javadoc Tecnology.** Artigo publicado no portal Oracle, disponível em: <http://docs.oracle.com/javase/7/docs/technotes/guides/javadoc/>. Acessado em 22 de abril de 2013.

SILVA, RAFAEL. 2011, **Desebvolvimento Android: Recursos da Plataforma.** DevMedia.com Disponível em: <http://www.devmedia.com.br/desenvolvimento-com-android-recursos-da-plataforma-android/22240>. Acessado em: 03 de outubro de 2012.

SQLITE. 2013**.** Portal: SQLite disponível em: <http://www.sqlite.org/>. Acessado em: 25 de Fevereiro de 2013.

STANEK, ROMAN. 2009, **Biography** Portal: Sys.Con disponível em: <http://romanstanek.sys-con.com/>. Acessado em: 26 de Março de 2013.

PEREZ, S. 2012, **The Number of Mobile Device will Exceed World's Population by 2012.** TechRunch.com. disponível em: <http://techcrunch.com/2012/02/14/the-number-of-mobile-devices>. Acessado em: 14 de maio de 2012.

VECTORNAV. 2013. **Accelerometer.** Artigo publicado no portal VectorNav Technologies, disponível em: <http://www.vectornav.com/support/library?id=84>. Acessado em 21 de abril de 2013.

WASHINGTONPOST. 2012. **Touchscreens: How they work.** Artigo publicado no portal The Washington Post, disponível em: <http://www.washingtonpost.com/national/health-science/touchscreens-how-they-work/2012/01/23/gIQAZHIAMQ_graphic.html>. Acessado em 21 de abril de 2013.

APÊNDICE A – Instalação do Android SDK

Nesta página da Figura 12, clicando em *Download the SDK,* será apresentada uma página de download como na Figura 57, onde se deve selecionar o *check* para aceitar os termos de uso, escolher a versão do sistema operacional (32-bit ou 64-bit) e clicar em *Download the SDK*. Neste ponto o download irá iniciar.

