Relazione progetto
Reti di calcolatori
GameServer

Gianluca Lutero
Nicola Mazzara
Filippo Soncini

2 marzo 2016

Indice

Introduzione

Il progetto realizzato tratta lo sviluppo di un videogioco su piattaforma *Android* che utilizza come interfaccia una mappa, ottenuta da *Google Map*, dove vengono visualizzati i giocatori con i quali si può interagire. In questa relazione verrà prima descritto l'obiettivo dello stesso, il progetto nelle sue componenti di client e server e i protocolli di rete adottati.

1 Obiettivi

L'obiettivo che ci siamo prefissati è stato quello di realizzare un server di gioco che mascherasse la sua struttura interna e in grado di servire più client indipendentemente dal tipo di dispositivo. Per questa ragione si è scelto di utilizzare il protocollo http, descritto nella sezione dei protocolli, per la comunicazione client-server e una servlet per realizzare l'interfaccia del server.

2 Descrizione del gioco

2.1 Client

Il client di gioco è realizzato su piattaforma Android. Il gioco prevede inizialmente una fase di *login* dopo la quale si accede alla fase di scelta del personaggio e infine alla fase di gioco vera e propria. Di seguito sono descritte le varie funzionalità.

2.1.1 Login e Registrazione

L'applicazione prevede una fase di login per accedere al gioco. Una volta effettuato il login il giocatore sceglie il personaggio da usare o ne crea uno nuovo. Se il giocatore non è in possesso di credenziali valide deve effettuare la procedura di registrazione attraverso l'apposita sezione in cui deve fornire una email e una password che lo identificheranno nell'applicazione.

2.1.2 Creazione del personaggio

Una volta premuto il bottone di *creazione nuovo personaggio* sarà possibile selezionare uno tra i possibili personaggi disponibili e assegnargli un nome che lo identifica.

2.1.3 Interfaccia di gioco

Dopo la scelta del personaggio viene mostrata la mappa alle coordinate del giocatore più gli altri utenti attivi in quel momento. Sulla mappa il giocatore seleziona il personaggio che vuole attaccare.

2.1.4 Scontro tra i giocatori

Lo scontro avviene selezionando il personaggio bersaglio e cliccando il bottone apposito. Se il personaggio viene sconfitto questi viene rimosso dalla mappa e al giocatore vengono assegnati 5 punti esperienza.

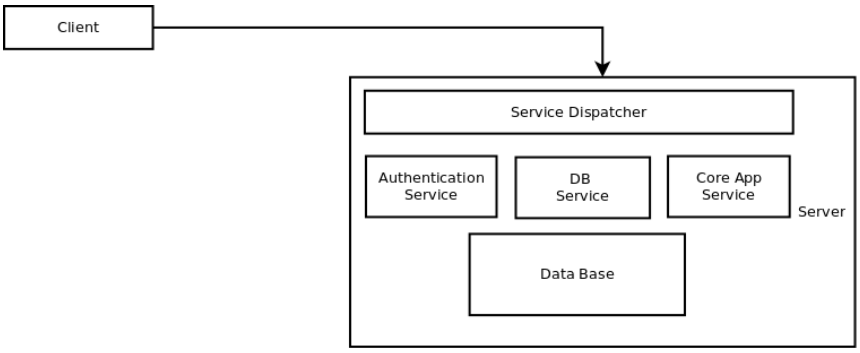
2.2 Server

Il server di gioco è realizzato attraverso una `Servlet` in Java. Il server riceve le richieste `Http` del client. Il client invia come parametro il servizio richiesto dal server. Di seguito è descritta la struttura e le funzionalità del server.

2.2.1 Struttura

Il server è formato dal *dispatcher*, che fa da interfaccia con i client, dall' *authentication service*, che gestisce la fase di autenticazione e registrazione dell'utente, dal *database service*, che gestisce la comunicazione con il database, e dal *coreapp service*, che gestisce le funzionalità proprie del gioco, e dal database.

Nella seguente figura è mostrata la struttura:



2.2.2 Dispatcher

La parte del server che fa da interfaccia con il client. Il componente riceve le richieste dai client, controlla che siano ben formate e le inoltra ai servizi sottostanti. Il componente attende le risposte dai servizi sottostanti e le restituisce ai client.

2.2.3 Authentication service

La parte del server che fornisce il servizio di autenticazione e registrazione. Se le credenziali sono esatte l'utente viene inserito nella lista degli utenti attivi e viene generato per esso un id di sessione che gli verrà restituito.

