Henry Lod

Monialustainen-laivanupotuspeli

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

Insinöörityö

27.9.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Henry Lod Monialustainen-laivanupotuspeli 40 sivua 27.9.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Jorma Räty Juha Kämäri

Insinöörityön tavoitteena oli kehittää Monialustainen-laivanupotuspeli, jota voidaan pelata Android-puhelimella ja tietokoneella. Lisäksi työntekijä halusi oppia tuntemaan paremmin Monialustainen-ympäristön ja sen haasteet.

Insinöörityössä toteutettiin toimiva peli. Tarvittavat sovellukset tehtiin Netbeans- ja Eclipse Java – sovelluksilla. Yhteyden luontiin käytettiin Socket-rajapintaa. Pelissä pelaajat luovat yksilölliset pelitilit PHP-sivujen kautta. Sivuilla pelaajat voivat nähdä listan parhaista pelaajista ja vaihtaa oman pelitilinsä salasanan.

Tässä insinöörityössä käydään läpi, miten ympäristö toteutettiin, mitä osia siihen tarvittiin ja miten eri laitteet saatiin toimimaan yhdessä. Lisäksi työssä esitellään lyhyesti käytetyn ohjelmaympäristön ominaisuuksia yleisesti.

Monialustainen ympäristö on järjestelmäriippumaton ympäristö missä ohjelma voi toimia erilaisten laitteiden kanssa yhdessä. Esimerkiksi sovellus voisi toimia puhelimien ja tietokoneiden välillä.

Monialustainen osoittautui mielenkiintoiseksi ja haastavaksi ohjelmistoympäristöksi, minkä takia projektissa otettiin huomioon jatkokehitysmahdollisuudet. Tietokanta ja ohjelmaympäristö on tehty niin, että sinne voidaan jatkossa lisätä myös muita pelejä.

Avainsanat	Java, Android, Monialustainen, Socket

Author(s) Title Number of Pages Date	Henry Lod Cross-platform Battleship game 40 pages 27 September 2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Jorma Räty Juha Kämäri

The goal in this project was to create cross-platform battleship game, which can be played on Android phones and computers. Also I wanted to learn more about the cross-platform environment and the challenges it gives.

In this project a working game was created. Game was created on NetBeans and Eclipse developing tools. Socket API was used to create the connection between devices. Players create their own gaming identities on the PHP-page at the game. Players can also see the highscore list of the player's statics and change their passwords on the webpage.

In this project we go through how to create cross-platform battleship game environment. Project goes through how the environment is setup, what parts do we need to run it and how does the connection work between all the devices.

Cross-platform environment isn't tied to one kind device and can work between different kinds of devices. Example between computers and phones.

Cross-platform environment turned out to be exciting and challenging. Future development of the software is considered in the project. The database and the software are build that way so in future it is possible to add more games to the platform.

Keywords	Java, Android, Cross-platform, Socket

Sisällys

Lyhenteet

HTML	Standardoitu kuvauskieli verkkosivujen luontiin.
CSS	Tyyliohje, joka on kehitetty erityisesti verkkosivuja varten.
SQL	Kyselykieli tietokantojen hallintaa varten.
PHP	Ohjelmointikieli erityisesti dynaamisten verkkosivujen luontia varten.
Java	Laitteistoriippumaton olio-ohjelmointikieli.
Socket	Internetprotokollayhteys, jolla voidaan yhdistää erilaisia laitteita yhteen.
MD5	Algoritmi, jota käytetään esimerkiksi salasanojen salaamiseen.
Injektio	Järjestelmään tunkeutumista, jossa käyttäjä ajaa omaa koodia ohjelman sekaan.
XML	Extensible Markup Language eli merkkauskieli tiedon esittämiseen.
Palvelin	Laite, johon käyttäjät yhdistyvät ja joka kuljettaa tietoa käyttäjiltä toisille.
Apache	HTTP-palvelinohjelma, jonka avulla voidaan ajaa PHP koodia ja lähettää sähköposteja.
Parametri	Tieto, jonka käyttäjä tai sovellus antaa.
Relaatiotietokanta	Tietokanta, jossa taulut viittaavat muihin tauluihin.
Validointi	Tarkistetaan verkkosivujen standardienmukaisuus.

Olio-ohjelmointi	Ohjelmointitapa, jossa ohjelman osat koostuvat olioista.
Olio	Ohjelmoinnissa pitää sisällään tietoa ja metodit.
Cross-platform	Monialustainen ympäristö.

1 Johdanto

Tässä insinöörityössä tehtiin projektityö omaan käyttöön. Tarkoituksena oli luoda peliympäristö laivanupotuksen pelaamiseen tietokoneiden ja Android-puhelimien välille. Android-käyttöjärjestelmästä on käytetty versiota 1.0. Eri laitteiden takia pelin tulee olla alustariippumaton. Ympäristössä tulee ottaa huomioon, että peli toimii samalla tavalla kaikilla laitteilla. Yksinkertainen tiedonsiirto on tärkeää, koska puhelimien prosessorit ovat paljon heikompia kuin tietokoneiden, ja pelin tulee toimia sulavasti kaikilla alustoilla.

Työssä valitsin peliksi laivanupotuksen, koska sen avulla voidaan esitellä helposti eri alustojen välistä viestintää. Ohjelmointikielenä projektissa käytän Java-ohjelmointikieltä. Android- ja tietokonelaitteet molemmat tukevat Java-sovelluksia, minkä ansiosta koko projekti pystyttiin luomaan Javalla.

Työn alussa kuvaillaan ympäristöä, jossa tehdyt ohjelmistot toimivat ja tutustutaan SQL:ään, salausmenetelmiin, tietokantoihin, Javaan, Apacheen, palvelimiin, Socketiin, PHP:hen, HTML:ään, CSS:ään ja algoritmeihin yleisesti.

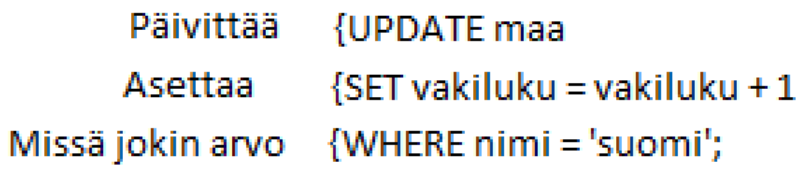
Insinöörityössä esitellään, kuinka tällainen peliympäristö luodaan ja miten saadaan ylläpidettyä yhteys kaikkien laitteiden kanssa samanaikaisesti. Lisäksi käydään läpi, miten ohjelmoija voi vaikeuttaa pelaajia huijaamasta ja käyttämästä omia koodinpätkiä. Projektissa selviää Android-puhelimien ja tietokonesovelluksien eroavaisuudet ja samanlaisuudet. Lisäksi työssä käydään läpi, kuinka tärkeitä algoritmit ovat tietokonesovelluksissa ja kuinka niiden avulla ohjelma saadaan toimimaan huomattavasti nopeammin kuin sovellus joka ei käytä algoritmeja tai käyttää erittäin huonoja algoritmeja.

2 Ohjelmistoympäristö

2.1 SQL

SQL (Structure Query Language) on tietokantojen kyselykieli. Se on standardoitu kyselykieli, minkä IBM on kehittänyt 1970-luvulla [1]. SQL:llä voidaan tehdä erilaisia hakuja, muutoksia, poistoja ja lisäyksiä relaatiotietokantaan. Projektissa relaatiotietokannassa pidetään käyttäjien tiedot sekä salasana MD5-salattuna. Projektissa SQL:llää käytetään tilastojen päivittämiseen ja pelaajan sisäänkirjautumisen tarkistamiseen. Lisäksi käyttäjillä on mahdollisuus muuttaa salasanaa. Myös tulosten päivittäminen tapahtuu SQL-kyselyjen kautta.

SQL-käskyllä on aina tietty rakenne. Tärkeimmät näistä ovat SELECT, UPDATE, INSERT ja DELETE.



Kuva 1. SQL-kyselyn rakenne [2].

Kyselyn alkuosa kertoo aina, mitä tullaan tekemään ja mihin tauluun kysely tapahtuu. Tämän jälkeen kyselyssä voidaan asettaa muutoksia tai hakea vain tietoa. Lopuksi voidaan vielä tarkentaa kyselyä, jotta se täyttää tietyt kriteerit. Kuvassa 1 oleva kysely tekee päivityksen maa-taulukkoon, jossa vakiluku solua kasvatetaan yhdellä vain, jos nimi on suomi. SQL-kyselyitä käyttäessä tulee aina varmistaa, että käyttäjä ei pysty vaikuttamaan suuresti kyselyn rakenteeseen. Jos käyttäjä voi vapaasti muokata kyselyä, voi hän hakea käyttäjätaulusta kaikkien käyttäjien tiedot itselleen tai vaikka tuhota koko tietokannan. Ohjelmissa kannattaa siis olla valmiit kyselyrungot, joihin ohjelma lisää käyttäjän tiedot ja suorittaa kyselyn niiden perusteella. Näin vältetään tietojen päätyminen vääriin käsiin.

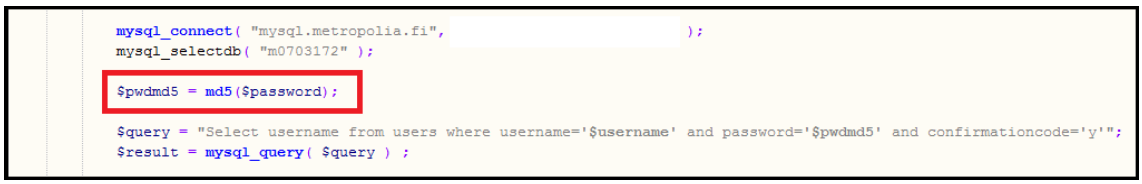
2.2 Salausmenetelmät ja MD5-salaus

MD5 on algoritmi, jota käytetään esimerkiksi kryptografiassa[17]. Sitä käytetään salaamaan tietoa. Itse MD5:n käyttämä algoritmi ei kuitenkaan ole salainen. Huolimatta julkisesta algoritmista ei kertaalleen salattua tietoa pystytä algoritmin avulla muuttamaan takaisin alkuperäiseen muotoon. Tämä johtuu siitä, että MD5 tiivistää tietoa. MD5:llä salattu tieto on 128-bittinen tiiviste, joka sisältää 32 merkkiä. Koska syntyvä merkkijono on niin lyhyt, on teoriassa mahdollista, että tiivistyksessä kahdesta erilaisesta lausekkeesta syntyy identtinen tiivistelmä. Tämä on kuitenkin varsin teoreettinen mahdollisuus, minkä takia se ei tässä insinöörityössä ole ongelma. Kuitenkin, jos käsiteltäisiin jotakin arkaluontoisempaa tietoa, kuten esimerkiksi verkkopankkitunnuksia, olisi syytä käyttää monimutkaisempaa salausta.

MD5 ottaa salauksessa huomioon isot ja pienet kirjaimet, numerot, lisämerkit ja merkkien järjestyksen. Salauksella syntyvä tiivistelmä on tästä syystä täysin erilainen esimerkiksi sanoilla ”koira” ja ”Koira”. Yhden merkin muutos ei siis muuta vain yhtä osaa salauksesta, vaan se muuttaa koko salauksen.

PHP:lla MD5-salaus tapahtuu yhdellä rivillä koodia (kuva 2). Tästä koodirivistä saatava hyöty verrattuna käytettyyn aikaan on suuri, minkä takia MD5:n käyttöä voidaan pitää kannattavana. MD5-salattujen tietojen alkuperäistä viestiä on lähes mahdotonta selvittää, vaikka se on teoriassa mahdollista.

Salauksen murtaminen tehdään kokeilemalla. Salauksen murtaja salaa ensin valitsemiaan sanoja ja katsoo, millaisen merkkijonon nämä sanat salauksessa muodostavat. Sen jälkeen tulee etsiä tietokannasta, sisältääkö se saatua salausta vastaavaa merkkijonoa. Mikäli merkkijono löytyy, on salaus tämän yksittäisen sanan osalta saatu murrettua. Kokonaisen tietokannan salauksen murtaminen yllä kuvatulla tavalla on kuitenkin varsin vaikeaa, minkä takia MD5-salausta voidaan yleisesti pitää turvallisena.

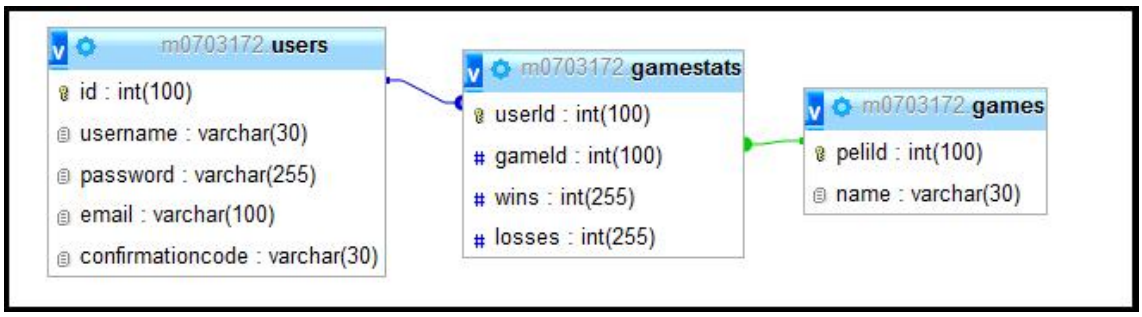


Kuva 2. MD5-salaus PHP:ssa.