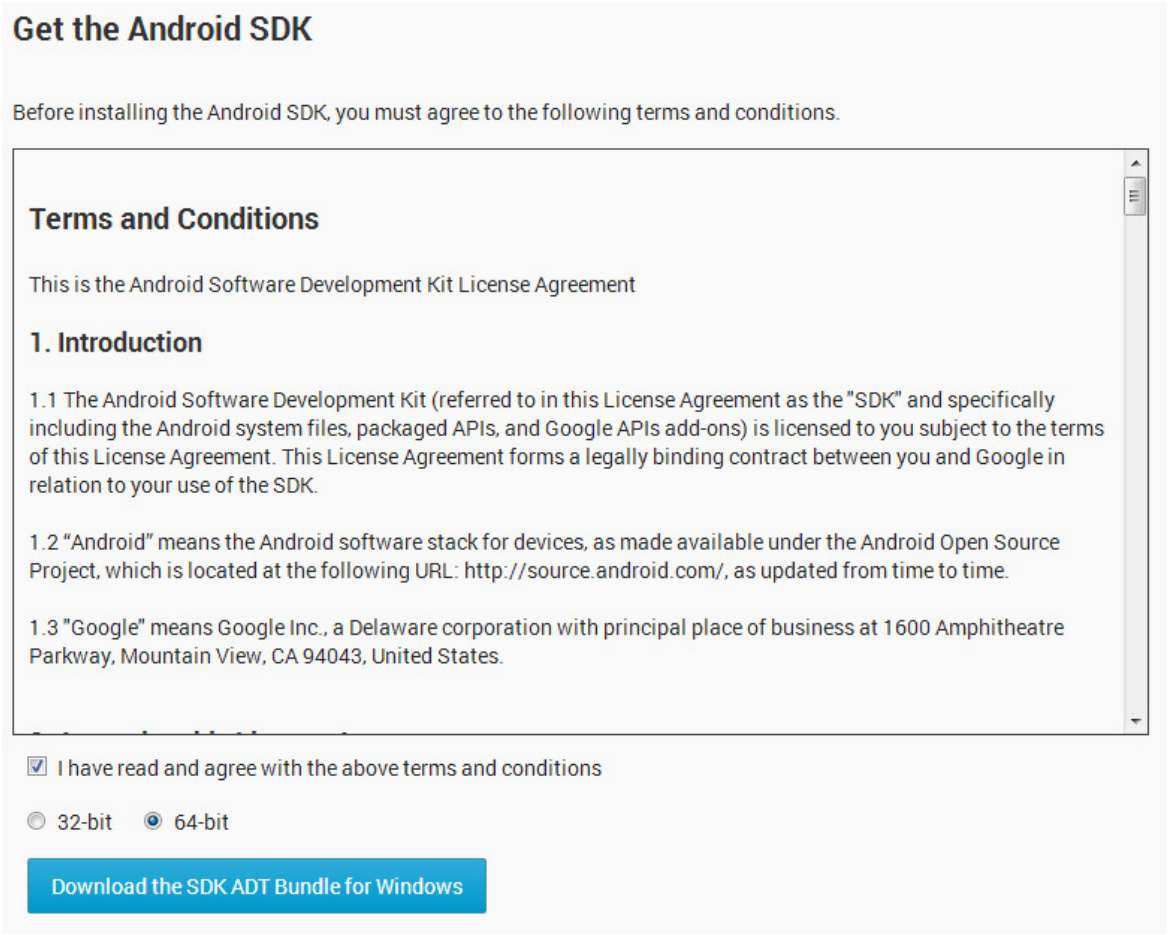

Figura 57 Android SDK download

Fonte: ANDROID SDK (2013)

Após o download, deve-se descompactar a pasta e executar o arquivo SDK Manager.exe, este irá abrir o aplicativo Android SDK Manager, conforme Figura 58.

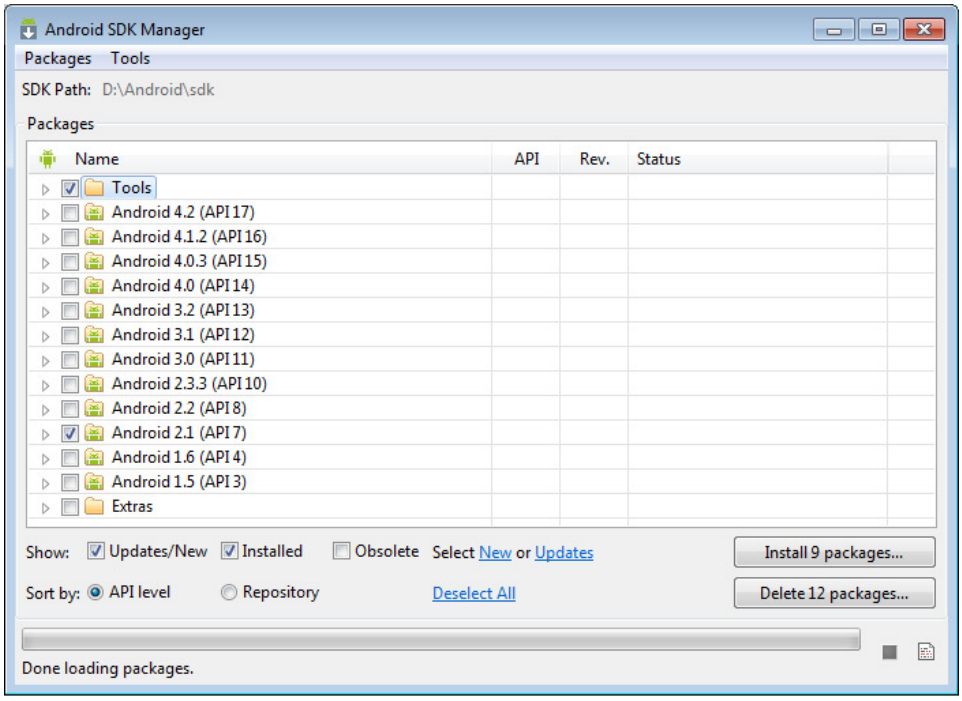

Figura 58 Android SDK Manager

Fonte: Autoria Própria

No Android SDK Manager, deve-se escolher os pacotes necessários para o desenvolvimento, o pacote *tools* traz as ferramentas para fazer testes com o emulador de dispositivo Android e os outros pacotes trazem as versões de Android. No caso do projeto deste documento, é usado a API 7, esta marcada na Figura 58. Depois de escolhido os pacotes, seleciona-se *Install X Packages...* (onde X será o número de pacotes que será feito download) e irá abrir a janela *Choose Packages to Install*, conforme Figura 59.

