

Jogo Para a Plataforma Android utilizando Web Service

Rodrigo Duarte Louro

MONOGRAFIA APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

Programa: Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Marco Dimas Gubitoso

São Paulo, 01 de dezembro de 2014

Agradecimentos

Agradeço a todos os funcionários e professores do Instituto de Matemática e Estatística da Universidade de São Paulo por todo o aprendizado que me foi oferecido. Agradeço em especial o Professor Doutor Marco Dimas Gubitoso pela orientação deste trabalho. Agradeço os meus colegas de curso, do BCC 2010, sem vocês não teria chegado até este ponto do curso. Agradeço principalmente a minha família e a minha namorada, que estiveram sempre ao meu lado me apoiando e me dando forças nos momentos mais difíceis, sem vocês eu não seria nada.

Resumo

LOURO, R. D. **Jogo para a plataforma android utilizando web service**. 2014. Monografia - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2014.

Com o grande aumento do número de celulares, estima-se que, no ano de 2015 este número seja maior do que a população do planeta [Nog14]. Com isso, a indústria de jogos, que movimenta cerca de 3 bilhões de dólares todos os anos no Brasil [Ind14] ganhou uma grande plataforma até então pouco utilizada para seus produtos.

Cerca de 85% dos celulares ativos utilizam sistema operacional android[And14]. A Google¹, empresa responsável pelo sistema, possui uma loja online de aplicativos chamada Google Play², com mais de 1,3 milhão de aplicativos, dentre os quais mais de 50% tratam-se de jogos.

Dentre tais jogos destacam-se os que possuem a opção multijogador. Os jogos multijogadores tratam-se daqueles em que de alguma maneira existe troca de informações entre usuários, e que não deixam a sensação do jogador estar jogando sozinho.

Neste contexto, este trabalho consiste no desenvolvimento de um jogo multijogador para o sistema android utilizando web service para a persistência de dados, fazendo com que seja possível a troca de informações entre jogadores.

Utilizando o ADT, a engine AndEngine e as ferramentas disponibilizadas pelo Facebook foi possível implementar um protótipo jogável para dispositivos com sistema operacional android.

Palavras-chave: jogo, multijogador, android, andEngine, web service.

¹Google: <http://www.google.com.br>

²Google Play: <http://play.google.com/store>

Sumário

| | |
|---|-------------|
| Lista de Abreviaturas | xi |
| Lista de Figuras | xiii |
| Lista de Tabelas | xv |
| I Parte Objetiva | 1 |
| 1 Introdução | 3 |
| 1.1 Motivação | 3 |
| 1.2 Objetivos | 4 |
| 1.3 Mecânica | 4 |
| 1.4 Organização do Trabalho | 4 |
| 2 Conceitos | 5 |
| 2.1 Jogos Móveis | 5 |
| 2.1.1 Jogos multijogador | 5 |
| 2.1.2 Jogos Sociais | 5 |
| 2.2 Android | 6 |
| 3 Tecnologias Utilizadas | 9 |
| 3.1 Ferramentas | 9 |
| 3.1.1 ADT | 9 |
| 3.1.2 AndEngine | 9 |
| 3.1.3 PostgreSQL | 10 |
| 3.1.4 Nginx | 10 |
| 3.1.5 Facebook SDK | 10 |
| 3.2 Linguagens de Programação | 10 |
| 3.2.1 Java | 10 |
| 3.2.2 PHP | 11 |
| 4 Banco de Dados | 13 |
| 4.1 Modelo Relacional | 13 |
| 4.2 Relações | 13 |
| 4.2.1 Jogador | 13 |

| | | |
|----------|---|-----------|
| 4.2.2 | TipoCarta | 14 |
| 4.2.3 | Carta | 14 |
| 4.2.4 | Dicas | 14 |
| 4.2.5 | Desafios | 15 |
| 4.2.6 | Historico_Jogo | 15 |
| 4.2.7 | Historico_Estatistica | 16 |
| 4.3 | Arquivos de criação e de inserção | 16 |
| 5 | Protocolo de comunicação | 17 |
| 5.1 | Sintaxe | 17 |
| 5.1.1 | Estrutura de Dados | 17 |
| 5.1.2 | Campos Obrigatórios | 17 |
| 5.1.3 | Outros Campos | 17 |
| 5.2 | Semântica | 18 |
| 5.2.1 | Requisição <i>SignUp</i> | 18 |
| 5.2.2 | Requisição <i>FinishOldRound</i> | 18 |
| 5.3 | Servidor | 19 |
| 5.3.1 | TrataRequisicao | 19 |
| 5.3.2 | ExecuteQuery | 19 |
| 5.4 | Jogo | 19 |
| 5.4.1 | HTTPResponseListener | 19 |
| 5.4.2 | HTTPPostRequester | 20 |
| 5.4.3 | MakeParameters | 20 |
| 5.4.4 | JSONParser | 20 |
| 5.4.5 | Fluxo da requisição | 20 |
| 6 | Implementação do Jogo | 23 |
| 6.1 | Fluxo de Telas | 23 |
| 6.2 | Principais Classes | 24 |
| 6.2.1 | GameActivity | 24 |
| 6.2.2 | ResourcesManager | 24 |
| 6.2.3 | SceneManager | 24 |
| 6.2.4 | GameManager | 24 |
| 6.2.5 | BaseScene | 25 |
| 6.2.6 | FacebookFacade | 25 |
| 6.3 | Integração Facebook | 25 |
| 6.4 | Novo Jogo | 25 |
| 6.4.1 | Amigo do Facebook | 25 |
| 6.4.2 | Oponente Aleatório | 26 |
| 7 | Resultados Obtidos | 27 |
| 8 | Conclusão | 29 |
| 9 | Trabalhos Futuros | 31 |

| | |
|---|-----------|
| II Parte Subjetiva | 33 |
| 10 Subjetivo | 35 |
| 10.1 Desafios e frustrações | 35 |
| 10.1.1 Não conhecimento das linguagens utilizadas | 35 |
| 10.1.2 Documentação da engine | 35 |
| 10.1.3 Interface gráfica | 35 |
| 10.1.4 SDK Facebook | 35 |
| 10.2 Disciplinas Relevantes para o desenvolvimento do trabalho | 36 |
| 10.2.1 MAC0122 Princípios de Desenvolvimento de Algoritmos | 36 |
| 10.2.2 MAC0242 Laboratório de Programação II | 36 |
| 10.2.3 MAC0439 Laboratório de Bancos de Dados | 36 |
| 10.2.4 MAC0448 Programação para Redes de Computadores | 36 |
| Referências Bibliográficas | 37 |

Lista de Abreviaturas

| | |
|------|---|
| ADT | Ferramentas de Desenvolvedor Android (<i>Android Development Tools</i>) |
| SDK | Kit de Desenvolvimento de Software (<i>Software Development Kit</i>) |
| GIMP | Programa de manipulação de imagem do GNU (<i>GNU Image Manipulation Program</i>) |
| JSON | JavaScript Oriented Notation |
| IDE | Ambiente Integrado de Desenvolvimento (<i>Integrated Development Environment</i>) |
| SGBD | Sistema Gerenciador de Banco de Dados (<i>Data Base Management System</i>) |
| IP | Protocolo de Internet (<i>Internet Protocol</i>) |
| SQL | Linguagem de Consulta Estruturada (<i>Structured Query Language</i>) |