Kaikissa tilanteissa MD5-salaus ei ole kuitenkaan tarpeeksi vahva. On olemassa paljon tehokkaampia salausmenetelmiä kuten RSA, joka on 1024-bittinen salaus [3]. RSA perustuu salaamiseen alkulukujen kautta. Siinä missä MD5 käyttää salaukseen viestin muuntamista tiivisteeksi, RSA:ssa salaus luodaan tekemällä algoritmi, jossa käytetään suurta alkulukua. Alkulukujen jaksollisuutta ei tiedetä, joten koneellisesti arvaamalla on lähes mahdotonta selvittää, miten salaus on luotu. RSA-salauksen luonti on hieman hitaampaa kuin MD5:n, minkä takia tietyissä tilanteissa kannattaa valita kevyempi salausmenetelmä. Projektissa päätin, että MD5-salaus olisi tarpeeksi tehokas suojaamaan pelitilin salasanat.

2.3 Tietokanta

Tietokanta koostuu tauluista, joissa taulujen sisälle on luotu soluja. Nämä solut pitävät sisällään tiedot. Relaatiotietokannassa taulujen tiedot on yhdistetty toisiinsa taulun avaimella. Yleensä yhdistämiseen käytetään ID-solua. Yhteyksien käyttäminen helpottaa tietokantojen käyttöä ja sen muokkaamista.



Kuva 3. Tietokantasolujen yhteys.

Projektissa käyttämässämme tietokannassa on kolme taulua, joissa tiedot on järjesteltynä niin, että jokaisen käyttäjän omat tiedot pysyvät erossa muista (kuva 3). Ensimmäinen users-taulu pitää sisällään käyttäjien tiedot ja tiedon siitä, onko tili vahvistettu sähköpostilla. Taulu on linkitetty käyttäjän ID:llä gamestats-tauluun. Tässä taulussa on tallennettu ID:n mukaan tieto siitä, missä pelissä käyttäjällä on minkäkin verran häviöitä ja voittoja. Lisäksi on viimeinen, ylimääräinen taulu, jossa ID kertoo pelin nimen. Tämä taulu on luotu sitä varten, että ohjelmaan voidaan myöhemmin helposti lisätä eri pelejä. Pelien luonti kantaan onnistuu helposti, koska jokaisen pelin tiedot ovat omassa taulussaan.

2.4 Java

Insinöörityön koko toiminnallinen puoli on luotu Javalla. Java on oliopohjainen ohjelmointikieli. Olio-ohjelmoinnilla tarkoitetaan sitä, että ohjelmoijan luodessa koodia hän voi luoda olioita. Oliot ovat sekä toiminnallisia että tietoa säilyttäviä rakenteellisia perusyksiköitä. Oliot voivat pitää sisällään tiettyjä tietoja ja erilaisia funktioita. Olioiden luonnin jälkeen ohjelmoija pystyy käyttämään valmiiksi luomiaan oliota ja antamaan niille sopivia arvoja.

Java-ohjelmointikieli kehitettiin 1990-luvun alussa [4]. Tavanomaiset ohjelmointikielet, kuten C++ ja C, kääntävät ohjelmointikielen konekieleksi. Konekieleksi kääntäminen mahdollistaa ohjelman käytön. Java kääntää koodinsa tavukoodiksi, jonka virtuaalikone suorittaa. Tästä syystä Java ei pysty vaikuttamaan suoranaisesti muihin tietokoneen prosesseihin toisin kuin jotkin tavanomaiset ohjelmointikielet.

```
import java.net.Socket;
import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;
import javax.swing.Box;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import javax.swing.text.DefaultEditorKit;

/**
 *
 * @author m0703172
 */
public class WebClient {
```

Kuva 4. Javan valmiita kirjastoja insinöörityössä.

Java on ohjelmointikielenä yksinkertainen ja käytännöllinen, koska sen toimintaympäristö on erittäin laaja. Javalla voidaan tehdä ohjelmia niin tietokoneille kuin puhelimillekin ja Java onkin yksi käytetyimmistä ohjelmointikielistä [5]. Javassa on sisäänrakennettu roskienkeruu, joka vapauttaa käyttämättömiä muistipaikkoja. Tämän ansiosta Javalla koodatut ohjelmat vievät käyttölaitteelta vähemmän muistia ohjelman käytön aikana eikä muistiylivuotoja pääse syntymään niin helposti.

Java sisältää paljon valmiita kirjastoja, joita ei muissa ohjelmointikielissä ole valmiina. Näitä ovat esimerkiksi valmiit verkko-ominaisuudet, graafiset käyttöliittymäkirjastot sekä suuri määrä rajapintoja. Oracle Corporation kehittää Java-ympäristöä koko ajan laajemmaksi ja luo siihen lisää valmiita osia ja kirjastoja ohjelmoijia varten.

Uutta ohjelmaa aloittaessa koodaaja valitsee, mitä valmiita kirjastoja hän haluaa käyttää. Valmiiden kirjastojen käyttöönotto on Javassa helppoa. Java-luokan alkuun kirjoitetaan import, minkä jälkeen annetaan halutun kirjaston sijainti ja nimi kuten kuvassa 4. Javassa ei ole rajoitettu käytettävien kirjastojen määrää. Koodaaja valitsee haluamansa kirjastot, että ohjelman suorittamisesta ei tule turhien kirjastojen takia raskaampaa kuin on pakko.

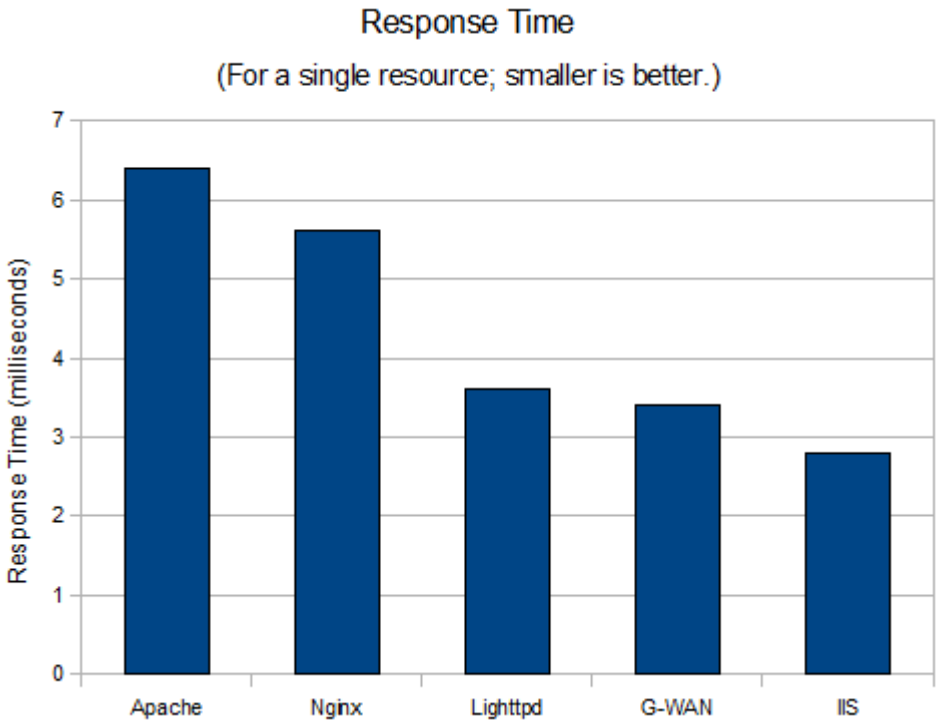
Valitsin Javan koodikieleksi projektia varten, koska se on alustariippumaton. Tämän ansiosta voidaan luoda yhtenäisiä koodinpätkiä, jotka toimivat samalla tavalla molemmissa käytetyissä alustoissa. Olin myös aikaisemmin tehnyt Java-sovelluksia Androidille, joten Javan käyttö tuntui luontevalta.

2.5 Apache

Apache on verkkopalvelimilla toimiva palvelinsovellus [6]. Apache on yksi markkinoiden suosituimmista palvelinsovelluksista. Apache alustana palvelimella. Se voi kääntää erilaisia ohjelmistokieliä, joita voidaan käyttää hyväksi luodessa verkkosivuja. Insinöörityön verkkosivujen palvelimella on käytössä Apache, jotta PHP-koodit saadaan käännettyä. Apache tukee virtuaalipalvelintoimintoa. Tämä tarkoittaa sitä, että yksi palvelin voi toimia samanaikaisesti useana eri palvelimena, joista jokainen voi toimia eri asetuksilla.

Apache perustuu avoimeen lähdekoodiin, minkä ansiosta palvelimen ylläpitäjät voivat muokata palvelimen toiminnallisuuksia vapaasti. Apachessa ovat käytössä suuret API-kirjastot, jotka tarjoavat sovelluskehittäjälle hyvät työkalut.

Apachen toiminta on kilpailukykyistä muihin palvelinsovelluksiin nähden, vaikka se ei ole markkinoiden nopein[7] (Kuva 5). Kilpailevia sovelluksia ovat mm. IIS (Microsoft) ja GWS (Google). Apache käyttää moniydintekniikkaa. Moniydintekniikalla Apache pystyy suorittamaan ajon aikana monia toimintoja samanaikaisesti.



Kuva 5. Apachen vasteaika muihin palvelimiin verrattuna [8].

2.6 Alustat

Insinöörityön alustoina toimivat Android-älypuhelin ja tietokone Windows-käyttöjärjestelmällä. Tässä insinöörityössä sovellus on suunniteltu sellaiselle Android-pohjaiselle älypuhelimelle, joka toimii kosketusnäytöllä. Kosketusnäyttö tarkoittaa sitä, että näyttö reagoi käyttäjän kosketukseen ja suorittaa toimintoja sen kautta. Tietokonetta voidaan ohjata perinteisesti hiirellä ja näppäimistöllä tai kosketusnäytöllä. Luotaessa ohjelmaa, joka toimii erilaisten alustojen välillä, tulee huomioida laitteiden väliset käyttöerot. Pelin perusideana on, että kaikilla käyttäjillä on samat mahdollisuudet voittaa.

Käyttöliittymän tulee siis olla mahdollisimman samanlainen ja toiminnallisuuden sama riippumatta käyttölaitteesta.

Alustoilla voi olla teknillisiä rajoituksia ja nämä pitää huomioida luotaessa ohjelmia. Tekniset rajoitukset voivat laiteriippumattomissa ohjelmistoissa estää kokonaan luomasta joitakin osia, jos niiden luonti edes yhdellä alustalla on mahdotonta. Tässä insinöörityössä yksi tärkeimmistä asioista on se, että kaikilla laitteilla on tarpeeksi tilava ruutu näyttää kaikki sovelluksen osat. Android-ohjelmissa voi olla eri näkymä, jos käytettävä laite on pysty- tai vaaka-asennossa. Tämä vaikuttaa olennaisesti ruudulla käytettävissä olevaan tilaan. Tarvittaessa sovellus voidaan lukita tiettyyn asentoon niin, että laitteen kääntäminen ei vaikuta sovellukseen.

Android-ympäristön loi Android Inc, jonka Google osti myöhemmin [9]. Android koostuu käyttöjärjestelmästä ja muista järjestelmäkirjastoista. Android on Linux-pohjainen ja käyttää ohjelmointikielenä Javaa. Androidissa käytetty Java-versio on erikseen räätälöity Androidille, eikä se sisällä kaikkia kirjastoja, joita täydet Java ME tai SE sisältävät. Android-ohjelmat koostuvat Java-koodista ja XML-merkintäkielestä. Android suorittaa ohjelmakoodin Dalvik-virtuaalikoneella samalla tavalla kuin Java suorittaa ohjelmakoodinsa [10].

```
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;



public class GameActivity extends Activity {
```

Kuva 6. Insinöörityössä käytettyjä Android-kirjastoja.

Google tarjoaa Android-ympäristössä erilaisia valmiita API-kirjastoja, joita ohjelmoija voi käyttää sovelluksessaan (kuva 6). Lisäksi Android sisältää muutaman valmiin kirjaston ohjelmointikielistä C ja C++. Android-laitteessa jokaisella sovelluksella on oma ympäristö ja virtuaalikone ajon aikana. Android ei automaattisesti sulje käynnistettyjä sovelluksia, vaan käyttäjän tulee sulkea ne manuaalisesti.

Android-sovellukset koostuvat aktiviteeteista. Aktiviteetit muodostavat kasan Android-ohjelman ajon aikana. Riippuen ohjelmasta käyttäjä voi siirtyä edes takaisin aktiviteettien välillä, jos niitä ei ole lopetettu siirtyessä uuteen aktiviteettiin. Jokainen aktiviteetti on itsenäinen osa, ja jos ohjelma tarvitsee toisessa aktiviteetissä jotakin muuttujaa vanhasta aktiviteetistä, se täytyy siirtää uuteen aktiviteettiin sitä luotaessa. Androidilla on myös käytössä oma Google Play -kauppa, johon ohjelmoijat voivat syöttää omia sovelluksiaan. Tätä kauppaa voi käyttää hyväkseen jos haluaa kaupallistaa omia ohjelmia.