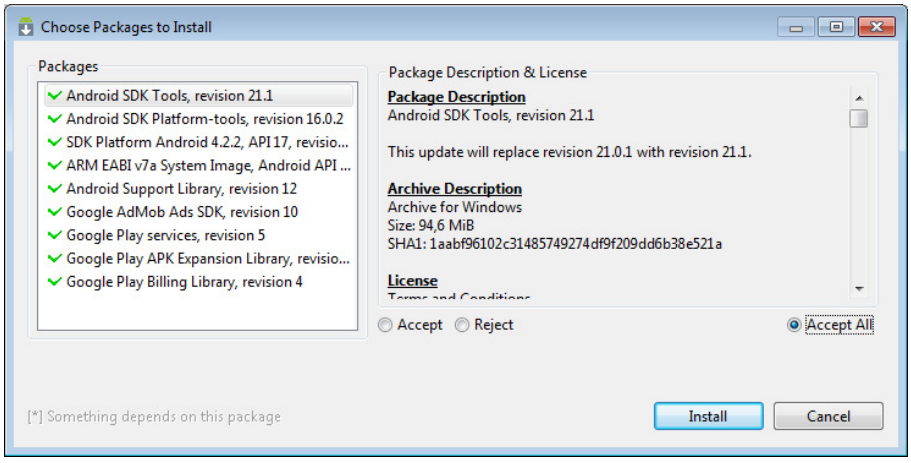

Figura 59 Android SDK Manager - download pacotes

Fonte: Autoria Própria

Nesta janela, será selecionado e aceitado as licenças para cada pacote ou selecionado *Accept All* para aceitar todas e acionado botão *Install* para iniciar o download dos pacotes.

Após finalizar o download, acessando o *menu Tools* e selecionando a opção *Manage AVDs...* a janela *Android Virtual Device Manager* será aberta como na Figura 60. Nesta janela pode-se adicionar uma AVD (Android Virtual Device – Dispositivo Virtual Android).

AVD será a plataforma de testes dos aplicativos, um emulador de dispositivos Android, simulando todas as características necessárias pelo desenvolvedor, por exemplo, pode-se criar uma AVD com o sistema operacional Android 2.1, com recursos de tela sensível a toque, além de definir o tamanho da tela, quantidade de memória, entre outras características.

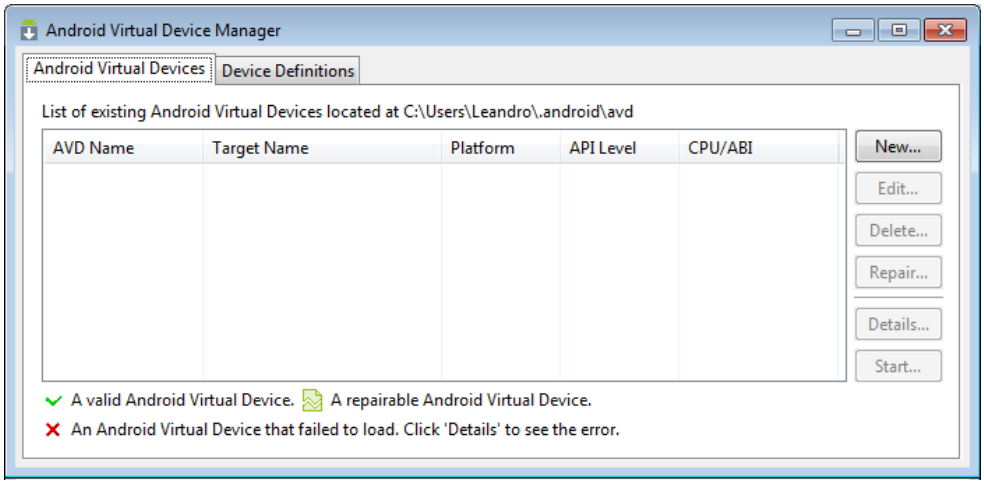

Figura 60 Android SDK Manager - Android Virtual Device

Fonte: Autoria Própria

Para criar uma AVD aciona-se o botão *New...* e será aberto a janela *Create new Android Virtual Device (AVD)* como na Figura 61, nesta tela pode-se configurar como será o dispositivo emulado, o mais importante neste caso é o campo *Target,* pois nele se escolhe a versão do Android que será emulado, no caso deste projeto será o API 7 para dispositivos com Android 2.1.