Lista de Figuras

| | | |
|-----|--|----|
| 2.1 | O T-Mobile G1, o primeiro dispositivo com Android | 6 |
| 2.2 | Distribuição dos sistemas operacionais para celular. | 7 |
| 4.1 | Modelo Relacional | 13 |
| 5.1 | Fluxo executado pelo jogo para uma requisição | 20 |
| 6.1 | Fluxograma de telas do jogo | 23 |
| 7.1 | Principais telas do jogo | 27 |

Lista de Tabelas

| | | |
|-----|--|----|
| 4.1 | Tabela Jogador | 13 |
| 4.2 | Tabela TipoCarta | 14 |
| 4.3 | Tabela Carta | 14 |
| 4.4 | Tabela Dicas | 14 |
| 4.5 | Tabela Desafios | 15 |
| 4.6 | Tabela Historico_Jogo | 15 |
| 4.7 | Tabela Historico_Estatistica | 16 |

Parte I

Parte Objetiva

Capítulo 1

Introdução

Os jogos eletrônicos se tornaram cada vez mais presentes no dia a dia das pessoas. O primeiro jogo de computador foi criado em 1958 por Willian Higinbotham e Robert Dvorak. Chamava-se “Tennis for Two” e se tratava da simulação de uma partida de tênis, não havia placar e a tela era feita de um cinescópio de fósforo verde monocromático [Num07]. Desde então os jogos acompanharam a enorme evolução da computação. Com o advento e a popularização da internet, a criação de aparelhos eletrônicos que possibilitassem que todas as pessoas ficassem conectadas de maneira mais simples e por períodos maiores era questão de tempo. Estes aparelhos são os conhecidos tablets e smartphones, que fizeram com que a computação móvel ficasse acessível para o grande público.

A partir de então, diferentes e inúmeros tipos de mercados se abriram ou se adaptaram à computação móvel. Um deles, talvez o maior deles, foi a indústria de jogos, que se adaptou de maneira muito rápida e eficaz a nova plataforma.

Praticamente qualquer pessoa possui fácil acesso a alguma plataforma móvel, como tablets e smartphones, o que fez com que o estilo de jogo chamado casual, que em geral são jogos que apresentam mecânica com alto grau de simplicidade e que, ao contrário dos jogos tradicionais, exigem menos tempo e esforço do jogador, se tornasse uma grande forma de entretenimento rápido.

Dentre os jogos casuais podemos citar alguns exemplos de grande sucesso em que o sistema é multijogador, ou seja, jogos que de alguma forma realizam duelos entre os jogadores, havendo um vencedor e um perdedor (SongPop, BikeRace, What’s the Movie?, etc) ou jogos em que exista a necessidade de que amigos do jogador enviem a ele algum tipo de ajuda para o andamento do jogo (CandyCrush, FarmVille, Hay Day, etc). Para o desenvolvimento do sistema multijogador, geralmente utiliza-se alguma rede social (facebook, twitter, instagram, orkut, etc) como fonte de dados. Jogos que utilizam tais dados ficaram conhecidos como jogos sociais.

Diante disto, a proposta deste trabalho consiste no desenvolvimento de um jogo com sistema multijogador para a plataforma android, que utiliza um Web Service para fazer a persistência de dados. Será um quiz inspirado no jogo de tabuleiro “Biografia”, da empresa Algazarra, onde são expostos fatos sobre a vida de personalidades conhecidas e, através desses fatos, o jogador terá como objetivo descobrir quem é a personalidade.

1.1 Motivação

O cenário atual do mercado de jogos é muito atrativo e diversificado. Hoje em dia, um país como o Brasil, que a poucos anos atrás possuía uma indústria de jogos praticamente inexistente, passou a ter uma indústria especializada e um número muito grande de pequenas e médias empresas voltadas ao desenvolvimento de jogos.

A maior motivação neste trabalho é desenvolver uma aplicação para a plataforma android, pois com a experiência adquirida no desenvolvimento de jogos para plataformas móveis da apple¹, o autor obteve grande interesse pela área. A oportunidade do maior aprofundamento na linguagem

¹Apple: <http://www.apple.com>

Java também é um dos fatores de motivação, uma vez que não houve grande contato com tal linguagem durante a graduação, além de aplicar diferentes áreas do conhecimento adquirido no curso de Bacharelado em Ciência da Computação.

1.2 Objetivos

Desenvolver um jogo multijogador para plataforma android, em que o objetivo seja descobrir uma personalidade utilizando o menor número de dicas possível. O jogo se comunicará com um web service para realizar a troca de informações entre banco de dados e o aplicativo, tornando possível a opção multijogador, que se realizará de maneira síncrona. O jogo será desenvolvido com auxílio da game engine AndEngine² e será escrito em Java, utilizando o ambiente ADT.

1.3 Mecânica

Biografia é um jogo baseado em turnos que só pode ser jogado contra um oponente por vez.

Um turno é começado pelo jogador desafiante, este sorteia uma carta que será a carta daquele turno, posteriormente o desafiante tenta descobrir qual a personalidade da carta sorteada através das dicas, ganhando então uma quantidade de pontos. Feito isso, o desafiante termina seu turno, fazendo com que o jogador desafiado tenha a vez. O jogador desafiado responderá o turno jogando exatamente a mesma carta que foi jogada pelo desafiante, com o objetivo de fazer mais pontos que ele. Ao fim do turno do jogador desafiado, aquele jogador, desafiante ou desafiado, que obter a maior quantidade de pontos é então declarado vencedor daquele turno, acrescentando um ponto ao placar geral que marca a quantidade de turnos vencidos, dando então fim ao turno corrente e liberando o próximo a ser iniciado.

1.4 Organização do Trabalho

No Capítulo 2, são apresentados conceitos sobre assuntos utilizados corriqueiramente no texto. No Capítulo 3 são descritas todas as ferramentas e tecnologias utilizadas no projeto. No Capítulo 4 é explicada toda a estrutura do banco de dados utilizado para a persistência das informações necessárias. No Capítulo 5 é explicada a sintaxe e a semântica do protocolo desenvolvido para a comunicação entre o jogo e o web service. No Capítulo 6 são mostrados os pontos mais importantes da implementação do jogo. No Capítulo 7 são apresentados os resultados do projeto. No Capítulo 8 é feita a conclusão do projeto e finalmente no Capítulo 9 são apresentados alguns dos possíveis trabalhos futuros em cima deste jogo.

²AndEngine: <http://www.andengine.org>

Capítulo 2

Conceitos

2.1 Jogos Móveis

2.1.1 Jogos multijogador

Jogos multijogador são simulações de ambientes onde cada jogador busca atingir um certo objetivo através da interação com outros jogadores e com o ambiente [Cec05]. Esta interação é efetuada de duas maneiras principais: Jogos de tempo real, onde o jogador envia comandos de maneira assíncrona a passagem do tempo no ambiente virtual. Um jogo de carros onde o jogador é o piloto, é um exemplo de jogo de tempo real; Jogos de turnos, em que o jogador envia comandos de forma síncrona ao tempo virtual. Um jogo de xadrez, onde a partida não avança até que o jogador da vez realize a sua jogada, é um exemplo de jogo baseado em turnos [Cec05].

Trataremos aqui apenas o caso síncrono, em que o jogo é dividido em turnos. O termo “Jogos Multijogador” será utilizado nesse sentido. Dentre jogos baseados em turnos destacam-se os que promovem algum tipo de duelo entre os jogadores, ou seja, há um vencedor e um perdedor a cada turno e ou, conjunto de turnos. Neste projeto esta é a forma escolhida; a cada turno é definido um vencedor e a contagem total de partidas é acrescida de uma unidade.