Android-puhelimien lisäksi on olemassa muitakin älypuhelimia kuten iPhone ja Windows Phone. iPhone ja Windows Phone ovat samantyylisiä älypuhelimia kuin Android-puhelimet. Puhelimissa ja sovellustentekomahdollisuuksissa on kuitenkin eroavaisuuksia. iPhoneissa on erittäin rajattu ympäristö omien ohjelmien ajamiselle. Kaikki iPhone-sovellukset pitää hyväksyttää Applella. Vasta hyväksytysten jälkeen ne voidaan ladata Apple Storeen ja omaan puhelimeen. Hyväksyttämisprosessi hidastaa ja vaikeuttaa ohjelman toteuttamista ja testaamista, sillä todellisuudessa ohjelmat toimivat oikealla puhelimella eri tavalla kuin virtuaalipuhelimen testiympäristössä. Projektissa valitsin Android-puhelimen, koska sen ympäristö on merkittävästi vapaampi verrattuna iPhoneen. Android-puhelimia on myös markkinoilla enemmän kuin Windows Phoneja ja halusin ohjelmani olevan käytettävissä mahdollisimman monelle käyttäjälle [11].

Android-ohjelmissa tiedot ovat ohjelmoijalla kahdessa paikassa. Ohjelmoijalla on src-kansio, jossa projektin Java-luokat sijaitsevat. Tämän lisäksi on layout-kansio, jossa sijaitsevat Android-aktiviteettien käyttöliittymänäkymät. Lisäksi ohjelman kansiossa on AndroidManifest.xml-tiedosto, jolla luodaan projektin rakenteelle viittaukset ja tiedot siitä, mitä oikeuksia Android-sovellus tarvitsee (kuva 7).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.battleships"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.battleships.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.battleships.ChatActivity"
            android:label="@string/chat_activity"
            android:parentActivityName="com.battleships.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.battleships.MainActivity" />
        </activity>
         <activity
            android:name="com.battleships.GameActivity"
            android:label="@string/chat_activitygame"
            android:parentActivityName="com.battleships.ChatActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.battleships.ChatActivity" />
        </activity>

    </application>

</manifest>
```

Kuva 7. Android manifest.

Esimerkiksi, jos ohjelmoija ei aseta sovellukselle oikeuksia käyttää internetiä, kaikki internetkomennot saavat sovelluksen kaatumaan. Manifest-luokassa kerrotaan, mikä ohjelman Java-luokista aukeaa, kun ohjelma käynnistetään. Lisäksi manifest-luokassa on tieto siitä, mitkä kaikki luokat kuuluvat ohjelmaan.

Layout.xml-tiedostot kertovat, millainen mistäkin aktiviteetin käyttöliittymästä tulee (kuva 8). Layout-tiedostossa luodaan tiedot siitä, mihin kohtaan mikäkin sovelluksen osa tulee ja mitä fontteja ja kokoja käytetään.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@layout/activity_chat"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="1.12" >

        <ListView
            android:id="@+id/list"
            android:layout_width="80dp"
            android:layout_height="match_parent"
            android:textSize = "20sp"
            android:cacheColorHint="#00000000"
            android:listSelector="@android:color/transparent"
            android:transcriptMode="alwaysScroll" >
        </ListView>

        <ListView
            android:id="@+id/list2"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:textSize = "20sp"
            android:cacheColorHint="#00000000"
            android:listSelector="@android:color/transparent"
            android:transcriptMode="alwaysScroll" >
        </ListView>

    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="fill_parent"
```

Kuva 8. Activity_chat xml-layout.

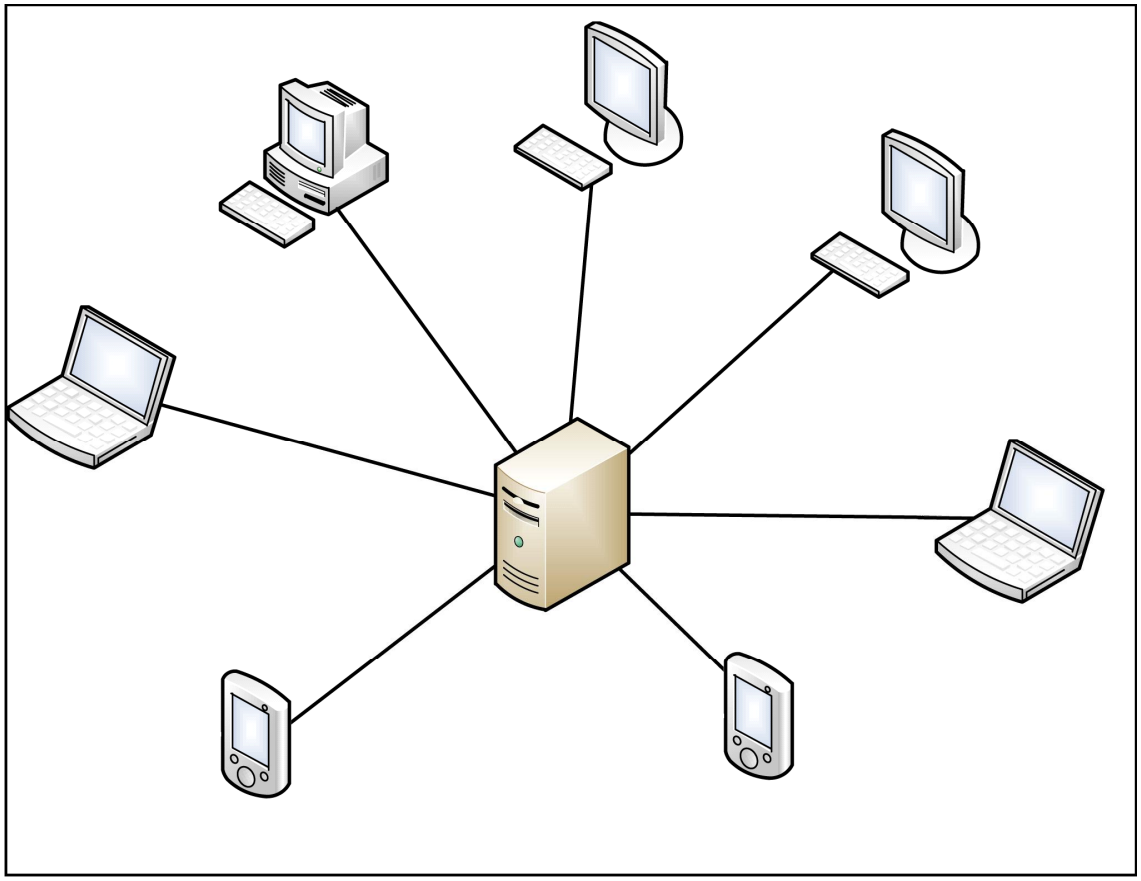
2.7 Palvelin ja Soketit

Insinöörityössä käytetään palvelinta. Palvelimen tarkoituksena on tarjota ympäristö pelaajille. Palvelimena voi toimia mikä tahansa tietokone, johon käyttäjät voivat muodostaa yhteyden. Yleensä palvelimina kuitenkin käytetään tietokoneita, jotka on valmistettu palvelinkäyttöön.

Palvelimen kokoonpano muistuttaa normaalia tietokonetta, mutta komponenttien täytyy olla kestävämpiä kuin tavallisissa tietokoneissa. Palvelimien komponentit ovat yleensä paljon kalliimpia, koska niiltä odotetaan parempaa suorituskykyä, vähemmän kaatumisia tai muita virheitä. Palvelimella täytyy olla erittäin nopea verkkokortti ja korkea sisään- ja ulostulevan liikenteen suorituskyky.

Monissa palvelimissa on niin sanottu hotswap-järjestelmä, jolla palvelimen osia voidaan vaihtaa lennosta sammuttamatta palvelinta. Palvelimen käynnistäminen kestää yleensä kymmeniä minuutteja. Ennen käynnistymistä palvelin tarkistaa kaikki toiminnot ja aukaisee etäkäyttömahdollisuudet. Tämän jälkeen uudelleenkäynnistystä ei tarvita pitkään aikaan.

Yleensä palvelimet ovat käytössä 24 tuntia vuorokaudessa, 7 päivänä viikossa - toisin sanoen koko ajan joka päivä. Palvelimet eivät saa kuumentua, minkä takia ne on yleensä sijoitettu kokonaan omaan tilaansa. Palvelintilassa pitää olla hyvä ilmanvaihto, jotta palvelimet saadaan pidettyä viileänä. Palvelimissa on tyypillisesti myös isot ja tehokkaat tuulettimet, jotka pitävät meteliä. Erillinen palvelintila auttaa paitsi pitämään palvelimet toimintakykyisinä myös estää tuulettimista syntyvän melun leviämistä.



Kuva 9. Palvelimen yhteyskaavio.

Insinöörityössä jokainen pelaaja luo yhteyden palvelimeen Javan Socketin API:n avulla. Kuva 9 havainnollistaa, kuinka eri laitteet ovat yhteydessä samaan palvelimeen. Socket luo käyttäjälle sisään- ja ulosmenevät yhteydet viestien lähetystä ja vastaanottamista varten. Nämä yhteydet ovat valmiina Socket API:ssa.

Yhteys luodaan käyttäjän ja palvelimen välille IP-osoitteen ja portin avulla. Socket API on käytössä Android-puhelimissa ja Javassa. Socketin muodostama yhteys on tarpeeksi kattava tätä insinöörityötä varten, koska sillä saadaan suoritettua kaikki tarvittava tiedonsiirto. Socket-yhteyden luonti tehdään aina Javassa try-catch-rakenteella, kuten kuvassa 10 on kuvattu. Tällä varmistetaan, että ohjelma ei kaadu, jos yhteyden muodostaminen ei onnistu, vaan se luo tapahtuneesta error-ilmoituksen.

```
public void run() {

    connected = true;

    try {
        //Sets the server address
        InetAddress serverAddr = InetAddress.getByName(SERVERIP);

        //create a socket to make the connection with the server
        Socket socket = new Socket(serverAddr, SERVERPORT);
```

Kuva 10. Socket-yhteyden luonti.

2.8 PHP

Insinöörityössä on käytössä PHP-pohjaiset internetsivut. PHP on komentosarjakieli, jota käytetään nettisivujen toiminnallisuuksien rakentamiseen [12]. PHP-koodi pysyy piilossa käyttäjältä ja jättää näkyviin vain HTML-pohjaiset sivut. PHP suorittaa koodin palvelimella, kun käyttäjä avaa nettisivun. PHP toimii erinomaisesti tietokantojen kanssa, koska sillä voidaan suoran suorittaa SQL-kyselyitä. PHP suorittaa koodiaan vain omien rajamerkkiensä sisällä. PHP:n syntaksi on lähellä C- ja Java-kieltä. PHP:n uusimmat versiot tukevat myös oliopohjaista ohjelmointia kuten Java. Palvelin, joka ylläpitää internetsivuja tarvitsee PHP-kääntäjän, jotta se pystyy näyttämään PHP-sivut oikein. PHP-koodia käytetään yleensä HTML-koodin kanssa sivujen tekemisessä. PHP:lla kirjoitetaan ne osat, jotka halutaan pitää käyttäjältä piilossa. PHP-koodi ei siis näy sivun lähdekoodissa.

2.9 HTML ja CSS

HTML on kuvauskieli. Tätä kuvauskieltä käytetään erityisesti verkkosivujen ulkoasun luontiin. HTML-koodi koostuu elementeistä. Jokainen elementti pitää sisällään tietoa, joka tullaan näyttämään käyttäjälle sivulla. Elementtien alussa ja lopussa on tunniste, joka kertoo, mitä tämä elementti tulee tekemään. Tunnisteet voidaan jakaa kolmeen ryhmään: rakenteelliseen, esitykselliseen ja hypertekstuaaliseen ryhmään.

Rakenteelliset tunnisteet kertovat esimerkiksi, mikä elementeistä on otsikko ja mikä kirjautumisikkunan ruutu. Esitykselliset tunnisteet voivat muokata tekstin ulkoasua ku-

ten kokoa, väriä tai muita ulkoasuun vaikuttavia tekijöitä. HTML-sivujen ulkoasut luodaan erikseen yleensä CSS:llä. Hypertekstuaaliset tunnisteet toimivat linkkeinä toisille sivuille.

HTML-koodin kirjoittaminen onnistuu millä tahansa tekstieditorilla, mutta periaatteessa koodin pitäisi aina läpäistä HTML-validointi. HTML-validoinnilla tarkoitetaan sitä, että sivujen koodillinen rakenne läpäisee tietyt standartoidut tyylit. Sivujen validointi onnistuu suoraan erilaisten validointisivujen kautta. Yleisin näistä on W3C-validointisivu. W3C-sivu lukee sille annetun verkkosivun lähdekoodin ja tekee siitä johtopäätökset, läpäiseekö sivu validoinnin. Validointisivut antavat yleensä myös ilmoituksen jokaisesta kohdasta, joka ei läpäise validointia ja kertoo ohjelmoijalle, miten tämän voisi korjata.