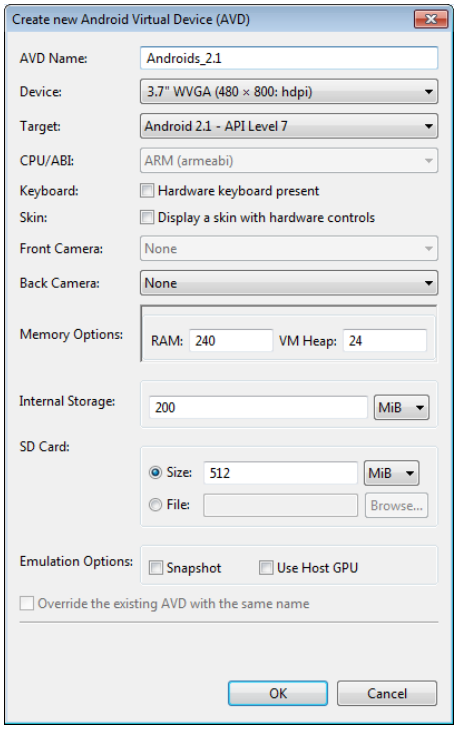

Figura 61 Android SDK Manager - Criando AVD

Fonte: Autoria Própria

Acionando o botão *OK* nesta tela, o AVD será criado como na Figura 62.

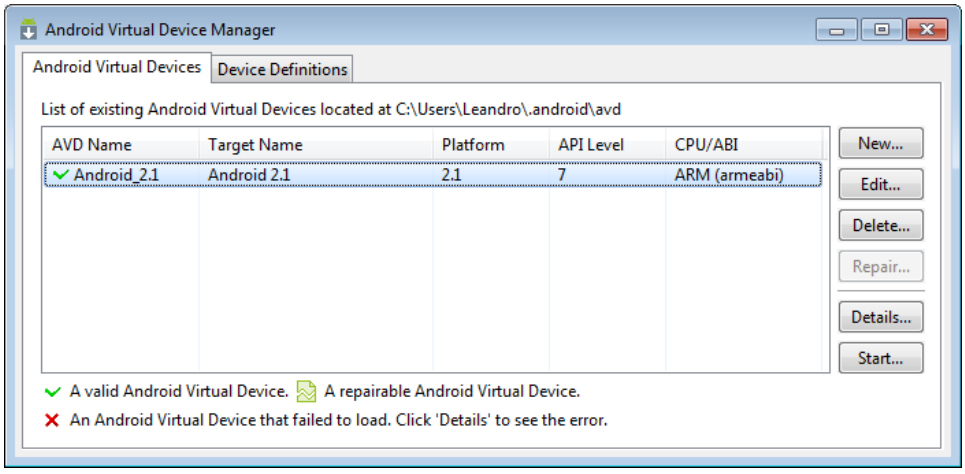

Figura 62 Android SDK Manager - Android Virtual Device com AVD criada

Fonte: Autoria Própria

Para iniciar a AVD deve-se selecionar a AVD e acionar o botão *Start...*, assim abrindo a janela *Launch Options* como na Figura 63.

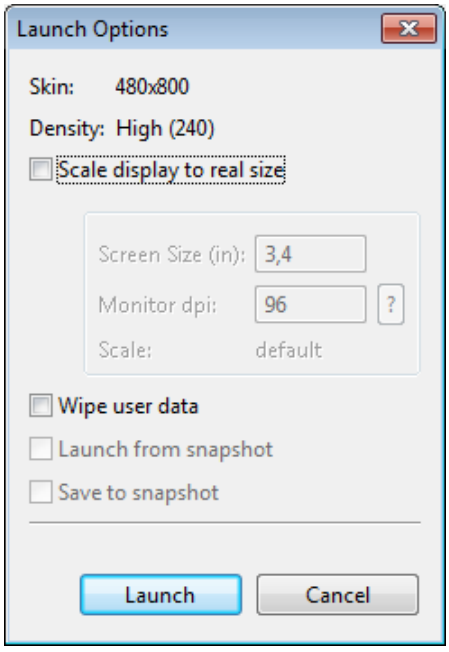

Figura 63 Android SDK Manager - Lauch Options

Fonte: Autoria Própria

Nesta tela acionando o botão *Launch*, a AVD pode ser iniciada, o que apresenta uma tela com o ambiente Android pronto para ser utilizado, conforme Figura 64.