2.1.2 Jogos Sociais

A definição de jogo social vem de muito antes da criação das redes sociais que conhecemos hoje (facebook, twitter, linkedin, etc). Toma-se por jogo social qualquer jogo que necessita da interação de duas ou mais pessoas para ser jogado. Temos como exemplo o xadrez e jogos de cartas em geral.

A partir da grande popularização da internet e do massivo crescimento das redes sociais, a expressão “jogos sociais” se popularizou como sendo jogos online que têm como característica não mais a interação física entre os usuários, mas sim uma interação virtual destes. Com a facilidade obtida em manter relações de diversos tipos com outros jogadores em um ambiente simulado, os jogos passaram a utilizar essa ferramenta para propagação, visando obter mais usuários.

As maneiras de utilizar as redes sociais para conseguir mais jogadores são inúmeras. Utilizar os dados de cadastro, as listas de relações e estatísticas sobre o comportamento dos usuários são algumas das maneiras que podem esboçar uma ideia do quão amplo e ao mesmo tempo segmentado este mercado pode se tornar. Além da obtenção de dados, as redes sociais colaboram de outras formas na disseminação de um jogo. Como principal exemplo podemos tomar o conteúdo obtido pelo jogo que os próprios jogadores compartilham com suas relações nas redes sociais; este conteúdo motiva outros usuários das redes a jogarem o jogo e funciona também como propaganda.

Neste projeto utilizamos dados cadastrais e a lista de relações do facebook para tornar possível o duelo entre usuários.

2.2 Android

Em 2003 Andy Rubin ¹ fundou, junto com outros grandes nomes, o Android Inc., uma sociedade que ofereceu uma nova tipologia de sistemas operacionais móveis, de código aberto e gratuito para quem quisesse utilizá-lo [Mer14].

A Google comprou em 2005 a Android Inc. e assim nasceu a Google Mobile Division. No ano de 2008 foi lançado o primeiro celular com sistema operacional android, o T-Mobile G1 [Mer14].



Figura 2.1: O T-Mobile G1, o primeiro dispositivo com Android
Fonte: [Mer14]

Desde o lançamento da primeira versão do android todas as subsequentes foram realizadas de maneira incremental, adicionando novas funcionalidades e corrigindo erros das anteriores. As versões que possuem mudanças significativas no sistema são batizadas com nomes de sobremesas em ordem alfabética: [And13]

- 1.0 e 1.1²
- 1.5 - Cupcake
- 1.6 - Donut
- 2.0 / 2.1 - Eclair
- 2.2 - Froyo
- 2.3 - GingerBread
- 3.x - HoneyComb
- 4.0 - Ice Cream Sandwich
- 4.1 - Jelly Bean
- 4.4 - KitKat
- 5.0 - Lollipop³

A Google mantém uma loja online de aplicativos chamada Google Play. Esta é a ferramenta que usuários utilizam para a instalação de novos programas de diversos tipos (utilitários, jogos, livros, etc) em seus celulares. Para os desenvolvedores a Google Play é uma excelente opção de mercado, dado a facilidade na disponibilização de seus produtos além do grande número de usuários.

¹1963, Chappaqua, Nova Iorque, EUA

²Versões lançadas sem nome

³Versão lançada em 15 de outubro de 2014

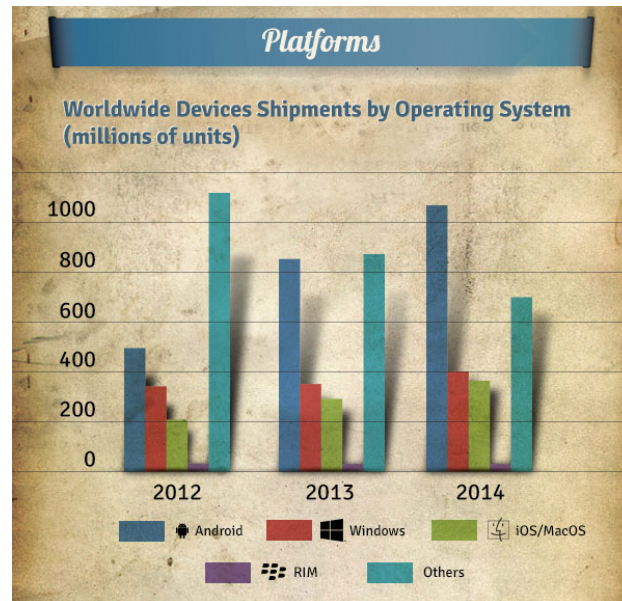


Figura 2.2: *Distribuição dos sistemas operacionais para celular.*
Fonte: [Hep13]

Estima-se que até o final do ano de 2015 existam 2,5 bilhão de smarthones ativos , destes cerca de 85 % utilizam o sistema android[And14], o que comprova a relevância e a importância de se obter conhecimento no processo de desenvolvimento de aplicativos para essa plataforma.

Capítulo 3

Tecnologias Utilizadas

3.1 Ferramentas

3.1.1 ADT

O ADT ¹ é um plugin para a IDE Eclipse² disponibilizado pela Google para o desenvolvimento de aplicativos android. Com ele a montagem do ambiente para o início do desenvolvimento de qualquer aplicação android é mais simples e rápida, pois neste ambiente já está configurado o android SDK.

O android SDK disponibiliza as ferramentas necessárias para começar a desenvolver as aplicações na plataforma, utilizando a linguagem de programação Java. Ele inclui funcionalidades úteis na criação de aplicativos, como por exemplo emuladores de aparelhos, ferramentas de depuração, visualização da utilização de memória, análise de desempenho, dentre outras.

3.1.2 AndEngine

A AndEngine é uma engine de jogos em duas dimensões desenvolvida por Nicolas Gramlich³, voltada unicamente para a plataforma android. Sua primeira versão foi implementada durante o ano de 2010.

O papel de uma engine no desenvolvimento de jogos é auxiliar o desenvolvedor com abstrações de alto nível, para que se evite complicações e preocupações com detalhes de níveis mais baixos. A AndEngine oferece suporte a propriedades físicas como gravidade, peso, densidade, velocidade, elasticidade, entre outros. A renderização de imagens na tela, detecção de toques simples e múltiplos, transições entre cenas e suporte a efeitos de sons são apenas algumas das facilidades que essa engine oferece.

A principal característica da engine criada por Nicolas Gramlich é que, além de totalmente gratuita, ela possui código aberto e pode ser incorporada por qualquer pessoa a qualquer jogo, tendo fins lucrativos ou não. Isso fez com que se criasse uma comunidade de desenvolvedores que utilizam a engine, e por possuir código aberto qualquer pessoa pode propor mudanças, corrigir erros, e principalmente criar novas extensões que possam ser utilizadas por qualquer pessoa sem a necessidade de atualização ou que se faça uma nova publicação para a adição da nova funcionalidade. Este processo é mais simples e ocorre de maneira natural, fazendo com que a AndEngine permaneça em constante evolução.

Neste projeto foi utilizada a última versão da AndEngine⁴, nela a engine trabalha com a nova versão da biblioteca gráfica OpenGL ES 2.0, além de fazer com que todos os objetos utilizados tenham como ponto de âncora (utilizados para posicionamento na tela) o seu centro geométrico.

¹ADT: <http://developer.android.com/tools/sdk/eclipse-adt.html>

²Eclipse: <https://www.eclipse.org/>

³1987, Schriesheim, Alemanha

⁴Disponível em: <https://github.com/nicolagramlich/AndEngine/tree/GLES2-AnchorCenter>

3.1.3 PostgreSQL

O PostgreSQL é um SGBD Relacional, utilizado para armazenar informações de soluções de informática em todas as áreas de negócios existentes, bem como administrar o acesso a estas informações [Mil08].