Java-sovelluksien ajo onnistuu esimerkiksi HTML-sivuilla. Sovellukselle luodaan tunniste ja annetaan oikeat parametrit niin, että sovellus aukeaa tämän jälkeen käyttäjälle internetsivun sisällä. Koska HTML-koodi näkyy aina käyttäjälle sivun lähdekoodissa, sitä ei kannata käyttää sisäänkirjautumisiin tai muihin tärkeisiin tai arkaluontoisiin operaatioihin. Arkaluontoiset tiedot kuten salasanat voisivat muutoin paljastua. Tästä syystä PHP:ta käytetään hyvin usein HTML-sivujen sisällä luodessa sisäänkirjautumisikkunoita tai muita elementtejä.

CSS (Cascading Style Sheets) on tyyliohje, jota erityisesti verkkosivut käyttävät. Tyyliohjetta käytettäessä tehdään css-tiedosto, joka toimii sivun tyyliohjeena. Tiedostoon annetaan säännöt siitä, kuinka dokumentti tulee näyttää käyttäjälle. Säännöt eivät ole ehdottomia ja HTML-sivulla pitää aina elementissä viitata siihen, mitä osaa tyyliohjeesta halutaan käyttää. Tyyliohjeiden käyttö helpottaa verkkosivujen luontia. Vaikka HTML-sivulla voitaisiin tehdä samanlaisia tyylejä, on ohjelmoijan kannalta parempi käyttää tyyliohjetta. Tyyliohjeen avulla, jos ohjelmoija haluaa muokata koko verkkosivujen ulkoasua, hänen tarvitsee vain muuttaa tyyliohjeen sisällä olevia arvoja. Ilman tyyliohjetta tyylit olisivat koodattu HTML-sivulle suoraan ja muutoksia tehdessä pitäisi ohjelmoijan muuttaa tyylit erikseen jokaiselta koodiriviltä.

CSS-tyylejä käytettäessä voi ilmetä ongelmia. Kaikki nykyiset selaimet eivät tue CSS-tyylejä samalla tavalla, ja tämä voi johtaa siihen, että sivujen ulkoasu näkyy käyttäjälle väärin. Tämä pystytään kiertämään sillä, että tehdään jokaiselle selaimelle omat tyyliohjeet ja sivulle saapuessa käyttäjälle näytetään juuri oikea tyyliohje. CSS:n käytöllä on muitakin etuja kuin helposti muokattava verkkosivujen ulkoasu. Tyyliohjeen käyttö

nopeuttaa käyttäjän verkkoselailua, koska käyttäjän saapuessa sivulle lataa selain tyyliohjeen selaimen välimuistiin. Tällöin liikuttaessa sivulla tyyliohje on koko ajan ladattu eikä käyttäjän tarvitse ladata näitä tietoja joka kerta uudestaan liikkuessaan sivuston sisällä.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="styling.css">
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title>Henrys Gamesite - Log In</title>
    </head>

    <body>
        <form id='login' action='login.php' method='post' accept-charset='ISO-8859-1'>
            <label for='username' >UserName:</label><br/>
            <input type='text' name='username' id='username' maxlength="50" /><br/>
            <label for='password' >Password:</label><br/>
            <input type='password' name='password' id='password' maxlength="50" /><br/>
            <input type='submit' name='Submitl' value='Submit' />
        </form>
        <a href="forgotpass.php">Forgot Password</a><br>
        <a href="index.html">Return</a>


    </body>

</html>
```

Kuva 11. HTML-koodinäkymä.

Kuvista 10 ja 11 voidaan huomata, että PHP-koodi ei näy verkkosivun lähdekoodissa. Kaikki PHP:n sisällä olevat funktiot pysyvät käyttäjältä piilossa.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="styling.css">
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title>Henrys Gamesite - Log In</title>
    </head>

    <body>
        <form id='login' action='login.php' method='post' accept-charset='ISO-8859-1'>
            <label for='username' >UserName:</label><br/>
            <input type='text' name='username' id='username' maxlength="50" /><br/>
            <label for='password' >Password:</label><br/>
            <input type='password' name='password' id='password' maxlength="50" /><br/>
            <input type='submit' name='Submitl' value='Submit' />
        </form>
        <a href="forgotpass.php">Forgot Password</a><br>
        <a href="index.html">Return</a>


    </body>

</html>

<?php
        function Login(){
            if(empty($_POST['username'])){
                echo("UserName is empty!");
                return false;
            }

            if(empty($_POST['password'])){
                echo("Password is empty!");
                return false;
            }

            $username = trim($_POST['username']);
            $password = trim($_POST['password']);
            if(!isset($_SESSION)){ session_start(); }
            session_destroy();

            mysql_connect( "mysql.metropolia.fi", "user", "pass" );
            mysql_selectdb( "database name" );

            $pwdmd5 = md5($password);
            echo $pwdmd5;
            $query = "Select username from users where username='$username' and password='$pwdmd5' and confirmationcode='y'";
            $result = mysql_query( $query ) ;
```

Kuva 12. PHP-sivun koodi.

2.10 Algoritmit

Algoritmit ovat matematiikasta tuttuja [3]. Algoritmeja käytetään ohjelmoinnissa sovellusten nopeuttamista varten. Algoritmeja voidaan käyttää moniin erilaisiin tarkoituksiin. Yleensä algoritmeja käytetään tietojen hakuun tai lajitteluun. Projektissa käytetään algoritmia laivojen asettamiseen pelilaudalle ja tarkistettaessa, onko laivan kaikkiin osiin osunut. Tämä on kuvattu kuvassa 13.

```
private void checkIfSunken(int r, int s){
    if(r!=0 && logic.grid[r-1][s]!=0 && logic.grid[r-1][s]!=2){       //Move upwards 1
        moveDirection(r-1,s,1);
    }
    else if(s!=0 && logic.grid[r][s-1]!=0 && logic.grid[r][s-1]!=2){  //Move left 2
        moveDirection(r,s-1,2);
    }
    else if(r!=9 && logic.grid[r+1][s]!=0 && logic.grid[r+1][s]!=2){  //Move downwards 3
        moveDirection(r+1,s,3);
    }
    else if(s!=9 && logic.grid[r][s+1]!=0 && logic.grid[r][s+1]!=2){  //Move right 4
        moveDirection(r,s+1,4);
    }
}

private void moveDirection(int r, int s, int direction){
    if(direction==1  && r!=0){
        if(logic.grid[r-1][s]!=0 && logic.grid[r-1][s]!=2)
            moveDirection(r-1,s,1);
        else
            shipMemorize(r,s,3);
    }
    else if(direction==1  && r==0)
        shipMemorize(r,s,3);
    else if(direction==2 && s!=0){
        if(logic.grid[r][s-1]!=0 && logic.grid[r][s-1]!=2)
            moveDirection(r,s-1,2);
        else
            shipMemorize(r,s,4);
    }
    else if(direction==2 && s==0)
        shipMemorize(r,s,4);
    else if(direction==3 && r!=9){
        if(logic.grid[r+1][s]!=0 && logic.grid[r+1][s]!=2)
            moveDirection(r+1,s,3);
        else
            shipMemorize(r,s,1);
    }
    else if(direction==3 && r==9)
        shipMemorize(r,s,1);
    else if(direction==4 && s!=9){
        if(logic.grid[r][s+1]!=0 && logic.grid[r][s+1]!=2)
            moveDirection(r,s+1,4);
        else
            shipMemorize(r,s,2);
    }
    else if(direction==4 && s==9)
        shipMemorize(r,s,2);
}
```

Kuva 13. Osuman tarkastusalgoritmi.

3 Pelin osat

3.1 Laivanupotus

Laivanupotus on kahden pelaajan peli, jossa päämääränä on upottaa toisen pelaajan laivat ensimmäisenä. Ensimmäinen markkinoille tullut laivanupotuspeli ilmestyi vuonna 1931, vaikka pelin ensimmäistä versiota oli pelattu jo ensimmäisestä maailmansodasta aikaan Ranskassa [13]. Pelistä on nykyisin monia erilaisia versioita, joissa joko laivat ovat isompia tai pelialue on suurempi. Alkuperäisessä pelissä (kuva 14) ruudukon koko oli 10 x 10 ja laivoja oli neljää erilaista kokoa 2-5 ruudun kokoisia.

Insinöörityössä käytettävässä versiossa pelialueena toimii 10 x 10 -ruudukko, jossa vaakariveillä ovat kirjaimet A-L ja pystyriveillä numerot 1-10. Molemmilla pelaajilla on yhteensä viisi laivaa. Laivat ovat 2-5 ruutua pitkiä ja kummallakin pelaajalla on sama määrä samankokoisia laivoja. Laivojen upottaminen tapahtuu vuoron perään sokko-ammunnalla. Aina, kun pelaaja osuu laivaan, ruudun väri muuttuu kertoakseen mahdollisesta osumasta.

Kuva 14. Alkuperäinen Laivanupotus [14].

Projektissa laivoina ovat yksi 5 ruudun kokoinen, yksi 4 ruudun kokoinen, kaksi 3 ruudun kokoista ja yksi 2 ruudun kokoista laivaa. Tämä on alkuperäisen laivanupotuksen mukainen määrä laivoja.

3.2 Projektin rakenne

Pelin ympäristö on tehty Netbeans-ohjelmointiympäristöllä. Netbeans on ohjelmointiympäristö, joka tukee monia eri ohjelmistokieliä [15]. Android-puoli pelistä on tehty Eclipse-ohjelmistoympäristössä, ja sen rinnalle on ladattu Android SDK plugin. Insinöörityön verkkosivut on kirjoitettu Notepad++-tekstieditorilla.

Insinöörityön palvelin koostuu yhdestä luokasta nimeltään ChatServer. Ennen kuin mitään muita osia projektista halutaan käyttää, palvelin käynnistetään. Pelin muut osat käyttävät palvelinta kaikkeen tiedon välitykseen.

Tietokonepuolella peli koostuu viidestä luokasta: GameLogicista, HighscoreUpdaterista, ShipSetterista, ShipSinkerista ja WebClientista kuten kuvasta 15 voi huomata.

Kuva 15. Projektin rakenne.

Pelin pääluokkana tietokoneella toimii WebClient-luokka, joka luo yhteyden ChatServer-luokkaan. WebClient pitää sisällään kaksi käyttöliittymää: chat- ja peli-liittymät (kuvat 18 ja 19). Se pitää sisällään ChatAccess-luokan, joka ylläpitää Socket-yhteyttä palvelimeen. Lisäksi WebClient-luokka pitää sisällään tiedon siitä, mistä pelin logiikka löytyy. Pelin logiikka on rakennettu kolmesta eri luokasta: GameLogicista, ShipSetteristä ja ShipSinkeristä.

```
public class GameLogic {
    protected int[][] grid;
    protected int[][] opponentsGrid;
    private ShipSinker sinker;
    private ShipSetter setter;

    public GameLogic(){
        grid = new int[10][10];
        opponentsGrid = new int[10][10];
        sinker=new ShipSinker(this);
        setter=new ShipSetter(this);
    }
```

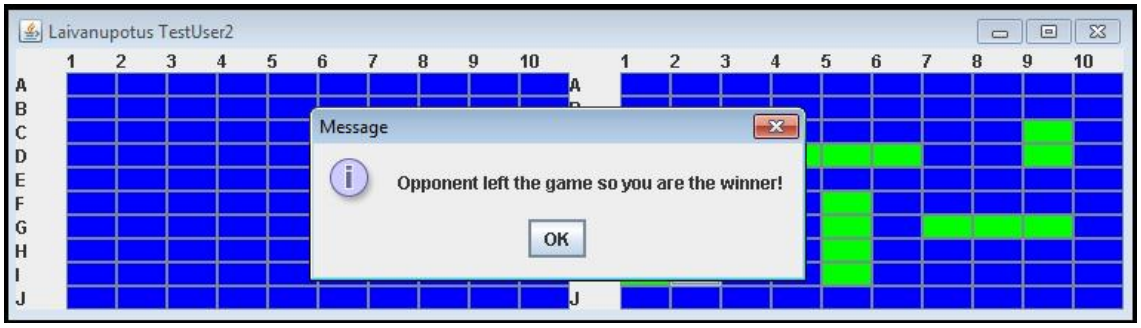
Kuva 16. Peliruudukon luominen.

