Examensarbete

HÖGSKOLAN
I SKÖVDE

AVVAKTANDE AJAX-ANROP
En avlastningsteknik för 3G-nätet

LONG POLLING WITH AJAX
A mobile network offloading technique

Examensarbete inom huvudområdet Datalogi
Grundnivå 30 Högskolepoäng
Vårtermin 2012

Tommy Andersson

Handledare: Helen Pehrsson
Examinator: Henrik Gustavsson

Sammanfattning

Den ökade användningen av nätuppkopplade mobilapplikationer har resulterat i en överbelastning i 3G-nätet. Förslag för att avlasta nätet har bland annat varit genom alternativa uppkopplingar, vilket innebär en uppoffring av 3G-nätets tillgänglighet. Detta leder till frågor om andra avlastningsmetoder. Arbetets målsättning var att se hur en applikations datatrafik påverkas av att klassiska intervallanrop ersätts av avvaktande AJAX-anrop. Detta görs genom att implementera respektive anropsmetod i två identiska alfapetapplikationer. Mätdata erhölls genom att låta användare spela längre omgångar av respektive applikation, vilket även genererar realistiska uppdateringar. Resultaten visar att den klientgenererade bandbreddskonsumtionen minskas vid användning av avvaktande AJAX-anrop. Detta gör den, i kontrast med klassiska intervallanrop, till en möjlig avlastningsmetod.

Nyckelord: AJAX, long polling, Javascript, datatrafik.

Innehållsförteckning

1 Introduktion

I samband med att smartphones tog över mobilmarknaden erbjuder mobiloperatörer användare tillgång till 3G-nätet för en fast månadskostnad (Balasubramanian, Mahajan & Venkataramani, 2010). Dock har efterfrågan blivit större än tillgängligheten, vilket resulterat i en överbelastning (Ristanovic, Le Boudec, Chaintreau & Erramilli, 2011). Zhuo, Gao, Cao & Dai (2011) föreslår ett sätt att avlasta 3G-nätet genom alternativa uppkopplingar. Trots dessa avlastningar sker fortfarande synkroniseringar oftast över 3G-nätet då tillgängligheten av WIFI surfzoner är bristfällig.

Fördelen med 3G-nätet är tillgängligheten och att behöva offra denna tillgänglighet för att erhålla en brukbar hastighet är oönskat. Mobilplattformen använder mycket högfrekventa serveranrop för att användare snarast skall erhålla aktuell information och i någon mån simulera realtid. Detta kan göras på flera olika sätt, men det vanligaste är att klienten anropar servern och erhåller ett omedelbart svar. Om klienten istället väntar på en förändring av information, exempelvis att det blir dennes tur i ett spel, sker dessa anrop i intervaller. Detta för att klienten snarast skall erhålla ny information. Har servern inte någon ny information att erbjuda klienten meddelas detta, oftast genom ett tomt svar.

Varje gång servern returnerar ett tomt svar slösas bandbredd. Desto oftare anrop sker, desto större är sannorlikheten för oförändrad data (Sanjaya, 2011). Med avvaktande AJAX-anrop ersätts intervallanropen mellan klient och server med ett anrop per förändring. Servern kör sedan förfrågningarna lokalt och returnerar inget svar förrän data är förändrad, vilket förebygger tomma svar. Genom att förebygga de tomma svaren flyttas även belastningen från nätet till servern (Bozdag & van Deursen, 2008; Bozdag, Mesbah & van Deursen, 2007).

Bozdag och van Deursen (2008) samt Sanjaya (2011) påvisar att avvaktande AJAX-anrop, med slumpmässigt genererade uppdateringsfrekvenser, minskar datatrafiken i ett trådlöst nätverk där den klassiska AJAX-applikationen använder sig av hög- och medelfrekventa anrop. Genom att hantera serveranropen på ett liknande sätt, men över 3G-nätet, kan tekniken testas på en mobilplattform.

Syftet med arbetet är att avgöra om avvaktande AJAX-anrop är en lämplig avlastningsmetod för 3G-nätet. Detta testas genom att återskapa en applikation med hög datatrafik, en Alfapet-applikation motsvarande Wordfeud (Böhmer, Hecht, Schöning, Krüger & Bauer, 2011). Fyra spelare skall samtidigt mötas i en spelsession för att testa hur ofta serveranrop behöver ske innan resultatet påverkas eller användaren uppfattar att applikationen inte arbetar i realtid.