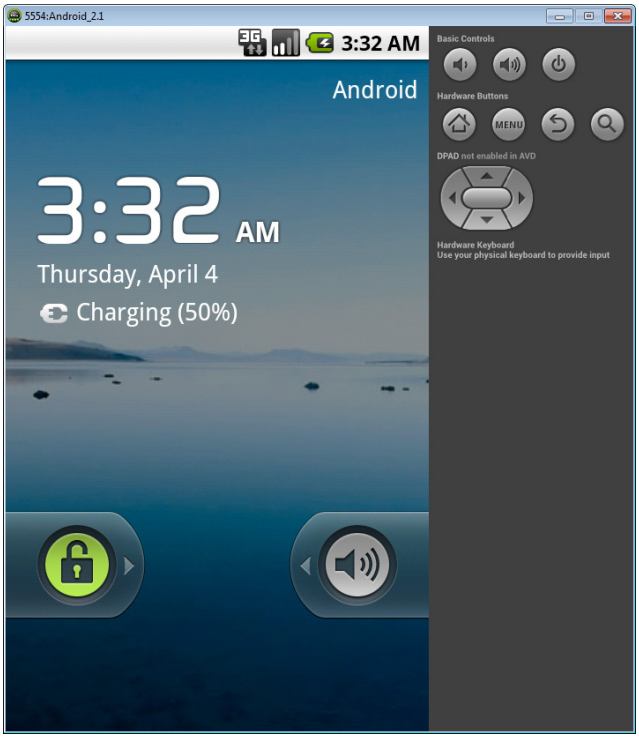

Figura 64 AVD com Android 2.1

Fonte: Autoria Própria

APÊNDICE B – Instalação do NBAndroid

Para instalar o NBAndroid, basta acessar o menu *Ferramentas* do NetBeans e selecionar a opção *Plug-ins*. Na tela apresentada, seleciona-se a aba *Definições* como na Figura 65.

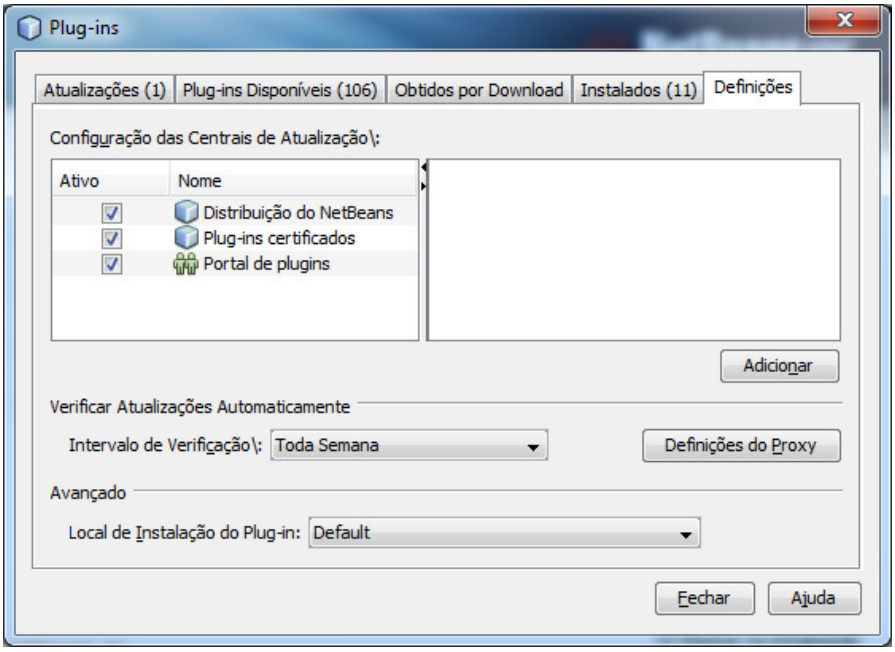

Figura 65 Netbeans - *Plug-ins* - Definições

Fonte: Autoria Própria

Clicando no botão *Adicionar* é apresentada a *Central de Atualização,* como na Figura 66.