GameLogic-luokka luo pelaamista varten kaksi taulukkoa (kuva 16), joissa pidetään yllä omien ja vastustajien laivojen sijainnit. GameLogic-luokka luo myös ilmentymän laivan asettajaluokasta ShipSetter ja laivan upottajaluokasta ShipSinker. Pelilogiikan tehtävänä on pitää huoli pelin kulusta ja siirtojen oikeellisuudesta. WebClient hoitaa ruudun päivittämisen GameLogic-luokan arvojen mukaan. ShipSetter-luokan tehtävänä on asettaa laivat pelilaudalle, kun uusi peli luodaan (kuva 17). Se arpoo jokaiselle laivalle paikan pelilaudalla ja tarkistaa, ettei mikään laivoista mene toistensa päälle. HighscoreUpdater päivittää pelin lopuksi SQL-tietokantaan pelitulokset.

```
private void setLineAndRow(){
    line=(int)(Math.random()*10);
    row=(int)(Math.random()*10);
    way=(int)(Math.random()*2);
}
   public void setShip(int size){
       randomSpot();
       while(true){
           boolean fits = checkIfFits(size);
           if(fits)
               break;
           randomSpot();
       }
       if(way==0){   //way down
           for(int i=0; i<size; i++)
               logic.setSq(line+i, row, 1);
       }
       else{            //way right
           for(int i=0; i<size; i++)
               logic.setSq(line, row+i, 1);
       }
   }
```

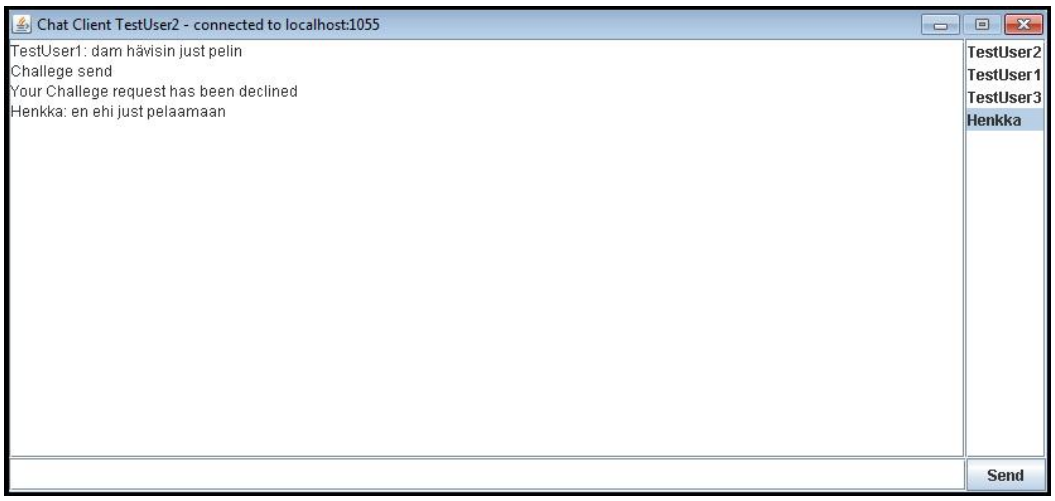
Kuva 17. Laivan asettaja.

Kun laivat on saatu asetettua pelikentälle, voi peli alkaa. Pelin aikana pelaajat ampuvat vuorotellen toistensa ruutuja ja yrittävät saada kaikki vastustajan laivat tuhottua. ShipSinker-luokka saa ammutut koordinaatit ja tarkistaa, tuliko osuma. Aina osuttaessa luokka tarkistaa myös, upposiko laiva. Webclient päivittää näkymän käyttäjälle jokaisen ammuksen jälkeen. Peli voi päättyä kahdella eri tavalla. Joko toisen kaikki laivat on tuhottu tai toinen pelaaja on lähtenyt pelistä ennenaikaisesti.



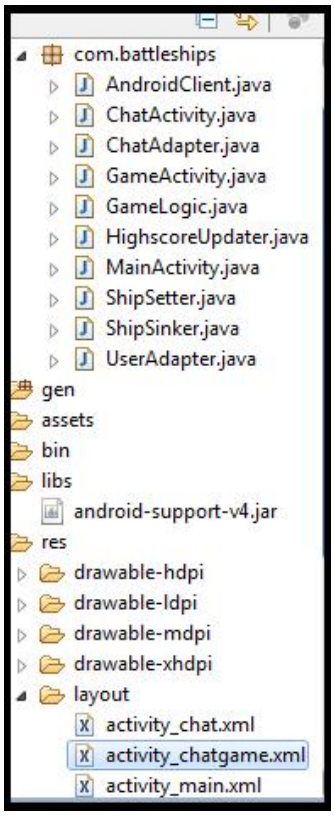
Kuva 18. Pelaajan lähtö.

Jos peliä ei lopetettaisi, kun toinen pelaaja sulkee sovelluksen, jäisi toinen pelaaja jumiin omaan peliinsä siihen asti, että hän sulkee pelin. Tämän takia, jos toinen pelaajista katkaisee yhteyden palvelimeen, julistetaan toinen pelaaja automaattisesti voittajaksi (kuva 18). Pelin jälkeen WebClient ottaa yhteyden PHP-sivulle HighscoreUpdater-luokan avulla. HighscoreUpdater-luokalla voidaan päivittää tietokantaan pelaajien voitot ja häviöt. Pelin päättymisen jälkeen WebClient-luokka vaihtaa käyttäjän näkymän takaisin kuvan 19 mukaiseen chat-ikkunaan.



Kuva 19. Pelin chat-ikkuna.

Android-puolen rakenne projektissa on samantyylinen kuin tietokonepuolen. Android-puolessa luokkia on tietokoneversioon verrattuna hieman enemmän, kuten kuvasta 20 näkyy.



Kuva 20. Android-puolen rakenne.

MainActivity on ensimmäinen luokka, joka ohjelmistossa ajetaan. MainActivity näyttää käyttäjälle sisäänkirjautumissivun. MainActivity-luokka ottaa yhteyden ohjelmiston tietokantaan HTTP-post-käskyllä, kun käyttäjä on antanut kirjautumistiedot. HTTP-post käsky menee PHP-sivun kautta, koska Android ei tue suoran yhteyden muodostamista ulkoisiin tietokantoihin. Tiedot käyvät PHP-sivulla, joka tarkistaa niiden oikeellisuuden tietokannasta ja palauttaa Android-puhelimelle tiedon siitä, löytyikö annetuilla tiedoilla käyttäjää tietokannasta.

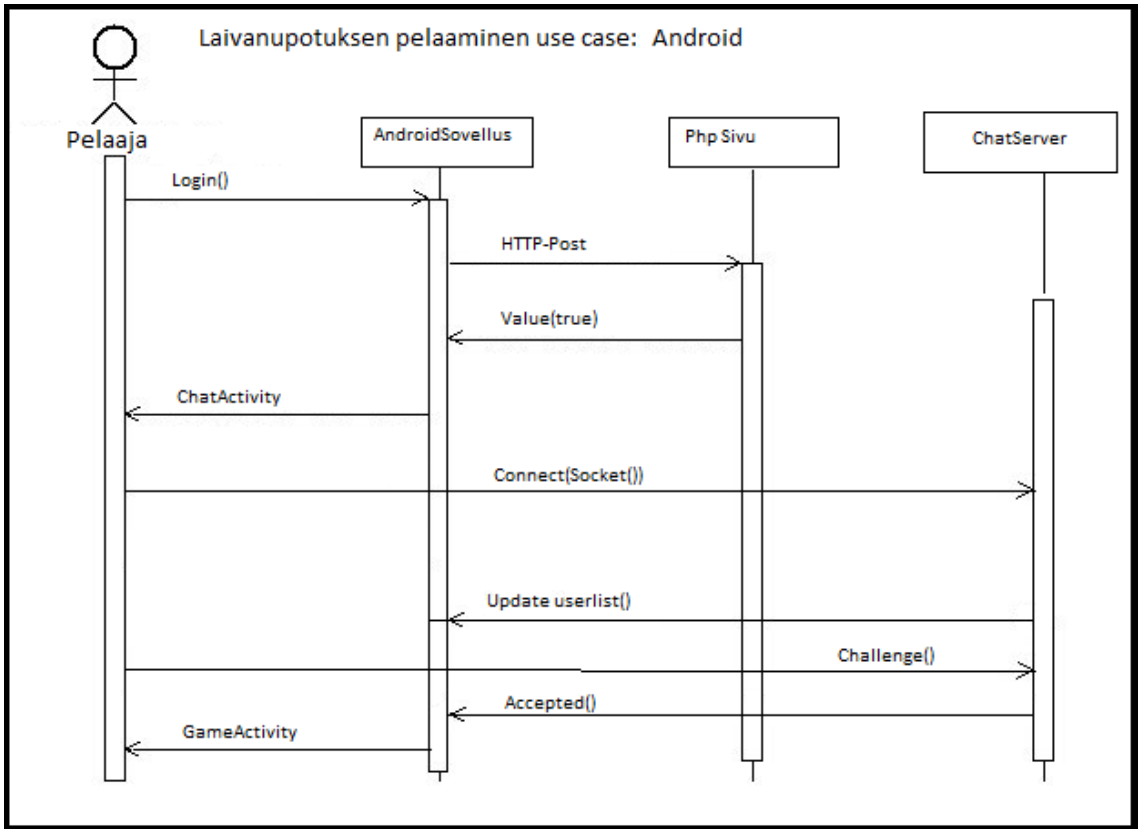
```php
<?php
    function Login(){
        if(empty($_POST['username'])){
            echo("UserName is empty!");
            return false;
        }

        if(empty($_POST['password'])){
            echo("Password is empty!");
            return false;
        }

        $username = trim($_POST['username']);
        $password = trim($_POST['password']);
        if(!isset($_SESSION)){ session_start(); }
```

Kuva 21. Post-sisäänkirjautuminen.

Tämän jälkeen MainAcitivity-luokka avaa ChatActivity-luokan ja sulkee itsensä. ChatAcitivityn tehtävä on luoda käyttäjälle chat-ikkuna näkyviin ja luoda yhteys pelin palvelimelle. ChatActivity käyttää yhteydenluontia varten AndroidClient-luokkaa. Tämä luokka luo Socket-yhteyden samalla tavalla kuin tietokonepuolella oleva WebClient-luokka. ChatAcitivity käyttää myös luokkia UserAdapter ja ChatAdapter, joissa on luotu metodit viestien ja käyttäjien näyttämistä varten ruudulle. Android-puolella pelin aloittamisessa tapahtuu teknisesti hieman eri tavalla kuin tietokonepuolella, sillä Socket-yhteys on luotu ChatActivityyn ja peliä aloittaessa pitää avata GameActivity (kuva 22). GameActivity tarvitsee siis myös Socket-yhteyden luonnin, mutta toiselle käyttäjälle pitää samanaikaisesti ilmoittaa siitä, että Android-sovellus joutuu katkaisemaan yhteyden ja luomaan sen uudelleen uudessa luokassa. Tästä syystä Android-sovellus lähettää vastustajalle tiedon siitä, että Android-laitteen tarvitsee muodostaa yhteys peliin uudelleen ja toinen pelaaja osaa jäädä odottamaan uudelleenyhdistystä, eikä peli pääty automaattiseen voittoon Android-laitteen katkaistessa yhteyden tässä vaiheessa. Pelin päätyttyä GameAcitivity kutsuu HighscoreUpdater-luokkaa samalla tavalla kuin tietokonepuolella ja päivittää pelaajan tilastot tietokantaan (kuva 23).



Kuva 22. Sekvenssikaavio Android.

```php
$username = trim($_POST['username']);
$result = trim($_POST['result']);
$win = "win";

mysql_connect( "mysql.metropolia.fi", "user", "pass" );
mysql_selectdb( "database" );

if(strcmp(string $result , string $win)==0){
    $query = "UPDATE gamestats set wins = wins+1 where users.username='$username'";
    $result = mysql_query( $query ) ;
}
else{
    $query = "UPDATE gamestats set losses = losses+1 where users.username='$username'";
    $result = mysql_query( $query ) ;
}
```

Kuva 23. Highscoreupdate PHP-koodi.

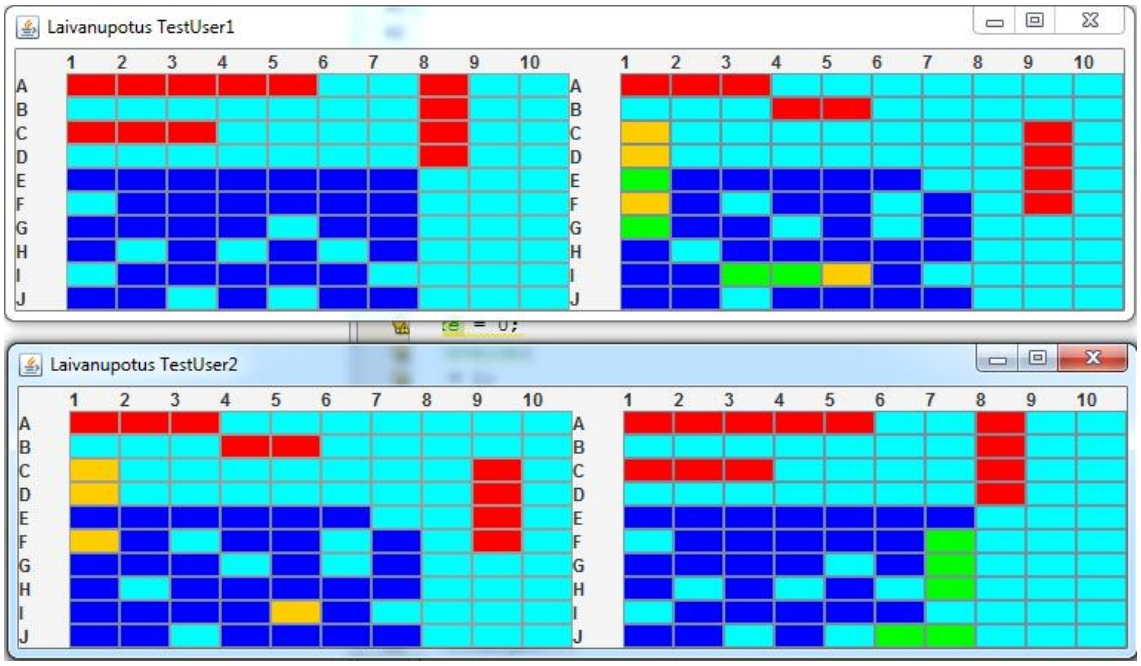
3.3 Käyttöliittymät

Tässä insinöörityössä on kaksi eri käyttöliittymää. Toinen on mobiilisovellus Android-puhelimelle (kuva 24) ja toinen tietokoneen selaimella käytettävä Java-sovellus. Sovellus koostuu käyttöliittymästä ja sen alla olevasta Java-pelistä. Pelaaminen tapahtuu sisäänkirjautumalla, minkä jälkeen käyttäjä ohjataan chat-ruutuun, jossa kaikki muut online-tilassa olevat pelaajat ovat. Sisäänkirjautuminen tekee HTTP-post-kyselyn PHP-sivun kautta ja tarkistaa, ovatko pelaajan antamat tiedot oikein. Mikäli tiedot ovat oikein, päästää sivusto pelaajan eteenpäin, mutta mikäli annetut tiedot ovat väärin, antaa sivusto käyttäjälle virheilmoituksen. Toisen pelaajan haastaminen tapahtuu klikkaamalla pelaajan nimeä pelaajalistalta. Pelihaasteesta lähtevä kysely menee palvelimen kautta haastetulle pelaajalle, ja hän saa siitä ilmoituksen ruudulle. Jos haaste hyväksytään, peli alkaa, jos haaste hylätään, molemmat käyttäjät jatkavat normaalisti chat-huoneessa.

