Universidade do Porto

FEUP Faculdade de Engenharia

Mestrado Integrado em Engenharia Informática e Computação

Sistemas Distribuídos

3º Ano 2ºSemestre

Jogo “Spy Counter Spy”

Relatório

27 de Maio de 2012

Ana Luísa Pires Magalhães Marques – 080509156

Carlos Eduardo Mesquita Frias – 040509116

Paulo Jorge de Faria dos Reis – 080509037

Pedro Miguel de Oliveira Ferreira – 090509120

Introdução

No âmbito da disciplina de Sistemas Distribuídos realizamos este trabalho prático, que tem como objetivo desenvolver uma aplicação para Android baseada em REST.

O propósito da nossa aplicação é implementar um jogo online multiplayer com localização GPS, que baseado na proximidade geográfica entre jogadores permite um grau de interatividade variável, para ser jogado num dispositivo móvel Android.

Cada jogador será representado por um avatar que possuirá um conjunto personalizável de itens (armas de ataque e armas de defesa), que tem influência na jogabilidade em termos de ataques e defesas. Estes itens são obtidos pelo jogador mediante o número de pontos que este possui. Os pontos são conquistados com o evoluir do jogo, consoante as ações que o jogador vá realizando ou opcionalmente em troca de valor monetário, isto é, compra de pontos (no caso de uma eventual rentabilização económica do produto).

Assim sendo, implementamos as funcionalidades necessárias recorrendo à linguagem Java e utilizando as API's do Google Maps.

Neste relatório descrevemos de forma mais pormenorizada a aplicação, apresentamos a arquitetura da mesma e indicamos as funcionalidades implementadas, descrevendo assim o trabalho realizado.

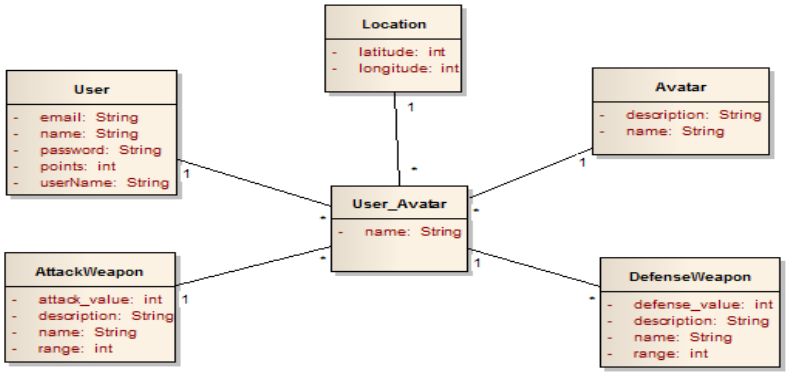
Arquitetura da Aplicação

A aplicação está logicamente dividida em 3 módulos distintos: Base de dados, WebServer e Android. Iremos fazer uma breve descrição da forma como cada módulo está implementado assim como a forma como estes interagem entre si. Descreveremos também o site elaborado, para servir de apoio à aplicação.

1. Base de Dados

A base de dados que serve de suporte a toda a aplicação está implementada em Mysql e encontra-se alojada num servidor Linux. Esta Base de Dados está configurada de forma a ser possível efectuar alterações às suas tabelas do exterior e

não apenas da máquina onde está alojada. O diagrama seguinte ilustra as tabelas existentes assim como a sua relação:



- a tabela User guarda toda a informação acerca dos utilizadores registados no jogo necessária ao seu login e pontos acumulados;
- as tabelas AttackWeapon e DefenseWeapon são analogas e guardam um conjuntos de armas de ataque e defesa que um utilizador poderá utilizar ao longo da sua experiencia no jogo.;
- a tabela Avatar armazena um pequeno conjunto de figuras que irão identificar o utlizador no mapa de jogo;
- a tabela User_Avatar é uma tabela que relaciona todas as anteriores e define o estado de um utilizador num dado momento: as suas armas, a sua localização e o seu Avatar.