Este SGBD possui código aberto sob a licença BSD⁵ e com isso o uso deste sistema em aplicações comerciais é autorizado de maneira gratuita [Mil08].

Neste projeto o PostgreSQL foi a ferramenta utilizada para fazer a recuperação e a persistência dos dados gerados pelas relações entre os jogadores. A versão utilizada foi a 9.1.13, lançada em 20 de março de 2014.

3.1.4 Nginx

Nginx é um servidor proxy HTTP e reverso, bem como um servidor de proxy de email, que possui código aberto, escrito por Igor Sysoev⁶ desde 2005. De acordo com a NetCraft⁷ o nginx é utilizado por 20,34% dos sites mais movimentados da internet [Ngi14].

O principal motivo pela escolha do nginx como servidor do projeto se deu pelo seu menor consumo de memória com relação ao Apache⁸. Já que a ideia do projeto é que o maior número possível de requisições sejam tratadas concomitantemente, o consumo de memória foi um ponto vital nesta decisão.

3.1.5 Facebook SDK

O Facebook developers⁹ (site direcionado para desenvolvedores que utilizam alguma ferramenta disponibilizada pelo facebook), possui um SDK específico para a plataforma android. Com o uso deste mecanismo é possível que qualquer aplicação android possa interagir de diversas maneiras com a rede social.

Neste projeto a funcionalidade de login, método que faz com que o aplicativo tenha acesso a informações como nome, idade, foto, etc, além da requisição de dados dos amigos do usuário, foram os métodos utilizados do SDK e viabilizaram a funcionalidade multijogador na implementação do jogo.

3.2 Linguagens de Programação

3.2.1 Java

Java é uma linguagem orientada a objetos que tem como principal característica a portabilidade. O bytecode, que é o código gerado pelo compilador Java, pode ser transportado entre plataformas distintas, desde que estas suportem Java, e com isso não é necessário recompilar um programa para que ele rode em máquinas e sistemas diferentes.

Esta linguagem foi criada na década de 90 por uma equipe de programadores chefiada por James Gosling¹⁰, na empresa Sun Microsystems¹¹

A versão 7 do Java foi a linguagem utilizada durante todo o processo de desenvolvimento do aplicativo.

⁵BSD: Berkeley Software Distribution

⁶1970, Almaty, Kazakhstan

⁷NetCraft: <http://www.netcraft.com/>

⁸Apache: <http://www.apache.org/>

⁹Facebook Developers: <https://developers.facebook.com/>

¹⁰1955, Calgary, Canada

¹¹Comprada pela empresa Oracle em 2010: <http://www.oracle.com/us/sun/index.html>

3.2.2 PHP

O PHP é uma linguagem de script, de código aberto e que tem como objetivo primário a geração de conteúdo dinâmico para páginas da internet [dMGFN07]. A sua primeira versão foi escrita em 1994 por Rasmus Lerdorf¹², que criou uma série de utilitários para monitorar sua página pessoal e obter informações sobre seus visitantes.

O PHP possui fácil conexão com bancos de dados, o que justifica sua escolha para fazer o tratamento das requisições ao servidor que precisam recuperar ou persistir alguma informação. Neste projeto a versão utilizada foi a 5.5.9, criada em 6 de fevereiro de 2014.

¹²Qeqertarsuaq, 22 de novembro de 1968

Capítulo 4

Banco de Dados

Para que as informações necessárias ao funcionamento correto do jogo pudessem ser recuperadas de maneira eficiente e consistente foi modelado um banco de dados relacional utilizando o SGBD PostgreSQL.

4.1 Modelo Relacional

Foram necessárias sete relações para armazenar os dados.



Figura 4.1: Modelo Relacional

Na figura os campos sublinhados representam as chaves primárias da relação, já as setas representam as respectivas chaves estrangeiras.

4.2 Relações

4.2.1 Jogador

A tabela Jogador armazena os dados dos usuários utilizados pelo jogo.

| | | | | |
|-----------|------|--------|-----------|------|
| <u>id</u> | nome | moedas | especiais | foto |
|-----------|------|--------|-----------|------|

Tabela 4.1: Tabela Jogador

- **id** - É a chave primária da tabela, se trata do id do facebook do jogador.
- **nome** - Nome completo do jogador, informação retirada do perfil no facebook.
- **moedas** - Quantidade de moedas que o jogador possui.
- **especiais** - Quantidade de especiais que o jogador possui.
- **foto** - Url da foto de perfil do facebook.

4.2.2 TipoCarta

A tabela TipoCarta armazena todos os possíveis tipos de cartas utilizados no jogo.

| | |
|-----------|------|
| <u>id</u> | tipo |
|-----------|------|

Tabela 4.2: *Tabela TipoCarta*

- **id** - Chave primária da tabela, trata-se de um id sequencial.
- **tipo** - Nome do tipo de carta.

4.2.3 Carta

A tabela carta armazena os dados referentes a cada carta cadastrada no jogo.

| | | | |
|-----------|------|---------------|-----------|
| <u>id</u> | nome | id_tipo_carta | link_foto |
|-----------|------|---------------|-----------|

Tabela 4.3: *Tabela Carta*

- **id** - Chave primária da tabela, trata-se de um id sequencial.
- **nome** - Nome da personalidade representada na carta.
- **id_tipo_carta** - Chave estrangeira que representa qual o tipo da carta.
- **link_foto** - Url da foto da personalidade.

4.2.4 Dicas

A tabela dicas representa cada uma das dicas presentes nas cartas do jogo. São 10 dicas para cada carta

| | | |
|-----------|----------|-------|
| <u>id</u> | id_carta | texto |
|-----------|----------|-------|

Tabela 4.4: *Tabela Dicas*

- **id** - Faz parte da chave primária da tabela, e representa o número da dica de uma determinada carta, podendo ter valores entre 1 e 10.
- **id_carta** - Faz parte da chave primária da tabela e é também chave estrangeira que representa de qual carta se trata a dica.
- **texto** - Texto de um fato sobre a biografia da personalidade da carta.

4.2.5 Desafios

A tabela de desafios representa os dois últimos turnos jogados por dois jogadores que estão competindo. Nesta tabela, o jogador 1 representa o desafiante e o jogador 2 o desafiado. Como a cada turno o desafiado manda um jogo de volta para o desafiante, este torna-se desafiado e aquele o desafiante, alternando assim os papéis. Por esse estilo de jogo cada partida possui duas tuplas na tabela desafios, uma que representa o turno de ida, e outra, que possui chave primária trocada, ou seja o jogador 1 vira jogador 2 e vice-versa, que representa o turno de volta.

| | | | |
|--------------------|--------------------|----------|------------|
| <u>id_jogador1</u> | <u>id_jogador2</u> | id_carta | pontuacao1 |
| pontuacao2 | status | | |

Tabela 4.5: *Tabela Desafios*

- **id_jogador1** - Faz parte da chave primária da tabela, representa o id do jogador desafiante.
- **id_jogador2** - Faz parte da chave primária da tabela, representa o id do jogador desafiado.
- **id_carta** - Representa a carta utilizada no último turno que o jogador desafiante enviou para o desafiado.
- **pontuacao1** - Quantidade de pontos obtidos pelo desafiante no turno corrente.
- **pontuacao2** - Quantidade de pontos obtidos pelo desafiado no turno corrente.
- **status** - Trata-se de um inteiro entre 0 e 4 que representa o estado do turno.
 - **0:** Representa que nenhum dos jogadores jogou este turno.
 - **1:** Representa que o desafiante está jogando.
 - **2:** Representa que o desafiante já enviou o desafio.
 - **3:** Representa que o desafiado está jogando.
 - **4:** Representa que o desafiado já respondeu o desafio.