Kuva 24. Android-käyttöliittymä.

Pelit molemmilla laitteilla arpovat molempien pelaajien laivojen paikat omille laitteilleen, ja pelaajat ampuvat vastustajan laivoja vuoron perään. Jos toinen pelaaja lähtee pelistä, palvelin ilmoittaa toiselle pelaajalle siitä ja peliin jäänyt pelaaja julistetaan voittajaksi, kuten kuvassa 18 on kuvattu. Ampuminen tapahtuu haluttua ruutua klikkaamalla. Annetut koordinaatit välittyvät palvelimen kautta vastustajalle, ja peli tarkistaa, osuiko ammus vastustajan laivaan. Tarkistettuaan tiedon peli ilmoittaa ampumavuorossa olleelle pelaajalle tiedon mahdollisesta osumasta. Jos ammus osui, peli tarkistaa myös uppo-

siko osuman saanut laiva. Ruutu muuttaa väriä sen perusteella, osuiko ammus vai ei. Tämä on kuvattu kuvassa 25.



Kuva 25. Tietokone käyttöliittymä.

Pelin jälkeen voittavan pelaajan ohjelma suorittaa SQL-kyselyllä voitonlisäyksen tietokantaan voittaneelle pelaajalle ja häviön hävinneelle pelaajalle. Tämä tehdään siitä syystä, jos pelaaja katkaisi yhteyden, niin hänen peli tiedot eivät päivittyisi koskaan. Pelissä pelaajalla on rajoitettu aika suorittaa ampuminen. Tällä estetään se, että peli ei jatku loputtomiin, mikäli toinen pelaaja jättää sovelluksen auki, vaikka on lopettanut pelaamisen. Jos pelaaja ei ammu aikarajan sisällä, suorittaa sovellus ampumisen satunnaisesti valittuun ruutuun. Näin peli ei lopu heti, jos pelaaja ei jostain syystä pysty ampumaan yhden vuoron aikana. Automaattisesti ammuttavien laukauksien määrää ei ole rajoitettu.

Tietokonepuolella pelaaja kirjautuu peliin verkkosivun kautta. Verkkosivu tarkistaa SQL-kyselyllä, samalla tavalla kuin Android-sovelluksessa, pelaajan tiedot. Tämän jälkeen sivu avaa pelaajalle Java-sovelluksen, jossa pelaaja näkee muut pelaajat. Tietokoneella pelaaminen tapahtuu siis selaimen avulla. Molempien laitteiden, tietokoneen ja Android-puhelimen, käyttöliittymät ovat hyvin samanlaiset, jotta käyttäjille ei voi syntyä etulyöntiasemaa. Vuoropohjaisen pelissä on hyvää se, että laitteiden tekniset ominaisuudet, kuten nopeus, ei anna etulyöntiasemaa pelaamisessa.

3.4 Insinöörityön internetsivut

Pelin verkkosivuilla käyttäjä voi vaihtaa salasanaa, ja uudet pelaajat voivat luoda käyttäjätilejä. Nettisivulla PHP-koodi toimii taustalla kaikessa. Se tarkastaa sisäänkirjautumiset tietokannasta SQL-kyselyillä, joissa kysely muuntaa käyttäjän salasanan MD5-muotoon ja tarkistaa, täsmääkö salattu salasana tietokannassa olevaan tietoon. Käyttäjien luonti tapahtuu samalla tavalla SQL-lauseella, joka lähettää tietokantaan käyttäjän antamat tiedot. Lisäksi PHP toimii rajapintajana Android-sovellukseen. Tämä johtuu siitä, että Androidilla ei voi ottaa suoraa yhteyttä ulkoisiin SQL-tietokantoihin. Android siis tekee tietokantakyselyt PHP-sivun kautta ja saa sieltä parametrinä takaisin tarvittavat tiedot. Palvelin, jossa internetsivut sijaitsevat, tarvitsee sisälleen myös Apache HTTP-Server-palvelinohjelman. Apache toimii sähköpostinvälittäjänä verkkosivuilla. Kun uusi käyttäjä luo tilin, hän saa sähköpostiinsa varmistusviestin. Tässä viestissä on linkki, jolla käyttäjä voi aktivoida tilinsä. Jos käyttäjä unohtaa salasanansa, hän voi resetoida salasanan sähköpostin avulla. Käyttäjien todellisuuden varmistamisella voidaan vähentää mahdollisten bottikäyttäjien määrää.

3.5 Palvelimen koodi

Insinöörityössä on käytössä Java-pohjainen palvelin. Palvelimen pitää olla yhteistyökykyinen puhelimien sekä tietokoneiden kanssa. Pelipalvelin on kirjoitettu Java-koodina. Se käyttää yhteyksien luontia varten Socket-internetprotokollia. Tietokoneet ja Android-puhelimet tukevat tätä Socket-internetprotokollaa.

Palvelinkoodiksi valitsin Javan, koska palvelinkoodia saadaan paljon yksinkertaisemmaksi, kun kaikkien laitteiden yhteydenotto tapahtuu samalla tavalla. Palvelimen suurimmaksi tehtäväksi insinöörityössä jää se, että se ylläpitää käyttäjälistaa kaikista pelaajista, jotka ovat yhteydessä palvelimeen. Lisäksi palvelin välittää käyttäjien viestit ja pelihaasteet muille käyttäjille. Palvelin tiedottaa käyttäjiä, jos esimerkiksi vastustaja sulkee pelin, ja julistaa oikean henkilön voittajaksi. Muutoin toinen pelaaja voisi teoriassa jäädä ikuisesti odottamaan toisen pelaajan siirtoa.

Palvelin ei siis itse pyöritä pelejä vaan kuljettaa pelien välistä tietoja käyttäjille, jotka ovat käyttäjien laitteilla päällä. Tämä estää palvelinta ylikuormittumasta silloin, kun samanaikaisia pelejä on paljon. Viestien filtteröintiä ei tehdä palvelimella, koska tämä

hidastaisi palvelinta huomattavasti. Jos käyttäjä lähettää haasteen peliin, tarkistaa jokaisen pelaajan oma laite, oliko haaste osoitettu hänelle. Palvelin saattaisi ylikuormittua, jos se tekisi itse tämän tarkastuksen ja vasta sen jälkeen lähettäisi pelihaasteen oikealle pelaajalle. Käytetty malli on kevyempi pelintekijälle tai -tarjoajalle, mutta teoriassa monta yhtäaikaista pelihaastetta voi kaataa yksittäisen pelaajan laitteen.

Jos palvelimella on samanaikaisesti 100 pelaajaa ja kymmenen näistä lähettäisi pelihaasteen satunnaisesti eri pelaajille, palvelin joutuisi tarkistamaan jokaisen haasteen yksitellen käyden 100 pelaajan listan Socketeista läpi. Jos yksi haku kestäisi yhden millisekunnin verran, kestäisi kymmenen haasteen läpikäynti 10*100=1000 ms. Jos käyttäjän laite tarkistaa itse, oliko haaste hänelle tarkoitettu, tapahtuu vertailu kaikilla pelaajilla samanaikaisesti. Näin palvelimella kestää vain 10 ms käydä läpi 10 haastetta. Tämä nopeuttaa palvelimen toimintaa huomattavasti ja purkaa rasitetta siltä. Vertailujen käyttämää aikaa ei huomaa helposti pienillä käyttäjämäärillä, mutta käyttäjien lisääntyessä aikavaatimukset kasvavat.

```
while (true) {
    Socket socket = server.accept();
    ClientHandler handler = new ClientHandler(socket);
    executor.execute(handler);
}
```

Kuva 26. Käyttäjien vastaanotto palvelimella.

Palvelin ottaa vastaan uusia käyttäjiä heidän yhdistäessä palvelimeen (kuva 26). Palvelin luo uuden säikeen jokaiselle käyttäjälle ja pitää niistä listaa. Palvelin käyttää Javan Observer-luokkaa, joka tarkkailee yhteyksien tilaa.

```
public void run() {
    logger.info("Accepted client " + socket.getInetAddress() + ":"
            + socket.getPort());
    observable.addObserver(this);
    try {
        outputStream = socket.getOutputStream();
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));
        String line;

        //reads Username from client
        username = reader.readLine();

        //Resolves Ip address to string
        ipAdress = socket.getInetAddress();
        ipAdd = ipAdress.toString();
        ipAdd = ipAdd+":"+socket.getPort();
        System.out.println("Username: "+username);
        System.out.println("Ip: "+ipAdd);
        //adds Username to serverlist
        addUser(username,ipAdd);

        Thread.sleep(400);
        //sends the chat clients new memberlist
        for(int i=0;i<Usernames.size();i++){
            String temp = Usernames.get(i).toString();
            observable.notifyObservers("!ulist:"+temp);
        }
        //Reads msgs from users
        while ((line = reader.readLine()) != null) {
            observable.notifyObservers(line);
        }
        //removes users from the list
        observable.notifyObservers("!urem:"+username);
        removeUser(username, ipAdd);
        //Closes the address
        socket.close();
    } catch (IOException ex) {
        logger.log(Level.SEVERE, null, ex);
        removeUser(username, ipAdd);
        observable.notifyObservers("!urem:"+username);
    } catch (InterruptedException ex) {
            Logger.getLogger(ChatServer.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Kuva 27. Palvelimen säikeen luonti.

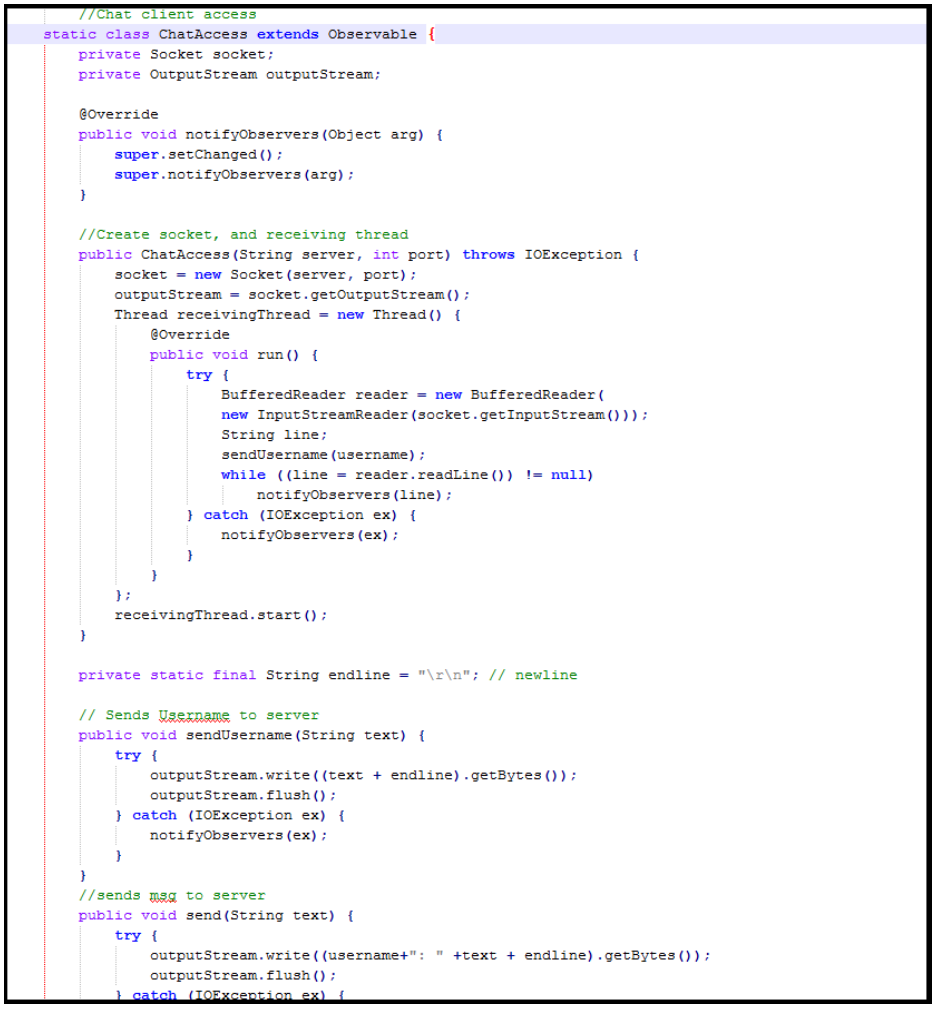
Kun palvelin vastaanottaa uuden käyttäjän, se luo uuden säikeen käyttäjästä (kuva 27). Säie lukee käyttäjän tunnuksen username muuttujaan ja ottaa IP-osoitteen ylös ipAdd muuttujaan. Tämän jälkeen säie ilmoittaa muille käyttäjille uuden pelaajan nimen. Säie seuraa käyttäjältä tulevaa tietoa ja välittää sen eteenpäin. Säie myös seuraa, onko laitteella vielä yhteys palvelimeen. Jos yhteys katoaa, palvelimen täytyy ilmoittaa muille pelaajille yhteyden katoamisesta.

3.6 Highscore ja tietokannan sisältö

Projektin tietokanta pitää sisällään kaikki tiedot päättyneistä peleistä ja käyttäjistä. Tiedot on järjestetty taulukoihin. Tämä tarkoittaa sitä, että käyttäjien tiedot ovat omassa taulukossa ja pelien historiatiedot omissaan. Taulukot on linkitetty avaimilla ja niillä saadaan tarkistettua, kuka on voittanut ja hävinnyt ja kuinka monta peliä. Lisäksi tämä tietokanta on suunniteltu niin, että mikäli tulevaisuudessa halutaan lisätä uusia pelejä käyttäjille, se onnistuu helposti. Jokaiselle käyttäjälle ei tarvitse luoda uusia soluja, vaan lisättävät pelit voidaan luoda suoraan games-taulukkoon. Käyttäjien salasanat on tallennettu MD5-salattuina. Tämä on tehty siltä varalta, että joku käyttäjä onnistuisi suorittamaan toimivan SQL-injektion ja saisi kopioitua tietokannan sisällön. Highscore tulee näkyviin pelin nettisivuilla ja sitä varten on tehty SQL-lauseke, joka etsii tietokannasta parhaimmat pistemäärät.

4 Tiedonkulku laitteiden välillä

Projektissa käytän Socket API -kirjastoja yhteyden luonnissa. Kirjasto tarjoaa suoran yhteyden muodostuksen ja sille ulos- ja sisääntulevalle tiedolle kanavat. Socket-yhteyden muodostaminen alkaa ottamalla yhteys palvelimeen antamalla Socket-luokalle ip-osoite ja portti. Palvelimessa, johon yhteys muodostetaan, pitää olla päällä SocketServer. Ilman SocketServeriä yhteyttä ei voida muodostaa. Kun yhteys on luotu, luodaan sisään- ja ulosmenevälle tiedolle kirjoittaja ja lukija. Nämä pitävät huolen siitä, että kaikki käyttäjän tekemät komennot menevät palvelimelle ja sitä kautta muille pelaajille. Kirjoittaja ja lukija myös vastaanottavat muiden pelaajien viestit ja pelihaasteet (kuva 28).