För att mäta skillnaden i datatrafik mellan avvaktande och klassiska AJAX-anrop skapas två olika versioner av applikationen, en som använder högfrekventa serveranrop samt en som använder avvaktande AJAX-anrop. Genom att använda avvaktande AJAX-anrop förväntas den mängd datatrafik applikationen genererar sänkas drastiskt.

2 Bakgrund

I samband med internets tillväxt har webbapplikationer med responsiva användargränssnitt och ökad interaktivitet blivit allt mer efterfrågade. Webbutvecklare har gjort försök att lösa den bristfälliga användarinteraktionen genom att använda diverse applikationer från bland annat Macromedia, Microsoft och Sun. Trots dessa försök har webbapplikationer, i kontrast med datorprogram, klassats som långsamma. Detta för att varje användarinteraktion ofta leder till serverkontakt samt att hela webbsidan måste laddas om (Paulson, 2005; Bozdag, et al., 2007).

2.1 AJAX

För att motverka webbapplikationers bristfälliga användarinteraktion använder webbutvecklare en implementation av AJAX (Asynchronous JavaScript and XML) som utvecklades redan under 1990-talet. Den första webbimplementationen av AJAX var tillgänglig för Internet Explorer i form av ActiveX-objektet. När andra webbläsare skulle implementera ActiveX-objektet uppstod dock kompatibilitetsproblem. Detta ledde till att W3C (World Wide Web Consortium) år 2006 släppte det numera standardiserade XMLHttpRequest-objektet (W3C, 2006; Paulson, 2005), hädanefter benämnt som objektet.

Objektet är baserat på Javascript och möjliggör en helt ny nivå av användarinteraktion. Det tillåter nämligen AJAX-applikationer att hämta och skicka data utan att användaren upplever några visuella avbrott eller att webbsidan behöver laddas om. En av de första webbimplementationerna av AJAX hittas i Googles kartapplikation. Applikationen är en satellitfototjänst där användaren med hjälp av musen kan navigera på kartan och direkt erhålla information om den nya platsen (Paulson, 2005).

Serverinteraktion med objektet kan ske på flera olika sätt, det vanligaste är dock HTTP server pull, vilket innebär att klienten anropar servern och erhåller ett direkt svar (Bozdag & van Deursen, 2008; Hauswirth & Jazayeri, 1999). När klienten istället inväntar en förändring av information sker dessa anrop i intervaller, vilket vid oförändrad data slösar bandbredd. En alternativ teknik som löser detta problem är avvaktande AJAX-anrop (Bozdag & van Deursen, 2008; Sanjaya, 2011).

2.2 Avvaktande AJAX-anrop

Avvaktande AJAX-anrop används då klienten väntar på att information skall förändras. Klienten anropar servern med bifogad information. Skiljer sig den bifogade informationen från den som finns lagrad på servern returneras den. Finns ingen differens returneras istället inget svar förrän informationen differerar, vilket drastiskt minskar antalet serveranrop (Bozdag & van Deursen, 2008). Detta gör avvaktande AJAX-anrop till en avlastningsmetod lämpad för 3G-nätet.

W3C (2010) nämner i sin officiella rekommendation för utveckling av mobila webbapplikationer flera metoder för att optimera nätverksförfrågningar. Den primära avlastningsmetoden pekar på att applikationer skall minimera antalet HTTP-förfrågningar. En metod som nämns för att göra detta är att monitorera användarens aktivitet och därefter bestämma hur frekvent intervallanropen skall ske. Avvaktande AJAX-anrop är en alternativ lösning som minimerar antalet HTTP-förfrågningar utan att behöva monitorera användarens aktivitet genom att flytta belastningen från nätet till servern (Bozdag & van Deursen, 2008).

Avvaktande AJAX-anrop är inte begränsat till webbapplikationer utan kan även användas vid utveckling av mobilapplikationer. Ramverket Phonegap gör det möjligt att utveckla mobilapplikationer i HTML5, CSS3 och Javascript för att sedan, med Phonegap, kompilera dessa till Java eller Objective-C för respektive mobilplattform (Charland & LeRoux, 2011). Detta möjliggör testning av avvaktande AJAX-anrop på en mobilplattform.