4.2.6 Historico_Jogo

A tabela Historico_Jogo guarda o placar total do jogo, fazendo a totalização de todos os turnos desde o primeiro. Nesta tabela o id do jogador 1 é sempre numericamente menor do que o do jogador 2, fazendo com que a consulta SQL utilizada para a recuperação dos dados seja feita de maneira mais simples.

| | | | |
|--------------------|--------------------|-----------|-----------|
| <u>id_jogador1</u> | <u>id_jogador2</u> | vitorias1 | vitorias2 |
|--------------------|--------------------|-----------|-----------|

Tabela 4.6: *Tabela Historico_Jogo*

- **id_jogador1** - Faz parte da chave primária da tabela, representa um dos jogadores.
- **id_jogador2** - Faz parte da chave primária da tabela, representa um dos jogadores.
- **vitorias1** - Quantidade de vitórias totais do jogador 1.
- **vitorias2** - Quantidade de vitórias totais do jogador 2.

4.2.7 Historico_Estatistica

A tabela Historico_Estatistica guarda informações estatísticas sobre o desempenho de cada jogador em determinado tipo de carta.

| | | | |
|------------|---------------|---------|---------|
| id_jogador | id_tipo_carta | jogadas | acertos |
|------------|---------------|---------|---------|

Tabela 4.7: *Tabela Historico_Estatistica*

- **id_jogador** - Faz parte da chave primária da tabela, representa um jogador.
- **id_tipo_carta** - Faz parte da chave primária da tabela, representa um tipo de carta jogado.
- **jogadas** - Quantidade de vezes em que o tipo de carta foi jogado pelo jogador
- **acertos** - Quantidade de vezes em que o jogador acertou a personalidade jogando este tipo de carta.

4.3 Arquivos de criação e de inserção

Foram criados dois arquivos com extensão SQL para fazer a criação das tabelas do banco de dados:

- **ModeloFisico** - O arquivo ModeloFisico.sql executa a criação de toda a estrutura (tabelas, chaves primárias, chaves estrangeiras, etc) presente no banco de dados de Biografia.
- **Populate** - O arquivo Populate.sql popula o banco de dados com uma quantidade de dados suficiente para que alguns testes envolvendo jogadores diferentes, e cartas diferentes possam ser executados.

Ambos os arquivos podem ser executados através da linha de comando. Dentro do SGBD PostgreSQL basta digitar:

“\i Nome_do_arquivo” para que o mesmo seja executado.

Capítulo 5

Protocolo de comunicação

Para que a troca de informações entre o jogo e o web service fosse feita de maneira confiável e bem definida, foi criado um protocolo de comunicação.

5.1 Sintaxe

5.1.1 Estrutura de Dados

A estrutura JSON¹ foi utilizada no protocolo. JSON é uma formatação leve de troca de dados, em formato texto e completamente independente de linguagem.[JSO]

JSON está constituído em duas estruturas:

- Uma coleção de pares chave/valor
- Uma lista ordenada de valores

O protocolo utiliza conjuntamente estas duas estruturas para a transmissão de dados.

5.1.2 Campos Obrigatórios

Nos objetos JSON utilizados para comunicação, foram definidos campos obrigatórios, que precisam estar presentes para que qualquer requisição seja respondida. São eles:

- **message:** Todas as informações do JSON precisam estar dentro da chave *message*. Toda a informação que estiver fora da chave *message* é desconsiderada, e caso a chave *message* não exista o servidor retorna um código de erro.
- **requestId:** Todas as possíveis requisições possuem um identificador único. Este identificador precisa ser passado como parâmetro dentro da chave *requestId*. Caso o conteúdo dessa chave não seja um identificador conhecido ou este campo não exista, o servidor retornará um código de erro.

5.1.3 Outros Campos

Os demais campos de conteúdo do JSON podem seguir qualquer formato, desde que sejam consistentes e que contenham as informações necessárias para cada requisição.

¹JavaScript Oriented Notation

5.2 Semântica

Para descrever a semântica do protocolo serão explicados a seguir, dois exemplos de possíveis requisições.

5.2.1 Requisição *SignUp*

A requisição *SignUp* é executada apenas uma vez para cada usuário. Essa requisição faz com que o banco de dados cadastre o usuário na tabela jogador, o que faz com que o usuário se torne apto a receber e enviar desafios. O JSON passado ao servidor para tal requisição deve seguir o seguinte formato:

```
1 {
2   "message": {
3     "requestId"    : "SignUp",
4     "userID"       : "12345678",
5     "userName",    : "Rodrigo Duarte",
6     "userCoins"    : "10",
7     "userPowerUps" : "10"
8   }
9 }
```

Para essa requisição, além dos campos obrigatórios, são necessários os campos:

- `userID`: Identificador do usuário no facebook, que servirá como identificador do jogador.
- `userName`: Nome do usuário utilizado no facebook, que será o nome do usuário no jogo.
- `userCoins`: Quantidade inicial de moedas que o jogador possui.
- `userPowerUps`: Quantidade inicial de especiais que o jogador possui.

5.2.2 Requisição *FinishOldRound*

A requisição *FinishOldRound* é executada ao fim do turno quando o jogador está respondendo um desafio. Essa requisição salva as informações do turno na tabela desafio em que o jogador que acabou de jogar é o jogador desafiado. O JSON passado ao servidor para tal requisição deve seguir o seguinte formato:

```
1 {
2   "message": {
3     "requestId"    : "FinishOldRound",
4     "userID"       : "12345678",
5     "friendID",    : "87654321",
6     "score"        : "2000",
7     "tipoCartaID" : "2",
8     "correct"      : "true"
9   }
10 }
```

Para essa requisição, além dos campos obrigatórios, são necessários os campos:

- `userID`: Identificador do jogador.
- `friendID`: Identificador do oponente.

- score: Quantidade de pontos alcançados pelo jogador.
- tipoCartaID: Identificador do tipo da carta jogado, utilizado para atualizar a tabela Historico_Estatistica
- correct: Booleano que indica se o jogador acertou ou não o nome da personalidade.

5.3 Servidor

Do lado do servidor, as classes php *TrataRequisicao* e *ExecuteQuery* além de um arquivo chamado infoDB, que guarda informações da conexão com o banco de dados, foram implementados e são responsáveis por toda a comunicação com o banco de dados, além de montarem a resposta das requisições em formato JSON.

5.3.1 TrataRequisicao

A classe *TrataRequisicao* é a classe que recebe todas as requisições feitas ao servidor. Nesta classe é feita a decodificação do JSON recebido através da função *json_decode* do php.

Após a decodificação é verificado se o JSON recebido possui o campo *message* e, em caso afirmativo, o novo JSON a ser considerado é o que se encontra como valor da chave *message*. Então é feita a chamada da função *tratandoRequisicao*, que verifica se a chave *requestID* existe e contém um valor válido para requisição. Novamente em caso afirmativo um objeto da classe *ExecuteQuery* é instanciado e o método predefinido de acordo com o conteúdo da chave *requestID* é chamado.

Em qualquer caso não afirmativo é feita a montagem de um objeto JSON que contém um único campo, possuindo a chave *status* e o valor *error*, que indica que a requisição não foi executada com sucesso.

5.3.2 ExecuteQuery

A classe *ExecuteQuery* é a classe que executa as chamadas SQL no banco de dados. Existe um método nessa classe para cada possível requisição que o servidor trata. Cada método recebe um objeto JSON que deve possuir os parâmetros necessários para a execução da consulta ao banco.