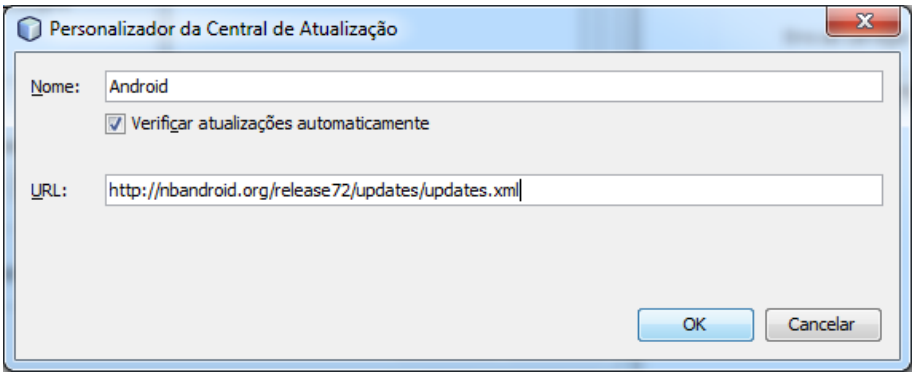

Figura 66 Netbeans - Personalizador da Central de Atualização

Fonte: Autoria Própria

Na tela da Figura 66, deve-se informar a URL do *plug-in* e marcar a opção *Verificar atualizações automaticamente,* para que o NetBeans encontre atualizações do NBAndroid. E por fim aciona-se o botão *OK* para instalar o *plu-gin*.

Após, na aba *Plug-ins Disponíveis* pode-se selecionar a atualização do Android como na Figura 67.

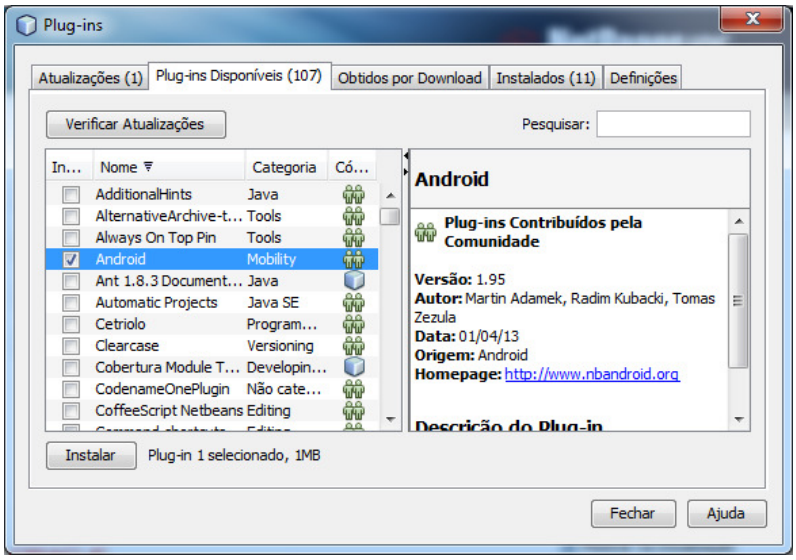

Figura 67 Netbeans - *Plug-ins* - Disponíveis

Fonte: Autoria Própria

Ao clicar no botão *Instalar* a instalação do NBAndroid, é iniciada a instalação, conforme apresentada na Figura 68.

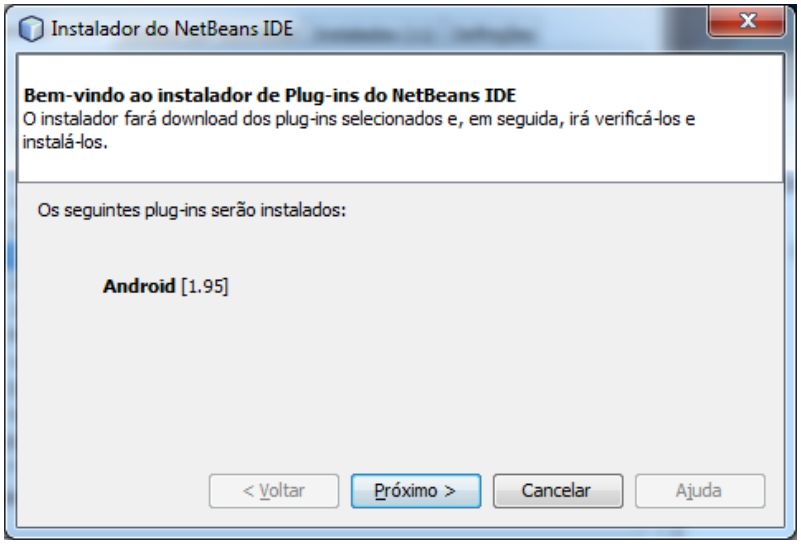

Figura 68 Instalador do NetBeans IDE – Android 1

Fonte: Autoria Própria

Nesta janela aciona-se o botão *Próximo* para continuar a instalação e abrir o próximo passo, conforme Figura 69.