2.3 Verktyg

Phonegap möjliggör utveckling av mobilapplikationer med webbstandarder. Resultatet blir en hybridapplikation som varken är plattformspecifik eller helt webbaserad. Applikationen arbetar med ett inbäddat webkit som agerar likt en plattformspecifik mobilapplikation. Även tillgång till diverse funktioner från enhetens API (Application Programming Interface) erhålls (Charland & LeRoux, 2011).

Ett genomgående problem vid utveckling av mobilapplikationer är faktumet att olika mobilplattformar använder olika språk, exempelvis använder Android programmeringsspråket Java medan iOS använder Objective-C. Detta leder i sin tur till att olika mobilapplikationer måste utvecklas för de olika plattformarna. Phonegap löser problemet genom att låta webbutvecklare skapa en applikation som i sin tur kan generera applikationer för olika mobilplattformar (Charland & LeRoux, 2011).

Ramverk som går hand i hand med Phonegap är jQuery och jQuery Mobile. Ramverket arbetar, liksom Phonegap, med syftet för applikationsutveckling till multipla plattformar och enheter (Charland & LeRoux, 2011). jQuery släpptes januari 2006 av John Resig och är idag det populäraste Javascript-ramverket vid klientbaserad AJAX-utveckling. jQuery bygger på öppen källkod och används oftast som en enstaka Javascript-fil. Filen erbjuder förenklad DOM-hantering, färdigprogrammerade attribut och event samt AJAX-funktioner (Kim. H, Kim, Jang, Han & Ceong, 2010).

I avlastningssyfte bidrar jQuery med verktyg för att hantera XMLHttpRequest-objektet med ”$post” och ”$get” (Kim, et al., 2010). Post och get är två olika metoder som underlättar för webbutvecklare att skicka data mellan klient och server. I utveckling av mobilapplikationer med Phonegap används respektive metod i domänöverskridande förfrågningar. Detta innebär att förfrågan inte bara kan skickas till värdservern utan även till andra domäner (Hai-ping, Yang, Wei & Yong-sheng, 2010).

För att mäta hur mycket datatrafik en applikation genererar krävs ett mätverktyg. Google (2012) erbjuder flera applikationer för att kontrollera bandbreddskonsumtionen på Android-telefoner, en av dessa är ”3G Watchdog”. Applikationens primära syfte är att övervaka bandbreddskonsumtionen för konsumenter med en begränsad bandbreddskvot. I applikationen kan användaren se hur mycket bandbredd som har används och kan därmed även användas som ett mätverktyg.

3 Problemformulering

Choi, Ji, Park, Kim och Silvester (2011), Balasubramanian, et al. (2010) samt Ristanovic, et al. (2011) skriver alla om en överbelastning i mobiloperatörens AT&T 3G-nät. Nätet har under tre år, i samband med lanseringen av iPhone 2006-2009, ökat datatrafiken med 5000%. Även Stenquist (2011), Efendić (2011) samt Andersson och Åberg (2011) har publicerat artiklar om överbelastning i Telia och Tele2's 3G-nät. Kunder som betalar för en anslutning med tre megabit per sekund erhåller, enligt Andersson och Åberg (2011), inte ens en halv megabit per sekund. Tele2's presschef Annika Kristersson (Andersson och Åberg, 2011) menar att den ökade användningen av nätuppkopplade mobilapplikationer är en bidragande orsak till överbelastningen. Telia hanterar situationen genom att bygga ut 3G-nätet (Andersson och Åberg, 2011), vilket är en dyr process (Choi, et al., 2011). Detta leder till frågan om möjliga avlastningsmetoder på applikationsnivå.

Avvaktande AJAX-anrop är en avlastningsmetod lämpad för 3G-nätet då det minskar bandbreddskonsumtionen genom att minimera antalet serveranrop. Arbetets syfte är att implementera avlastningsmetoden i en realistisk mobilapplikation för att kunna jämföra bandbreddskonsumtionen med högfrekventa intervallanrop. Detta ska leda till svar på följande frågor:

> Hur påverkas datatrafiken om högfrekventa intervallanrop ersätts av avvaktande AJAX-anrop och hur sällan kan dessa anrop ske utan att användaren uppfattar att applikationen inte sker i realtid?