2.2.4 Database service

Servizio che fa da interfaccia con il database sottostante.

2.2.5 Coreapp service

Implementa tutte le funzionalità lato server dell'applicazione.

3 Ptotocolli di rete utilizzati

3.1 Http

HTTP (protocollo di trasferimento di un ipertesto) è usato come principale sistema per la trasmissione di informazioni sul web. Un server HTTP generalmente resta in ascolto delle richieste dei client sulla porta 80 usando un protocollo TCP a livello di trasporto. L'HTTP funziona su un meccanismo richiesta/risposta (client/server): il client esegue una richiesta e il server restituisce la risposta.
Il messaggio di richiesta è composto da tre parti:

- riga di richiesta (request line)
- sezione header (informazioni aggiuntive)
- body (corpo del messaggio)

Il messaggio di risposta è di tipo testuale ed è composto da tre parti:

- riga di stato (status-line)
- sezione header
- body (contenuto della risposta)

Client è server comunicano solo attraverso questo protocollo, in particolare si sfruttano alcuni parametri,descritti in seguito, che generano il linguaggio di comunicazione tra i due.

3.1.1 Specifiche dei parametri

Il client seleziona i servizi nel server attraverso il parametro `service`. Questo parametro può assumere i seguenti valori:

- `signin`: Invoca il servizio di registrazione. Devono essere forniti come parametri l'email e la password.
- `login`: Invoca il servizio di login. Devono essere forniti come parametri l'email e la password, se l'autenticazione ha avuto successo il giocatore viene aggiunto a quelli attivi.

- `logout`: Invoca il servizio di logout. Deve essere fornita come parametro l'email del giocatore
- `create`: Invoca il servizio di creazione del personaggio. Deve essere fornito l'email del giocatore che lo crea, il nome e la classe del personaggio.
- `get_players`: Invoca il servizio di richiesta dei personaggi di un giocatore. Deve essere fornita l'email del giocatore.
- `near player`: Invoca il servizio di richiesta dei giocatori vicini. Deve essere fornito come parametro il nome,la classe, la posizione, l'esperienza e il livello del personaggio che viene inserito tra quelli attivi e poi restituito insieme alla lista dei personaggi.
- `logout player`: Invoca il servizio di logout del personaggio. Deve essere fornito il nome del personaggio.
- `attack`: Invoca il servizio con cui un personaggio attacca gli altri personaggi. Deve essere fornito il nome dell'attaccante e il nome del difensore.

3.2 Algoritmo crittografico SHA-1

L'algoritmo sha-1(Secure Hash Algorithm) genera un *message digest* di lunghezza fissa partendo da un messaggio di lunghezza variabile.

3.2.1 Funzionamento dell'algoritmo

Passo 1 (Imbottitura): Al messaggio originale vengono aggiunti dei bit di "imbottitura" affinché la lunghezza finale del messaggio risulti congruente a 448 modulo 512, così facendo la lunghezza di "messaggio+imbottitura" è pari ad un numero 64bit più piccolo di un multiplo di 512bit.
Passo 2 (Aggiunta lunghezza): Alla sequenza di bit (messaggio+imbottitura) creata durante il passo 1 viene aggiunto un intero unsigned di 64bit contenente la lunghezza del messaggio originale. Alla fine di questi due primi passi otteniamo una sequenza di bit che è un multiplo di 512.
Passo 3 (Inizializzazione del buffer MD): Un buffer di 160bit suddiviso in 5 registri da 32bit ciascuno viene creato per la memorizzazione di alcuni

passaggi intermedi. I 5 registri verranno convenzionalmente indicati con (A,B,C,D,E) ed inizializzati con i seguenti valori esadecimali:

- A = 67452301
- B = EFCDAB89
- C = 98BADCFE
- D = 10325476
- E = C3D2E1F0

Passo 4 (Elaborazione dei blocchi da 512bit): La sequenza di bit "messaggio+imbottitura+lunghezzaMessaggio" viene divisa in blocchi da 512bit, che identificheremo con Bn con n che va da 0 a L. Il fulcro dell'algoritmo SHA-1 è chiamato compression function ed è formato da 4 cicli di 20 passi cadauno. I cicli hanno una struttura molto simile tra di loro se non per il fatto che utilizzano una differente funzione logica primitiva. Ogni blocco viene preso come parametro di input da tutti e 4 i cicli insieme ad una costante K e i valori dei 5 registri. Alla fine della computazione otterremo dei nuovi valori per A,B,C,D,E che useremo per la computazione del blocco successivo sino ad arrivare al blocco finale F. Nel progetto l'algoritmo viene utilizzato per criptare le password degli utenti, che varranno salvate nel database in questa forma, usando la libreria Java `MessageDigest`.
Di seguito il codice:

```
private String cripta(String s) throws NoSuchAlgorithmException{
      MessageDigest mess = null;
        mess = MessageDigest.getInstance("SHA-1");
        mess.reset();
        try {
        mess.update(s.getBytes("UTF-8"));
      } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
      }

        byte[] dig = mess.digest();

        StringBuffer sb = new StringBuffer();
```

```
        for (int i = 0; i < dig.length; i++) {
          sb.append(Integer.toString((dig[i] & 0xff) + 0x100,
              16).substring(1));
        }

       System.out.println(sb.toString());

       return sb.toString();
 }
```

3.3 Login

L'utente si autentica nel server invocando il servizio di login e fornendo una email e una password. Il server cerca l'utente nel database e recupera la password criptata associata. A questo punto viene criptata, con algoritmo *sha-1*, la password fornita dall'utente e confrontata con quella del database, se l'operazione ha successo viene generato un id di sessione e restituito all'utente altrimenti viene restituito un messaggio d'errore.
Di seguito il codice che effettua il login:

```
public boolean loginUtente(String email,String password){

    String control = null;

    Connection connection = null;
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    try{
      connection = databaseConnect();
      statement = connection.prepareStatement("SELECT * FROM utenti
          WHERE email = ?;");
      statement.setString(1, email);
      resultSet = statement.executeQuery();

      resultSet.next();
```

```
      control = resultSet.getString("pass");

      System.out.println("DB:"+control);
      System.out.println("USER:"+cripta(password));

      if(control.equals(cripta(password))){
        return true;
      }

    }catch(SQLException s){
      databaseDisconnect(connection, statement, resultSet);
      s.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
      }

    return false;

  }
```

3.4 Sicurezza delle operazioni

3.4.1 Sicurezza del protocollo Http

Il problema nell'utilizzo del protocollo http sta nel traffico dati in chiaro, inoltre non viene garantita ne l'identità del client ne del server. Per ovviare a questa mancanza può essere usata la versione sicura di http chiamata https che sfrutta un canale cifrato, mediante protocollo SSL, per la comunicazione. Il protocollo https garantisce l'integrità dei dati, l'identità del client e del server.

3.4.2 Sicurezza del protocollo di login

Il protocollo di login utilizzato impone che le password vengano salvate e confrontate in versione criptata attraverso l'algoritmo sha-1. Inoltre a login avvenuto l'utente ottiene l'id di sessione per garantire la legalità dell'operazione.

3.4.3 Sicurezza del server

Il server è strutturato in modo tale da avere un'interfaccia con l'esterno, il dispatcher. che ne maschera la struttura interna e controlla che le richieste siano ben formate. Inoltre nel server il database service maschera e gestisce le richieste al database.

4 Sviluppi futuri e conclusioni

4.1 Sviluppi futuri

Vista la struttura del server i possibili sviluppi futuri riguardano maggiormente nuove funzionalità implementate nel gioco, come nuove classi o aggiunta dell'inventario. Per quanto riguarda la struttura stessa del server i possibili sviluppi futuri riguardano:

- Utilizzare il protocollo `HTTPS` anzichè `HTTP` per rendere sicura la comunicazione tra client e server, falla attuale del progetto
- Implementare il logout automatico dei giocatori inattivi
- Controllare la validità delle email fornite per garantire l'identità dell'utente
- Funzionalità di recupero password dimenticate
- Maggiore controllo sulle operazioni effettuate dall'utente per evitare il fenomeno di cheating

4.2 Conclusioni

Il progetto sviluppato ci ha portato ad affrontare le problematiche legate alla gestione di un server di gioco real-time, per il quale si è dovuto trovare prima di tutto soluzioni per gestire i vari client che condividevano i dati di gioco e rendere persistenti le loro operazioni. Inoltre si è dovuto trovare il modo per mascherare la struttura interna del server per mantenere la compatibilità tra i vari client.

4.3 Riferimenti esterni

Indirizzo del repository:https://github.com/GianlucaLutero/ProgettoRetiAndroid
Host del server: https://www.openshift.com/