Existem duas funções auxiliares que são chamadas por qualquer método que trata requisições, a função *getInfo* e a função *setInfo*, a primeira executa apenas operações de leitura de dados do banco, já a segunda executa as operações de escrita no banco. Ambas fazem tratamento da conexão com o banco, abrindo-a e fechando-a nos devidos momentos.

Em todos os casos em que a requisição é executada com sucesso é retornado um objeto JSON com um campo contendo a chave *status* e valor *ok*, além dos outros dados necessários para a correta resposta da requisição. Nos casos de falta de dados necessários, ou erro na conexão com o banco de dados, ou qualquer outro erro que venha a acontecer durante o tratamento da requisição é retornado um JSON de erro contendo uma única chave *status* com valor *error*.

5.4 Jogo

Do lado do jogo foi criado um pequeno conjunto de classes que tem como objetivo abstrair a comunicação com o servidor.

5.4.1 HTTPResponseListener

A classe *HTTPResponseListener* é uma interface Java que faz com que todas as classes que a implementem sejam obrigadas a possuir um método chamado *onResponse*, utilizado para receber a resposta da requisição feita ao servidor.

5.4.2 HTTPPostRequester

A classe HTTPPostRequester é a classe que de fato executa a requisição ao servidor. Ela guarda em um atributo privado o endereço IP do servidor, além de possuir uma classe interna chamada HttpPostRequest que herda da classe AsyncTask (presente no pacote android.os) a capacidade de executar código em segundo plano, ou seja, fazer com que a classe que fez a requisição não fique travada enquanto a resposta não chega.

O método *asynPost* cria uma nova instância da classe HttpPostRequest e faz com que a requisição para o banco de dados seja feita de maneira assíncrona.

5.4.3 MakeParameters

A classe MakeParameters é responsável pela montagem dos objetos JSON que serão passados ao servidor. Trata-se de um conjunto de métodos estáticos, em que cada um dos métodos representa uma requisição.

Todas as classes que desejam trocar informações com o servidor precisam chamar um método da classe MakeParameters que seja correspondente à requisição desejada. Tal método retorna um objeto JSON populado com todos os parâmetros necessários para a requisição. Posteriormente esse objeto JSON é repassado à classe HTTPPostRequester.

5.4.4 JSONParser

A classe JSONParser existe apenas para fazer a tradução da resposta do servidor para um objeto JSONObject do Java. A classe é chamada durante o tratamento da resposta de todas as requisições, e o objeto populado por ela é retornado como resposta para as classes que pediram as informações do servidor.

5.4.5 Fluxo da requisição

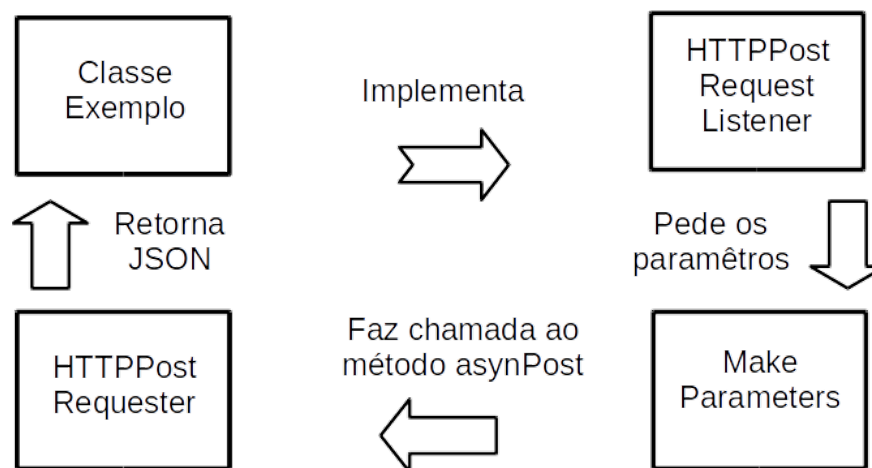


Figura 5.1: Fluxo executado pelo jogo para uma requisição

A figura acima mostra um fluxograma seguido por uma classe de exemplo do jogo para obter informações do servidor. Os passos a serem seguidos são:

- **Passo 1:** Primeiramente é necessário que a classe que deseja se comunicar com o servidor implemente a interface `HTTPResponseListener`, tornando-se obrigada, portanto, a ter um método que trata a resposta do servidor.
- **Passo 2:** O segundo passo é fazer uma chamada ao método correto, de acordo com a requisição, a classe `MakeParameters`, tendo como resposta um objeto JSON populado com os parâmetros a serem passados ao servidor.

- **Passo 3:** Posteriormente é feita a chamada ao método `asynPost` da classe `HTTPPostRequester`, que executa a requisição em segundo plano e após a chegada da resposta devolve o JSON com os dados pedidos à classe que os pediu.

Capítulo 6

Implementação do Jogo

Neste capítulo serão abordadas as principais características e funcionalidades implementadas no jogo.

6.1 Fluxo de Telas

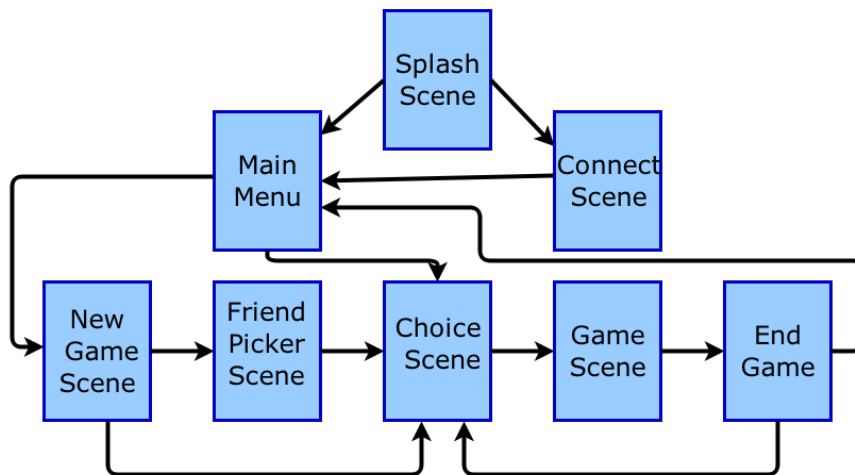


Figura 6.1: Fluxograma de telas do jogo

A figura acima representa os possíveis fluxos de telas do jogo Biografia. Todas as sessões são iniciadas a partir da tela *Splash Scene*, que nada mais é do que a tela de loading do jogo.

Caso seja a primeira sessão do usuário, o jogo o redireciona para a tela *Connect Scene*. Esta tela pede que o usuário forneça informações da sua conta no facebook para o jogo. Após a obtenção destes dados, ou caso não seja a primeira sessão do usuário, este é redirecionado para a tela *Main Menu*.

Na tela *Main Menu* o jogador pode tomar dois possíveis caminhos. O primeiro é a criação de um novo jogo. Caso este caminho seja o escolhido, o jogo é redirecionado para a tela *New Game Scene*. Nesta tela o usuário pode escolher jogar contra um oponente aleatório e ser redirecionado para a tela *Choice Scene* ou jogar contra um amigo do facebook. Neste caso o jogador será redirecionado para a tela *Friend Picker Scene*, que apresentará um menu com os amigos que ele pode desafiar, sendo posteriormente redirecionado para a tela *Choice Scene*. O segundo caminho possível a partir do *Main Menu* é responder um desafio. Neste caso o jogador será diretamente redirecionado para a tela *Choice Scene*.