Implementationen under arbetets förutsättningar medför avgränsningar. Då arbetets primära fokus är riktat mot mätningar av bandbreddskonsumtion avgränsas arbetet från irrelevanta implementationer och frågeställningar. Faktorer som inte påverkar dessa mätningar innefattar flerspråkstöd och applikationsdesign. Arbetet kommer därför inte ta upp dessa punkter.

Sanjaya (2011) påvisar i sin artikel att avvaktande AJAX-anrop i samband med oförutsägbar uppdatering av information resulterar i en klar minskning av datatrafik. I rapporten finns dock ingen programkod tillgänglig för att evaluera hur de oförutsägbara värdena genereras, vilket leder till frågor kring tillförlitlighet. Sanjayas testsviter har inte heller någon koppling till riktiga användare, utan använder en applikation för att generera oförutsägbara uppdateringar. Genom att utveckla en mobilapplikation där klienten själv står för både uppdatering och anrop kan tekniken testas av riktiga användare.

4 Metod

För att garantera den intervallfrekvens som används samt erhålla tillförlitlig mätdata utvecklas två nya mobilapplikationer liknande Wordfeud eller Words with Friends. Applikationerna använder en förenklad implementation av XMLHttpRequest-objektet via jQuery för att skicka uppdateringar och anrop till servern. Tekniken implementeras då en spelare väntar på sin tur. Traditionellt sker detta med högfrekventa intervallanrop. Dessa anrop kommer sedan jämföras med avvaktande AJAX-anrop.

Kasemvilas och Firpo (2009) samt Bozdag, et al. (2007) påvisar att experiment är en lämplig metod för att jämföra prestanda mellan olika webbtekniker, samt identifiera eventuella användbarhetsproblem. Kasemvilas och Firpo (2009) kompletterar sitt experiment med en enkätundersökning medan Liew, Kaziunas, Liu och Zhuo (2011) använder en intervju som en sekundär utvärderingsmetod för att besvara användbarhetsrelaterade frågor. I detta arbete kommer experimentet kompletteras med en öppen riktad intervju vilket resulterar i en mer kvalitativ studie än en enkätundersökning (van Velsen, van der Geest & Klaassen, 2007).

4.1 Experiment

I brist på tillförlitliga resultat från litteratursökningar krävs ytterligare metoder för att besvara arbetets frågeställning. Metoden mest lämpad att besvara den tekniska delfrågan är ett experiment. Syftet med experimentet är att verifiera om bandbreddskonsumtionen, under experimentets omständigheter, minskas eller inte. För att garantera giltig mätdata är det viktigt att utomstående faktorer så som applikationssynkroniseringar och alternativa uppkopplingar kan uteslutas. Detta garanteras genom att samla testkandidaterna och kontrollera respektives smartphone innan experimentets genomförande.

När användbarhetsfrågor är inblandade anser Kasemvilas och Firpo (2009) att max fem kandidater skall delta i experimentet åt gången. Alfapetapplikationen är designad för fyra spelare, vilket även blir testgruppens storlek. Dessa fyra personer, som idealt använder likadana smartphones, ska köra flera spelomgångar av respektive applikation. För att erhålla giltig mätdata avslutas alla applikationer som använder nätåtkomsten för synkronisering. Datatrafiken kan sedan noggrant mätas av applikationen 3G Watchdog. En viktig egenskap mätapplikationen har är att kunna skilja på skickad och mottagen data. Detta gör det möjligt att se om klienten eller servern står för den primära belastningen. Den mängd data som applikationen genererar lagras och publiceras i den slutgiltiga rapporten för att stödja vilken av applikationerna som under experimentets omständigheter använder minst datatrafik. Exakt vad som skickas och vem som skickar det är dock irrelevant och lagras inte.

Platsen för experimentet är teknikmässigt väldigt mobil, eftersom det enda beroendet är respektive testkandidats smartphone. Platsen behöver dock kunna tillhandahålla en acceptabel anslutning till 3G-nätet. Detta för att minimera den fördröjning användaren kan uppleva på grund av internetanslutningen. Den primära fördröjningen skall istället upplevas på grund av applikationens anropsintervaller.

4.2 Intervju

Experimentet behöver kompletteras med ytterligare en metod för att kunna besvara arbetets användbarhetsrelaterade delfråga. Det finns tre metoder lämpliga att besvara användbarhetsrelaterade frågor, bland dessa ingår: att tänka högt, enkätundersökning och intervju. Metoden som ensamstående resulterar i bäst kvalitativa studie är intervju (van Velsen, et al., 2007).