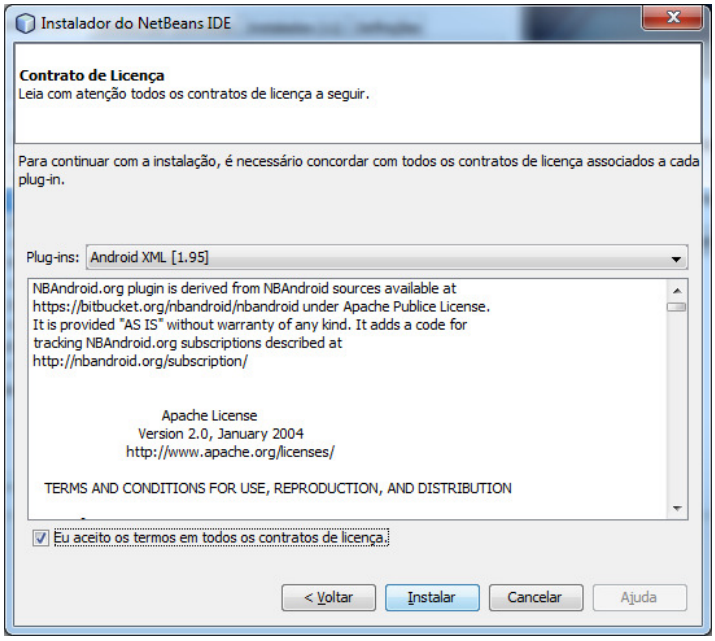

Figura 69 Instalador do NetBeans IDE – Android 2

Fonte: Autoria Própria

Nesta janela, deve-se selecionar o *check* para aceitar os termos de uso e acionar o botão *Instalar*. Isso irá abrir a janela *Verificar Certificado,* como na Figura 70.

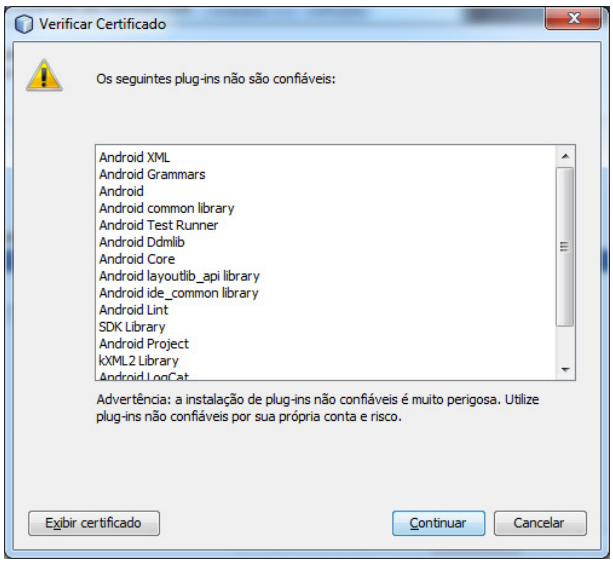

Figura 70 NetBeans Verificar Certificado – Android

Fonte: Autoria Própria

Nesta janela, clica-se em *Continuar* para iniciar a instalação, ao final será apresentada a janela da Figura 71.