Na tela *Choice Scene* o usuário sorteia um tipo de carta a ser jogado se ele é o desafiante, e caso contrário, caso seja o desafiado, apenas lhe é mostrado qual tipo de carta foi sorteada pelo desafiante. Qualquer que seja o papel do jogador, a partir deste ponto a próxima tela será a *Game Scene*.

A *Game Scene* é a tela em que são apresentadas as dicas ao jogador, que deve tentar acertar qual a personalidade da carta a partir das informações contidas nas dicas. Quando o usuário tenta responder, ele é finalmente redirecionado para a tela *End Game*, que mostra o resultado final, ou seja, se o jogador acertou a personalidade ou não, e qual a pontuação atingida.

A partir de então existem dois possíveis caminhos. Caso o jogador esteja respondendo um desafio, agora é a vez dele mandar um novo desafio para o oponente. Sendo assim, ele é então redirecionado para a tela *Choice Scene* para que seja sorteado um novo tipo de carta e seja dado prosseguimento ao jogo. Caso o jogador já esteja mandando um novo desafio para o oponente, ele é redirecionado para o *Main Menu*, dando fim ao fluxo do jogo.

6.2 Principais Classes

6.2.1 GameActivity

Em aplicações android o conceito de activity é de vital importância. Basicamente uma activity é um espaço na tela em que se pode desenhar qualquer tipo de aplicação, além de ser o ponto inicial de uma aplicação android.

Em Biografia, a única activity implementada chama-se *GameActivity*. Esta classe herda da classe *BaseGameActivity*, presente na andEngine, a capacidade de iniciar o jogo, instanciar os primeiros objetos, além de chamar a primeira cena do jogo.

6.2.2 ResourcesManager

A classe *ResourcesManager* é responsável por todos os recursos utilizados no jogo. Entende-se por recursos as imagens, as fontes e os sons utilizados no jogo.

ResourcesManager é uma classe singleton¹ que possui dois métodos para cada entidade do jogo que precisa instanciar recursos. Um método de *load* que carrega estes recursos na memória e os deixam prontos para serem utilizados, e um método de *unload* que libera a memória utilizada por estes recursos, tornando a utilização dos mesmos impossibilitada.

6.2.3 SceneManager

A classe *SceneManager* é responsável pela transição de cenas do jogo e também se trata de uma classe singleton.

É a classe *SceneManager* que executa toda a lógica da troca de cena. No escopo dessa classe existe um método *create* para cada cena presente no jogo. Este método deve ser chamado sempre que se deseja trocar a cena corrente por outra. Nele é instanciado um novo objeto que representa a nova cena a ser inserida no jogo, além de ser feita a chamada ao método *setScene* presente na classe Engine do jogo (andEngine), que é a responsável por desenhar a nova cena na tela de fato.

6.2.4 GameManager

O *GameManager* é uma classe singleton, que é responsável por guardar valores que classes sem relação precisam compartilhar, fazendo com que estes valores sejam buscados no servidor apenas uma única vez embora classes diferentes o utilizem. Alguns exemplos de valores guardados pelo GameManager são:

- *UserName*: Variável que guarda o nome do jogador.
- *userPictureURL*: Variável que guarda o link da foto do jogador.
- *friendPictureURL*: Variável que guarda o link da foto do oponente.

¹Padrão de projeto que garante a existência de apenas uma instância da classe

6.2.5 BaseScene

BaseScene é uma classe abstrata que representa uma cena genérica. No jogo todas as cenas herdam da classe *BaseScene*, o que uniformiza a implementação de qualquer trecho de código relacionado às cenas.

A *BaseScene* possui quatro métodos abstratos que devem ser implementados por qualquer classe que queira herdar suas configurações. São eles:

- *createScence*: Construtor da cena.
- *onBackKeyPressed*: Chamado quando o botão voltar, presente em todos os celulares android, é pressionado
- *getSceneType*: Método que retorna qual cena está sendo mostrada.
- *disposeScene*: Método chamado quando a cena será retirada da tela. É responsável por fazer a chamada ao método correspondente do *ResourcesManager* que libera a memória utilizada.

6.2.6 FacebookFacade

FacebookFacade é a classe responsável por abstrair as requisições de informações ao facebook. Se uma classe necessita de informações do facebook, ela precisa instanciar um novo objeto *FacebookFacade* e fazer a chamada ao método que executa a requisição desejada. Os métodos presentes na classe *FacebookFacade* são:

- *login*: Método que pede permissões para o compartilhamento de informações entre o facebook e o jogo.
- *getFriends*: Método que retorna a lista de amigos que possui o jogo
- *getUserPicture*: Método que retorna a url da foto do usuário cujo identificador foi passado como parâmetro.

6.3 Integração Facebook

A integração do facebook com o jogo Biografia fez com que o jogo se tornasse mais pessoal, e principalmente social. Através do compartilhamento de informações do facebook com o jogo, tornou-se possível mostrar informações como nome e foto do usuário durante o fluxo do jogo, além de ter sido de vital importância para fazer com que a opção multijogador fosse viável, através da busca pela lista de amigos desta rede social.

6.4 Novo Jogo

A partir do momento em que as informações do facebook do usuário são compartilhadas com o jogo, este pode iniciar novas partidas. Em Biografia existem duas maneiras de se começar um novo jogo. São elas:

6.4.1 Amigo do Facebook

Ao ser escolhida a opção de criar um novo jogo com amigos do facebook, é feita uma requisição ao mesmo para que sejam compartilhadas as informações sobre listas de amigos que possuem o aplicativo Biografia instalado. Após a requisição ser completada é mostrado ao jogador um menu, onde cada item corresponde a um amigo que está apto a ser escolhido para o início de um novo jogo.

6.4.2 Oponente Aleatório

A opção de oponente aleatório dá ao jogador a chance de poder iniciar uma nova partida contra qualquer outro jogador cadastrado na base de dados do Biografia. É feita uma requisição ao servidor que busca um oponente, de maneira aleatória, que o jogador ainda não tenha partida iniciada, devolvendo ao jogo as informações desse oponente para que a nova partida seja jogada.

Capítulo 7

Resultados Obtidos

Como resultado final do projeto foi desenvolvido um protocolo de comunicação entre o jogo e o Web Service bem definido e funcional, além do fluxo de telas, e a mecânica principal, envio e resposta dos turnos, totalmente implementados. Dentro da proposta de monografia, os principais itens propostos foram executados.



Figura 7.1: Principais telas do jogo

Capítulo 8

Conclusão

O escopo do trabalho proposto mostrou-se maior do que o inicialmente planejado, o que fez com que fosse necessário uma maior demanda de tempo e maior esforço durante o desenvolvimento.

Uma das propostas deste trabalho era a implementação de um projeto que utilizasse diversas tecnologias e áreas diferentes da computação. Durante o desenvolvimento do jogo foi exatamente o que aconteceu, o projeto final mistura várias ferramentas e conhecimentos de áreas distintas.

Quanto às ferramentas utilizadas, a andEngine se mostrou uma engine de jogos de duas dimensões muito adequada, com diversas funcionalidades e extensões úteis no desenvolvimento de jogos. O único ponto fraco é a falta de documentação, porém o fórum¹ e a grande ajuda do livro [Bro13] supriram boa parte dessa falta. Quanto às outras tecnologias utilizadas, todas se mostraram uma boa escolha para o desenvolvimento do projeto, em especial o ADT, que oferece grande número de ferramentas ao desenvolvedor.