```
    //Chat client access
static class ChatAccess extends Observable {
    private Socket socket;
    private OutputStream outputStream;

    @Override
    public void notifyObservers(Object arg) {
        super.setChanged();
        super.notifyObservers(arg);
    }

    //Create socket, and receiving thread
    public ChatAccess(String server, int port) throws IOException {
        socket = new Socket(server, port);
        outputStream = socket.getOutputStream();
        Thread receivingThread = new Thread() {
            @Override
            public void run() {
                try {
                    BufferedReader reader = new BufferedReader(
                    new InputStreamReader(socket.getInputStream()));
                    String line;
                    sendUsername(username);
                    while ((line = reader.readLine()) != null)
                        notifyObservers(line);
                } catch (IOException ex) {
                    notifyObservers(ex);
                }
            }
        };
        receivingThread.start();
    }

    private static final String endline = "\r\n"; // newline

    // Sends Username to server
    public void sendUsername(String text) {
        try {
            outputStream.write((text + endline).getBytes());
            outputStream.flush();
        } catch (IOException ex) {
            notifyObservers(ex);
        }
    }
    //sends msg to server
    public void send(String text) {
        try {
            outputStream.write((username+": " +text + endline).getBytes());
            outputStream.flush();
        } catch (IOException ex) {
```

Kuva 28. Tietokoneen Socket-yhteys.

Tieto kulkee Socket-yhteyden avulla palvelimelle ja sieltä takaisin muihin laitteisiin. Molemmat, tietokone ja Android-puhelin, toimivat samalla Socket API:lla. Socketin rinnalla toimii Javan Observer-luokka, joka seuraa muutoksia. Socket-yhteys, joka toimii koko ajan taustalla, on säikeen sisällä. Kun käyttäjä aloittaa pelin tietokoneella ja käyttöliittymän näkymä muuttuu peliruuduksi, säie pysyy taustalla auki ja pitää yhteyden palvelimeen yllä. Pelin loputtua käyttöliittymä vaihtaa näkymän takaisin chat-ruutuun ja käyttää edelleen samaa säiettä. Näin yhteys ei katoa missään vaiheessa. Androidissa yhteyden luonti tapahtuu samalla tavalla luomalla säie, jossa Socket lukee tulevia viestejä siihen asti, että yhteys on katkaistu (kuva 29).

```
public void run() {

    connected = true;

    try {
        //Sets the server address
        InetAddress serverAddr = InetAddress.getByName(SERVERIP);

        //create a socket to make the connection with the server
        Socket socket = new Socket(serverAddr, SERVERPORT);

        try {

            //OutPutStreamWriter sends the messages that user types to the server.
            out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())), true);

            //InputStreamReader reads messages coming back to client
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            //sends username to the server after connecting
            sendUsername(username);

            //while we are connected to the server InputStreamReader keeps reading up coming messages
            while (connected) {
                serverMessage = in.readLine();

                if (serverMessage != null && msgListener != null) {
                    //call the method messageReceived from MyActivity class
                    msgListener.messageReceived(serverMessage);
                }
                serverMessage = null;

            }

        } catch (Exception e) {
            Log.e("Chat", "Error", e);
        } finally {
            //Closes the socket.
            socket.close();
        }

    } catch (Exception e) {
        Log.e("Chat", "Error", e);
    }

}
```

Kuva 29. Socket-yhteys Androidissa.

Socket-yhteyden muodostamisessa kaikki tieto kulkee yhden kanavan kautta. Tästä syystä tieto pitää jotenkin merkata lähtiessä. Tässä ohjelmistossa kaikki viestit on merkattu alkumerkillä, jotta käyttäjät eivät voi chat-ikkunan kautta antaa ohjelmalle komentoja. Esimerkiksi silloin, kun käyttäjä ulos- tai sisäänkirjautuu peliin, on viestin alkuun laitettu merkkijono, jonka perusteella palvelin osaa päivittää käyttäjälistan.

```
//System msg user left
else if(finalArg.toString().toLowerCase().startsWith("!urem")){
    line= finalArg.toString();
    split = line.split(":");
    if(list.contains(split[1])){
        list.removeElement(split[1]);
    }
}
else if(finalArg.toString().toLowerCase().startsWith("!android")){
    line= finalArg.toString();
    split = line.split(":");
    if(split[1].equals(challengedName)){
        androidstart=1;
    }
}
```

Kuva 30. Käyttäjän poisto listalta ja pelin aloitus Android-laitteella.

Kuvan 30 koodi tarkistaa, onko palvelimelta tullut viesti käyttäjän lähdöstä. Jos viesti alkaa "!urem", sovellus tietää silloin, että rivin sisältämä tieto pitää sisällään nimen käyttäjästä, joka lähti pois palvelimelta. Viestin alkaessa ”!ulist” käyttäjä lisätään chat-ikkunassa pelaajalistaan (kuva 31). Jokaisen viestin alku on siis merkattu, jotta käyttöliittymä ja palvelin tietävät, mikä komento on tapahtumassa. Tämä estää sen, että käyttäjä ei pysty itse lähettämään virheellisiä komentoja palvelimelle. Itse pelaaminen tapahtuu Socket-yhteyden avulla.

```
@Override
protected void onProgressUpdate(String... values) {
    super.onProgressUpdate(values);
    String line;
    String[] split;