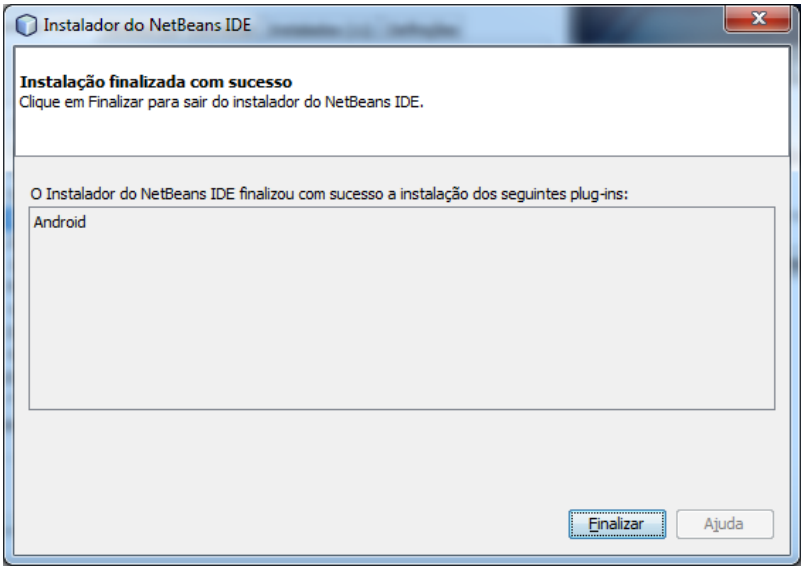

Figura 71 Instalador do NetBeans IDE – Android Final

Fonte: Autoria Própria

Como o NBAndroid instalado no NetBeans, basta configurar o ambiente informando onde se encontra o Android SDK. Para isso, deve-se acessar o menu *Ferramentas* e selecionar *Opções*, isso irá abrir a janela de Opções do NetBeans, como na Figura 72.

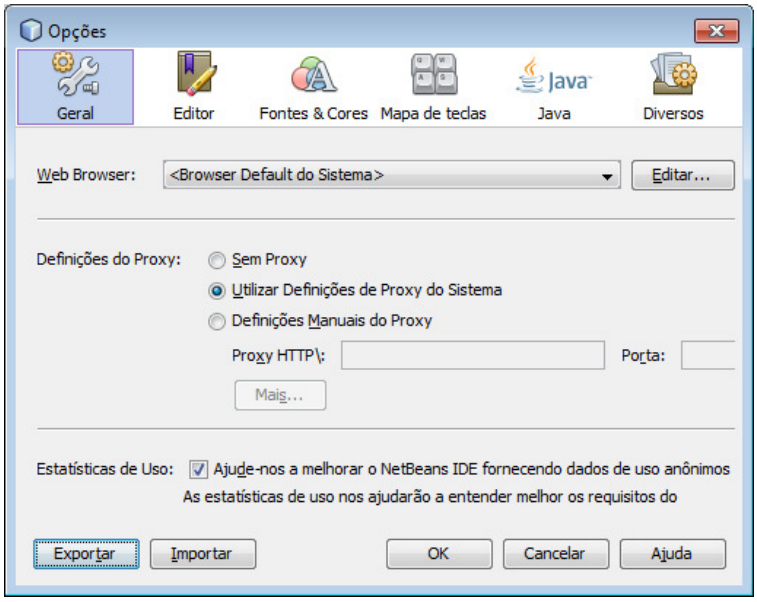

Figura 72 NetBeans – Opções

Fonte: Autoria Própria

Nesta janela, na aba *Diversos,* dentro da sub-aba *Android,* deve-se informar a pasta onde o Android SDK foi instalado, conforme Figura 73.

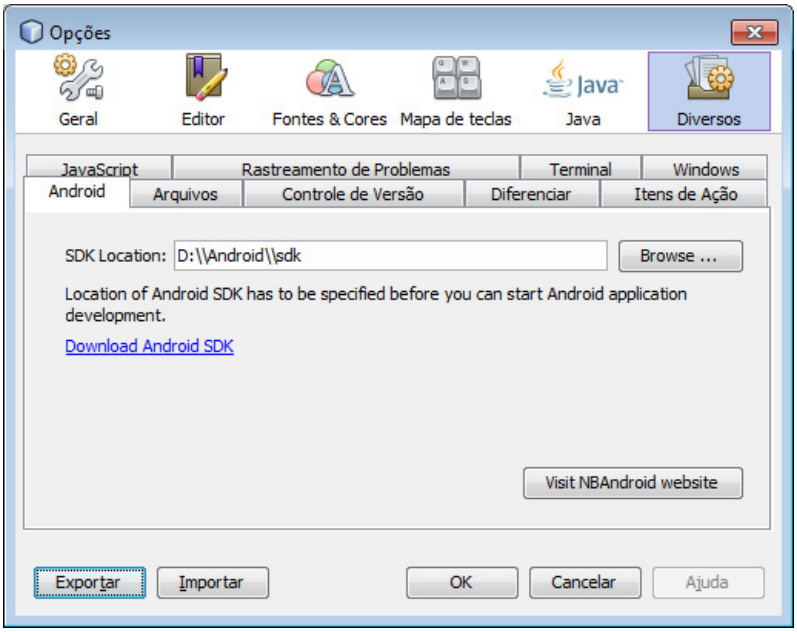

Figura 73 NetBeans – Opções Android

Fonte: Autoria Própria

Desta forma, o ambiente de desenvolvimento Android está completo e pronto para o desenvolvimento do jogo proposto pelo presente trabalho.