2. WebServer

O WebServer utilizado foi desenvolvido em Java usando as funcionalidades de apoio e teste existente no IDE NetBeans e disponibiliza serviços com base na arquitectura Rest (Representational State Transfer) que tem como um ponto forte a forma simples como é possível aceder a recursos utilizando apenas identificadores universais (URI's). A escolha do NetBeans para o desenvolvimento e teste do WebServer surge de forma natural dado o grande número de funcionalidades e interfaces que este disponibiliza de forma automática no processo de criação de um servidor que tem como ponto de partida uma Base de Dados.

Estrutura:

O WebServer tem dois pacotes de classes distintos: entidades e serviços.

As entidades são representações das tabelas existentes na base de dados. Cada tabela irá ter correspondência a uma classes diferente que servirá de ponte entre os dados na base de dados e os objetos a serem criados e manipulados pelo WebServer.

Os serviços contêm uma classe por cada entidade. Nestas classes são criados por defeito serviços básicos de inserção, seleção e alteração de dados (PUT , GET e POST). A estes serviços podem ser atribuídos caminhos que serão posteriormente concatenados ao seu caminho universal (URI) que será utilizado pelos clientes para obter informação. Sendo que os serviços básico criados são claramente insuficiente um grande número de novos serviços foram desenvolvidos pela equipa de desenvolvimentos. Apresentamos em seguida, para exemplo, um desses serviços que tem como objetivo de criação/registo de um novo utilizador

```
@POST
@Produces(
   {
    "application/xml", "application/json"
   })
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public void newPlayer(
      @FormParam("id") int id,
      @FormParam("name") String name,
           @FormParam("userName") String userName,
           @FormParam("email") String email,
      @Context HttpServletResponse servletResponse
) throws IOException {
   User c = new User(id);
     c.setName(name);
     c.setUserName(userName);
     c.setEmail(email);
   super.create(c);
}
```

Como é possível observar, são inicialmente definidos que tipo de parâmetros são necessários para o serviço e que tipo de informação será devolvida ao cliente. Fazendo uso das entidades criadas, é criado um objecto do tipo User e definidos os seus parametros . Posteriormente é chamada um classe que criará este registo na base de dados.

3. Android

A aplicação Android, isto é o jogo em si, foi desenvolvido em linguagem Java no IDE Eclipse. A escolha deste IDE deveu-se à facilidade de instalação do Android SDK nesta plataforma e também da enorme quantidade de tutoriais existentes para desenvolvimento Android neste IDE.

Estrutura:

A aplicação Android está distribuída por três pacotes de classes distintos: com.sdistp2.spycounterspy, entidades e rest.

No pacote com.sdistp2.spycounterspy estão as classes java responsáveis por cada uma das Atividades da aplicação Android, a saber:

- LoginActivity: utilizada para efectuar o login no jogo;
- EquipActivity: onde o utilizador procede à escolha da sua arma de ataque, arma de defesa e tipo de avatar (imagem a utilizar). Cada arma de ataque e defesa têm um custo associado que é descontado no número de pontos do utilizador;
- ViewAvatarAct: onde o utilizador pode visualizar o seu avatar e armas escolhidas com respetiva descrição, se não gostar dos itens escolhidos pode voltar à atividade anterior e efetuar nova escolha, sem penalização no número de pontos;
- ViewMap.java: que permite visualizar o mapa já com todos os utilizadores online colocados na respetiva posição, colocando o mapa centrado na posição do próprio, com um zoom definido por defeito. Existe a possibilidade de alternar o tipo de mapa, entre modos “satélite” e “não satélite”. Também é possível alterar o zoom e deslocar o mapa com o simples toque com o dedo.

- ItemsOnMap.java: subclasse de ItemizedOverlay<OverlayItem> e permite colocar "objetos clicáveis" no mapa. É usada para colocar na posição respetiva cada um dos utilizadores online no mapa, apresentado em ViewMap;
- ViewEnemyAct.java: atividade que permite observar alguns dos dados do inimigo (ou do próprio utilizador, se for clicado o próprio OverlayItem), nomeadamente, o nome, o número de pontos, a defesa, descrição da defesa e distância ao inimigo. A distância ao inimigo é dada pelo WebService disponibilizado pelo Google para o efeito. Se o número de pontos do próprio utilizador e do inimigo for maior que zero e se a distância ao inimigo for inferior ao alcance da arma de ataque um botão "Atacar" fica disponível. Esse botão desencadeia uma ação de ataque que provoca alterações no número de pontos dos dois utilizadores. A diferença entre o número pontos da nossa arma de ataque e da arma de defesa do inimigo reverte favoravelmente ou desfavoravelmente, consoante o caso, para o total de pontos de cada utilizador;
- ViewAttackAck: esta atividade mostra o ataque, isto é, apresenta uma pequena imagem a simular o ataque e apresenta os resultados do mesmo, ou seja, o número de pontos com que cada um dos utilizadores ficou.