    //checks that is the string from server Userlist
    if(values[0].toString().toLowerCase().startsWith("!ulist")){
        line = values[0].toString();
        split = line.split(":");
        //checks if the user is already in Userlist
        if(!userList.contains(split[1])){
            userList.add(split[1]);
            userAdapter.notifyDataSetChanged();
        }
    }
    //System Decline msg to player
```

Kuva 31. Android-käyttäjän lisäys chat-näkymään.

Android-sovelluksessa on käytössä AsyncTask, jolla voidaan ajon aikana muuttaa käyttöliittymän tapahtumia. Kun palvelimelta tulee viesti käyttäjän poistoa varten, käyttäjä poistetaan listalta ja päivitetään näkymä muille käyttäjille.

Tietokoneen Java-sovelluksessa WebClient-luokalla on pelin ja chatin käyttöliittymät molemmat samassa luokassa ja luokka pystyy vaihtamaan sitä, kumpi näkymistä on päällä. Tietokone siis käyttää samaa Socket-yhteyttä pelissä ja chatissä. Android-sovelluksessa en saanut yrityksistä huolimatta luotua kahta käyttöliittymää yhdelle luokalle. ViewFlipper API tukee Android-sovelluksissa vain parin elementin muutoksia ja projektissa tarvittiin koko käyttöliittymän vaihtamista chat-näkymästä pelinäkymään. Koska toista käyttöliittymää ei saatu näkyviin Androidin ChatActivity-luokkaan, jouduttiin pelille tekemään uusi luokka. Tämä tarkoittaa sitä, että Android-sovelluksessa pitää sulkea Socket-yhteys ja muodostaa se uudestaan GameActivity-luokassa. Normaalista yhteyden katkaisusta toinen pelaaja olisi saanut automaattisen voiton. Tämän takia jouduin tekemään ”!androidstart”-muuttujan, joka kertoo vastapelaajalle pelaajan käyttävän Android-puhelinta. Kun Android-laitteella pelattaessa yhteys palvelimeen katkeaa, tunnistaa vastustajan laite tämän ja odottaa uudelleenyhdistymistä. Uudelleenyh-

distymistä odotetaan vain kerran. Kuvassa 30 näkyy, kuinka ”!androidstart” merkataan ykköseksi, mikä tarkoittaa vastapelaajan käynnistävän yhteyden uudestaan.

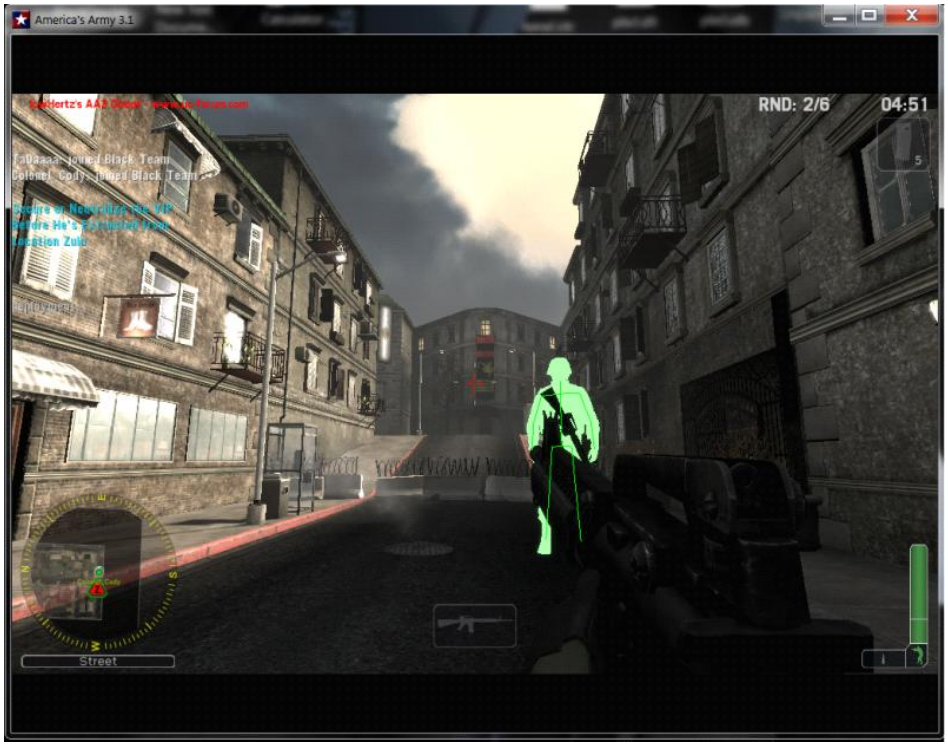
5 Peleissä huijaaminen ja siltä suojautuminen

5.1 Huijaaminen

On tyypillistä, että peleissä yritetään huijata. Nykyaikana monet pelaajat yrittävät löytää keinoja huijata peleissä. Huijaamistyylejä on monia. Yksi yleisimmistä huijaustavoista on etsiä ohjelmoijan tekemiä virheitä ja käyttää niitä epäreilun edun luomisessa muita pelaajia vastaan. Tästä syystä ohjelmoijan tulee olla erittäin tarkka ohjelmakoodin kanssa. Ohjelmaa tehdessä ohjelmoijan on hyvä tehdä erilaisia testitilanteita ja varmistaa, että sovellus toimii oikein. Ohjelmoijan pitää osata myös keksiä tilanteita, joita erilaiset käyttäjät voivat vahingossa tai tarkoituksella kohdata pelissä: onnistuuko kahden pelaajan haastaminen samanaikaisesti? Loppuuko peli, jos toinen pelaaja lähtee pelistä ennen sen loppumista vai jääkö toinen pelaaja ikuisesti odottamaan pelin loppumista? Lisäksi ohjelmoijan pitää miettiä, miten tietoa käsitellään pelin aikana.

Aina ei kannata kirjoittaa ohjelmakoodia helpoimmalla tavalla. Nykyaikana on iso lista erilaisia ohjelmia, joilla ajon aikana käyttäjä pääsee käsiksi suoraan ohjelmistokoodiin. Jos ohjelmoija ei tee sovellukseen erilaisia tarkistuspisteitä, käyttäjä voi ajaa omaa koodiaan pelin välissä ja saavuttaa siten etulyöntiasema.

Ohjelmoijan on tärkeää myös pohtia, missä pelin tietoja säilytetään. Ohjelmoija voi miettiä, että sovelluksen ajaminen on helpompaa, jos pelaajat jakavat tiedot keskenään ja jokainen pelaaja hoitaa koodin ajamisen omalla laitteellaan. Tämä ei aina ole paras vaihtoehto, koska kaikki tiedot, joita käyttäjälle annetaan, voidaan kaivaa ulos. Esimerkiksi tässä insinöörityössä molempien pelaajien laivojen paikat pidetään omana tietona ja toiselle käyttäjälle kerrotaan ainoastaan, osuiko ammus vai ei. Tällä vältytään siltä, että toinen pelaaja voisi urkkia pelin muistista toisen pelaajan laivojen paikat ja ampua jokaisen laivan ensimmäisellä yrityksellä.



Kuva 32. Pelaaja on muokannut koodia etulyöntiaseman saavuttamiseksi [16].

Kuvan 32 tilanteessa käyttäjä on päässyt käsiksi ohjelman koodiin ja onnistunut muokkaamaan sitä niin, että pelissä pelihahmot näkyvät kirkkaanvihreinä. Näin käyttäjän on helpompi erottaa muut pelaajat maastosta ja saada etulyöntiasema muita pelaajia vastaan. Tämä olisi voitu estää piilottamalla pelin koodi paremmin tai tekemällä siitä vaikeaselkoisempi.

5.2 Koodin piilottaminen käyttäjältä

Java-koodin kääntäminen tavukoodista takaisin lähdekoodiksi on mahdollista. Tämä tapahtuu samalla tavalla kuin Java-koodi ajetaan tavukoodiksi, mutta toisin päin. Parhaimmat kääntäjät voivat kääntää tavukoodin lähes alkuperäisen tasoiseen lähdekoodiin. Yleensä koodin täydellinen kääntäminen on kuitenkin haastavaa.

Koodin kääntäminen on mahdollista, koska Java ei käännä koodiaan samalla tavalla kuin esimerkiksi C tai C++. C ja C++ kääntävät lähdekoodin binääriseksi konekoodiksi, jota ei pysty enää kääntämään takaisin. Javan tuottama tavukoodi ei ole suoraan sidoksissa käyttöjärjestelmään, ja siksi se sisältää kaiken tärkeän informaation, joka löytyy lähdekoodista. Tavukoodi pitää sisällään kaikkien metodien nimet. Muuttujien nimet

on mahdollista myös palauttaa, jos niille on olemassa vianmäärittelyyn käytettyjä tietoja.

Tästä syystä monet ohjelmoijat "obfuskoivat" lähdekoodinsa julkaistessaan sovelluksen (kuva 33)[18]. Obfuskoinnilla tarkoitetaan koodin monimutkaistamista. Koodi pidetään samana, mutta kaikki koodin metodit ja muuttujat vaihdetaan merkitsemättömiksi teksteiksi. Laivanupotus-luokka voidaan nimetä suoraan pelkäksi i-kirjaimeksi ja sen sisäiset muuttujat kaikki yhden kirjaimen mittaisiksi. Tämä vaikeuttaa koodin lukemista ja voi tehdä siitä lähes mahdotonta, jos tavukoodi yritetään kääntää takaisin lähdekoodiksi. Yleensä metodien ja muuttujien nimet antavat kuvan siitä, mitä ohjelman osat tekevät. Jos ohjelman jokainen osa on vain yksi kirjain, sen ymmärtäminen on huomattavasti vaikeampaa kuin normaalimuotoisen koodin. Ko. tilanteessa ohjelmoija pitää kahta versiota ohjelmastaan: yksi, joka on normaalissa muodossa ja jota hän voi muunnella ja uudistaa, ja toinen, joka on kokonaan obfuskoitu ja on käyttäjien saatavilla.

```
package lopputyochat;

/**
 * Sisältää laivanupotuspelin toimintalogiikan.
 * @author m0703172
 */
public class PeliLogiikka {
    protected int[][] ruudukko;
    protected int[][] vastustajanRuudukko;
    private Laskuri laskuri;
    private LaivanUpottaja upottaja;
    private LaivanAsettaja asettaja;

    /**
     * Luo PeliLogiikka ilmentymän ja sille Laskuri, LaivanUpottaja sekä LaivanAsettaja ilmentymät.
     */
    public PeliLogiikka(){
        ruudukko = new int[10][10];
        vastustajanRuudukko = new int[10][10];
        laskuri = new Laskuri();
        upottaja=new LaivanUpottaja(this);
        asettaja=new LaivanAsettaja(this);
    }

    /**
     * Kertoo mitä parametrien mukaisessa koordinaateissa on.
     * @param r rivi koordinaatti.
     * @param s sarake koordinaatti.
```

```
package lopputyochat;

/**
 *
 * @author m0703172
 */
public class A {
    protected int[][] i;
    protected int[][] b;
    private V v;
    private L l;
    private A a;


    public A(){
        i = new int[10][10];
        b = new int[10][10];
        v = new V();
        l=new L(this);
        a=new A(this);
    }

    public int rO(int r, int s){
        return i[r][s];
    }
    public int vRO(int r, int s){
        return b[r][s];
    }
```

Kuva 33. Esimerkki obfuskoinnista.

Ohjelmoijalla on myös muita tapoja salata koodiaan. Yksi suosittu tapa on lisätä muuttujia ja metodeita, jotka eivät oikeassa koodissa tee mitään. Nämä koodinpätkät vaikeuttavat lähdekoodin tulkitsemista lisää, koska toinen ihminen ei voi tietää, mitkä osat koodista ovat oikeita metodeja ja mitkä ovat sinne ujutettuja hämäyksiä. Lisäksi osilla tällaisista salausmetodeista pystytään kaatamaan ohjelmia, jotka kääntävät tavukoodia takaisin lähdekoodiksi.

Ohjelmistokoodin monimutkaistamisella on myös haittoja: jos sovellus kaatuu, on virheen tulkitseminen vaikeaa sekoitetun ohjelmistokoodin seasta. Ohjelmoijan on vaikea

myös uudelleen saada aikaan ongelman aiheuttanut tilanne, koska ei ole tietoa, mikä osa koodista sai kaatumisen aikaan.

6 Yhteenveto

Insinöörityössä on esitelty, mitä tarvitaan peliympäristön tekemiseen erilaisille laitteille. Työssä tehtiin yksinkertaiset käyttöliittymät ja saatiin verkkosivut käyttäjien luontia ja hallinnointia varten. Työssä tehtiin pelille palvelin ja käytiin läpi, mitä asioita kannattaa suorittaa palvelimella ja mitkä osat kannattaa tehdä käyttäjien laitteilla. Projektissa käytettiin Socket API:a yhteyden muodostamista ja viestien suodattamista varten.

Työssä on kuvailtu, miten ohjelmoija voi suojata sovellustaan väärinkäytöltä. Lisäksi työssä opittiin, miksi erityisesti Java-koodin monimutkaistaminen on kannattavaa ja miksi jotain muuttujia ei kannata jakaa suoraan käyttäjien välillä. Lisäksi huomattiin, miksi koodin monimutkaistaminen voi luoda ylläpitäjälle ongelmia.

Projektin runko ja tietokanta antavat jatkokehitysmahdollisuuden muiden pelien lisäämiseksi sovellukseen. Peliin haastaminen tapahtuisi samalla tavalla, mutta sillä lisäyksellä, että haastaja valitsisi myös pelattavan pelin. Tietokanta on suunniteltu niin, että sinne pelien lisääminen on helppoa, eikä se aiheuta paljoa lisätyötä.

Omasta mielestäni sovelluksen tekeminen meni hyvin, vaikka haasteita tuli matkalla. Työn aikana opin paljon Socket-yhteyden luonnista. Yhteyden modostaminen Socket-API:lla oli minulle kokonaan uutta ja sen olisi mahdollisesti voinut toteuttaa paremmin. Insinöörityötä varten Socket-yhteys kuitenkin toimii.

Työn viimeistely on venynyt ja olisin toivonut, että olisin saanut työn kokonaan valmiiksi lyhemmässä ajassa. Työn venyminen on aiheuttanut sen, että tehty sovellus toimii huonosti nykyisissä Android-käyttöjärjestelmissä. Sovellus toimii sellaisenaan, koska ympäristö tietokonepuolella ei ole muuttunut niin paljon.

Lähteet

1 SQL. Verkkodokumentti. Saatavissa: http://en.wikipedia.org/wiki/SQL. Luettu 1.4.2013.

2 SQL kyselyn rakenne. Kuva. Saatavissa: http://en.wikipedia.org/wiki/File:SQL_ANATOMY_wiki.svg. Luettu 1.4.2013.

3 Salausalgoritmeja. Verkkodokumentti. Saatavissa: http://www.cs.uta.fi/kurssit/titu/luennot/7_luento_salausalgoritmeja.pdf Luettu 3.4.2013.

4 Java. Verkkodokumentti. Saatavissa: http://www.java.com/en/about/. Luettu 10.4.2013.

5 Java-ohjelmointikieli. Verkkodokumentti. Saatavissa: http://en.wikipedia.org/wiki/Java_%28programming_language%29 Luettu 10.4.2013.

6 Apache-palvelinsovellus. Verkkodokumentti. Saatavissa: http://en.wikipedia.org/wiki/Apache_HTTP_Server. Luettu 15.4.2013.

7 The Fasted Webserver, Webperformance. Verkkodokumentti. Saatavissa: http://www.webperformance.com/load-testing/blog/2011/11/what-is-the-fastest-webserver/. Luettu 26.9.2015.

8 Response time. Kuva. Saatavissa: http://www.webperformance.com/load-testing/blog/wp-content/uploads/2011/11/response_time1.png. Luettu 26.9.2015.

9 Android operating system. Verkkodokumentti. Saatavissa: http://en.wikipedia.org/wiki/Android_(operating_system). Luettu 15.4.2013.

10 Training for Android developers. Verkkodokumentti. Saatavissa: http://developer.android.com/training/index.html. Luettu 17.4.2013.

11 Androidin markkinaosuus nousi 79 pronsenttiin, Kauppalehti. Verkkodokumentti. Saatavissa: http://www.kauppalehti.fi/uutiset/androidin-markkinaosuus-nousi-79-prosenttiin/fDs4MNkM. Luettu 26.9.2015.

12 PHP. Verkkodokumentti. Saatavissa: https://en.wikipedia.org/wiki/PHP. Luettu 15.4.2013.

13 Battleship game. Verkkodokumentti. Saatavissa: https://en.wikipedia.org/wiki/Battleship_(game). Luettu 20.4.2013.

14 Battleship. Kuva. Saatavissa: http://images3.wikia.nocookie.net/__cb20120303020434/battleship/images/f/fd/Battleship-1.jpg. Luettu 20.4.2013.

15 NeatBean. Verkkodokumentti. Saatavissa: http://fi.wikipedia.org/wiki/NetBean. Luettu 25.4.2013.

16 Kuva. Saatavissa: http://img.photobucket.com/albums/v410/garyemmitt/2uji1sj.png. Luettu 25.4.2013.

17 MD5 Salaus. Verkkodokumentti. Saatavissa: https://fi.wikipedia.org/wiki/MD5. Luettu 20.4.2013.

18 Obfuskointi. Verkkodokumetti. Saatavissa: https://en.wikipedia.org/wiki/Obfuscation_%28software%29. Luettu 21.4.2013.