¹Forum:<http://www.andengine.org/forums/>

Capítulo 9

Trabalhos Futuros

Este projeto pode ser melhorado e ampliado através de novas funcionalidades e melhorias. Algumas delas são:

- Imagens: De maneira geral as imagens do jogo podem ser melhoradas.
- Sons: Não foram implementados sons no jogo. Uma implementação seria uma grande melhora.
- Especiais: Implementar possíveis especiais a serem gastos durante o jogo. Algo como mostrar a primeira letra do nome da personalidade, por exemplo.
- Tela de compra: Implementar a economia do jogo, vendendo moedas que serviriam para a compra de especiais.
- Tela de Estatísticas: Implementar a tela no jogo que mostra as estatísticas persistidas na tabela do banco de dados `Historico_Estatistica`

Parte II
Parte Subjetiva

Capítulo 10

Subjetivo

10.1 Desafios e frustrações

Os principais desafios e frustrações encontrados durante o desenvolvimento deste projeto foram:

10.1.1 Não conhecimento das linguagens utilizadas

Uma das principais dificuldades no andamento do projeto foi justamente um de seus maiores objetivos: O aprendizado de linguagens que tive pouco contato durante a graduação. As duas principais linguagens utilizadas no projeto foram Java e PHP. Durante o desenvolvimento de um jogo relativamente extenso, o desconhecimento de estruturas e facilidades oferecidas pela linguagem são empecilhos que fazem com que tarefas simples demorem mais tempo do que o esperado.

10.1.2 Documentação da engine

A andEngine possui grande número de usuários, e um fórum bastante ativo, o que ajuda muito em momentos de dúvidas sobre como instanciar certos objetos e utilizar alguns de seus métodos. Porém, a falta de documentação faz com que o processo de implementação seja muito ligado a pesquisa sobre a engine. Este foi um ponto que fez com que atividades, que a princípio, supondo que existisse uma documentação da biblioteca, seriam rápidas, se tornassem dependentes de repetidas buscas e experimentos.

10.1.3 Interface gráfica

A total falta de conhecimento sobre ferramentas de modelagem gráfica foi outro ponto impactante no trabalho. Como a parte gráfica de um jogo é um fator determinante para seu sucesso, empreguei uma parte do tempo reservado ao trabalho aprendendo a utilizar o programa GIMP.

10.1.4 SDK Facebook

A principal frustração durante o desenvolvimento foi o fato de o facebook ter mudado sua política de compartilhamento de informações com aplicações. Quando este projeto foi proposto, no começo de 2014, era possível fazer com que através das informações de um usuário pudesse se chegar a toda a sua lista de amigos, fazendo assim com que o jogo rapidamente ganhasse maior número de usuários, pois pessoas que não tinham o jogo eram convidadas por pessoas que já o tinham através de novos jogos. A partir do dia 30 de abril desse ano isso não é mais possível, as requisições que antes traziam toda a lista de amigos passaram agora a responder apenas com os dados dos amigos que já tem o aplicativo instalado, reduzindo assim drasticamente o principal fator do jogo social que é a rápida disseminação na rede.

10.2 Disciplinas Relevantes para o desenvolvimento do trabalho

Praticamente todas as disciplinas cursadas durante a graduação foram de grande proveito intelectual e ajudaram muito na minha formação. Para o desenvolvimento desde trabalho especificamente se destacam:

10.2.1 MAC0122 Princípios de Desenvolvimento de Algoritmos

Esta matéria foi a primeira matéria de lógica mais aprofundada que tive. Foi MAC0122 que me fez começar a pensar como um programador, tomando conhecimento de algoritmos mais eficientes, clareza de código, como as estruturas eram guardadas na memória, enfim uma matéria básica sem a qual não seria possível desenvolver este projeto.

10.2.2 MAC0242 Laboratório de Programação II

Em MAC0242 foi a primeira vez que tive contato com o paradigma da orientação a objetos, além do que no projeto proposto para o semestre o meu grupo desenvolveu um pequeno jogo em Java. Esta foi uma disciplina que abriu a minha mente para a realização deste projeto.

10.2.3 MAC0439 Laboratório de Bancos de Dados

A matéria MAC0439 foi de vital importância, pois nela foi modelado um banco de dados real, que era utilizado pelo projeto do semestre. Nessa matéria tivemos a parte prática das peculiaridades do SQL, além de obter grande experiência prática com bancos de dados PostgreSQL, o que ajudou muito no desenvolvimento dessa parte do jogo Biografia.

10.2.4 MAC0448 Programação para Redes de Computadores

A importância da matéria MAC0448 está no aprendizado de conceitos sobre redes e servidores de diversos tipos, conteúdo até então totalmente desconhecido por mim, que me ajudou nas configurações do web service desse projeto.

O desenvolvimento do trabalho de conclusão do curso foi muito importante para mostrar o quanto os diferentes assuntos e áreas da computação estudadas durante a graduação de maneira separada podem na verdade ser complementares umas as outras. A base sólida conseguida através da graduação foi de vital importância na boa realização deste projeto.

Referências Bibliográficas

- [And13] Cupcake, donut, eclair, froyo, gingerbread, honeycomb, ice cream sandwich... google android os code names and history. <http://hubpages.com/hub/Cupcake-Donut-Eclair-Froyo-Gingerbread-Honeycomb-Android-OS-Version-Codenames-and-Why>, Setembro 2013. Acessado em 16 de outubro de 2014. 6
- [And14] Android: 85 <http://www.meioemensagem.com.br/home/marketing/noticias/2014/08/06/Android-atinge-85-por-cento-do-mercado-de-smartphones.html>, Agosto 2014. Acessado em 16 de outubro de 2014. v, 7
- [Bro13] Jayme Schroeder/Brian Broyles. *AndEngine for Android Game Development Cookbook*. Packt, 2013. 29
- [Cec05] Fábio Reis Cecin. Freemng: uma arquitetura cliente-servidor e par-a-par de suporte a jogos maciçamente distribuídos. Master's thesis, Instituto de Informática. Universidade Federal do Rio Grande do Sul, Março 2005. 5
- [dMGFN07] Alexandre Altair de Melo/Mauricio G. F. Nascimento. *PHP Profissional*. Novatec, segunda edição edição, 2007. 11
- [Hep13] Aden Hepburn. Infographic: 2013 mobile growth statistics. <http://www.digitalbuzzblog.com/infographic-2013-mobile-growth-statistics/>, Outubro 2013. Acessado em 16 de outubro de 2014. 7
- [Ind14] Bndes faz estudo sobre games no brasil para estimular indústria. <http://computerworld.com.br/negocios/2014/04/02/bndes-faz-estudo-sobre-games-no-brasil-para-estimular-industria/>, Abril 2014. Acessado em 20 de outubro de 2014. v
- [JSO] Introdução ao json. <http://www.json.org/json-pt.html>. Acessado em 15 de novembro de 2014. 17
- [Mer14] Mattia Mercato. A doce história do android. <http://www.androidpit.com.br/historia-do-android>, Junho 2014. Acessado em 14 de outubro de 2014. 6
- [Mil08] André Milani. *PostgreSQL Guia do Programador*. Novatec, 2008. 10
- [Ngi14] <http://nginx.org/en/>, Outubro 2014. Acessado em 19 de outubro de 201. 10
- [Nog14] Ana Clara Nogueira. Quantidade de celulares no mundo será maior que a de pessoas em 2015. <http://www.psaf.com/blog/mundo-tera-mais-celulares-pessoas-2015/>, Junho 2014. Acessado em 20 de outubro de 2014. v
- [Nun07] Maira Nunes. Social impact games: Uma nova possibilidade de comunicação. <http://www.bocc.uff.br/pag/nunes-maira-social-impact-games.pdf>, Julho 2007. Acessado em 27 de abril de 2014. 3