No pacote entidades foram criadas classes para guardar as informações localmente que vão sendo carregadas de e para a base de dados e carregadas entre atividades. Nomeadamente:

- Avatar: onde é guardada informação sobre cada utilizador, nomeadamente: nome, tipo, pontos, localização geográfica, item de ataque e item de defesa escolhidos;
- Item: utilizada para guardar informação sobre itens (ou armas), nomeadamente: nome, descrição e custo;
- AttackItem: subclasse de Item para guardar o id da arma de ataque, o número de pontos de ataque e o alcance (em km);
- DefenseItem: subclasse de Item para guardar o id da arma de defesa, o número de pontos de defesa e o alcance (em km);

Finalmente o pacote rest contém apenas a classe RestClass que contém uma série de métodos responsáveis pelos pedidos REST: os pedidos por GET são efetuados pela função callWebService(String pedido) que retorna a resposta em XML e pedidos POST são efetuados pela função httpPost(String pedido, String[] paramName, String[] paramVal) que devolve também o resultado XML e leva como argumentos URL do pedido, um array de Strings com os parâmetros e outro com os valores desses parâmetros. Os restantes métodos são responsáveis pelo parser das diversas respostas XML e retorno dos objetos pretendidos. Todos estes métodos são estáticos para que possam ser chamados a qualquer momento de qualquer uma das atividades sem necessidade de criação de um objecto RestClass.

4. Site

Foi construído um site para a aplicação com objetivo de servir de apoio à restante estrutura. Nele os jogadores poderão realizar o seu registo na Base de Dados da aplicação, e terão acesso a uma loja onde poderão adquirir (comprar) mais pontos para o seu Avatar.

Destas duas funcionalidades apenas o registo de novos jogadores ficou a funcionar.

Funcionalidades Implementadas

As funcionalidades implementadas foram as seguintes:

- registo do utilizador: esta funcionalidade permite ao jogador realizar o seu login e logout da aplicação;
- escolha das suas armas: esta funcionalidade permite ao jogador escolher as armas de defesa e de ataque que o seu avatar possuirá;
- visualização das caraterísticas do respetivo Avatar: esta funcionalidade permite que o jogador observe as caraterísticas do seu Avatar, antes de avançar para o terreno de jogo.
- visualização do mapa: esta funcionalidade permite observar ao jogador visualizar ver a sua posição no mapa assim como a dos seus inimigos.

- visualização das características do inimigo: esta permite observar as características de um inimigo, de forma a decidir se o pretende atacar.
- atacar um inimigo: esta funcionalidade permite ao jogador realizar uma ação de ataque sobre um outro jogador.

Conclusão

Este trabalho prático teve como principal objetivo desenvolver uma aplicação para Android baseada em REST. A aplicação desenvolvida não cumpre todos os objetivos a que nos propusemos, pois por limitações de tempo não conseguimos implementar todas as funcionalidades propostas, mais concretamente a publicação dos principais eventos no Facebook.

Consideramos que o resultado é positivo, uma vez que as funcionalidades essenciais foram implementadas com sucesso e que as dificuldades com que nos fomos deparando no decorrer do projeto foram superadas.

No que se refere à distribuição de tarefas e ao envolvimento dos elementos do grupo nas mesmas, estas foram divididas da seguinte forma:

Ana Luísa Marques - apoio na realização de tarefas do módulo Android, apoio na realização de tarefas da construção do Site e responsável pela elaboração do relatório (20%);

Carlos Frias - responsável pelo módulo Android (30%);

Paulo Reis - responsável pelo Site (20%);

Pedro Ferreira - responsável pelo módulo Base de Dados e WebServer (30%).