Det finns olika sätt att utföra en intervju, den form som är lämpligast för syftet är dock en öppen riktad intervju. Intervjuformen kan liknas vid en enkätundersökning då de frågor som ställs är förberedda. Genom att hålla intervjun öppen kan även diskussioner kring frågorna uppkomma vilket resulterar i en mer kvalitativ studie än exempelvis en halv- eller helstrukturerad intervju (Sallnäs, 2007).

En alternativ metod hade varit att använda ett webbaserat formulär. Testkandidaten skulle under spelets gång fylla i formuläret liknande en webbaserad enkätundersökning. På grund av spelets tidsbegränsning och den redan tidskrävande implementationen förväntas en öppen riktad intervju vara mer lämplig och resultera i en mer kvalitativ studie.

Frågorna i intervjun kommer att relatera till experimentet. Det är därför viktigt att experimentet tar plats innan intervjun. Målet med intervjun är att se hur sällan intervallanrop kan ske utan att användaren uppfattar att applikationen inte sker i realtid. En utomstående faktor är 3G-nätet som beroende på tid och plats kan bidra till denna fördröjning. Genom att utföra experimentet på en plats med en acceptabel anslutning till 3G-nätet och under en mindre aktiv tidpunkt (Ristanovic, et al., 2011; Balasubramanian, et al., 2010; Choi, et al., 2011), förväntas denna faktor vara ett mindre problem. Frågorna kommer under intervjun att riktas mot eventuell fördröjning testkandidaten kan komma ha uppfattat.

Platsen där intervjun kommer äga rum är relativt irrelevant, det viktigaste är att testkandidaten känner sig bekväm i miljön. Idealt skulle detta ske i respektive kandidats hemmiljö. Detta kommer dock inte vara möjligt då alla fyra testkandidater intervjuas separat under samma tillfälle som experimentet. Intervjun kommer därför istället att ske på samma plats som experimentet genomförs, alternativt en närliggande plats mer lämplig för syftet.

Under intervjuerna kommer anteckningar att föras. Delar av dessa anteckningar kommer även att publiceras i den slutliga rapporten. Inga integritetskänsliga uppgifter kommer dock publiceras. När projektet är över kommer anteckningar från alla intervjuer att förstöras i integritetsskyddandesyfte.

5 Testapplikation

Testapplikationen är ett alfapetspel med målet att användare skall spela ord för mest poäng, och på så vis generera realistiska uppdateringsfrekvenser. Applikationen fungerar som ett instrument för att avgöra om avvaktande AJAX-anrop genererar mindre datatrafik än klassiska intervallanrop.

Grundtanken var att med Phonegap konvertera alfapetspelet till en Androidapplikation. Under implementationen uppmärksammades dock att Androids hämtningsfrekvens på touchmove-eventet är betydligt lägre än iOS. Denna upptäckt resulterade i en avgränsning av Phonegap och domänöverskridande förfrågningar då iOS SDK (Software Development Kit) kräver en betallicens till skillnad från Androids utvecklingsverktyg som är gratis. Applikationen är istället helt webbaserad och experimentet kommer att genomföras med iOS-enheter via webbläsaren.

Mätapplikationen 3G-Watchdog är plattformspecifik för Android och kan därför inte användas under experimentet. Istället kommer skickad och mottagen data mätas och lagras separat direkt i applikationen, vilket även möjliggör spelsessioner på distans. I och med att spelare inte behöver sitta lokalt för att mätdata skall erhållas kan även flera testsviter genomföras efter experimentet.

5.1 Högfrekventa intervallanrop

Intervallanropen implementeras i applikationens turhantering. Varje spelsession består av en till fyra spelare. Användare som registreras på en spelsession tilldelas ett unikt id mellan ett och det bestämda antalet spelare. Det är användarens tur när dennes unika id matchar det officiella id som lagras i spelsessionen.

När en användare ansluter till en spelsession kontrollerar applikationen om det är dennes tur att spela eller inte. Intervallanropen initieras vid händelser där användaren lämnar ifrån sig turen till en annan spelare, exempelvis när denna spelar ett ord, passar sin tur eller byter bokstäver. Figur 1 visar den funktion som anropas i intervaller. Funktionen initierar ett AJAX-anrop för att kontrollera om det är användarens tur eller inte. Returnerar anropet data är det spelarens tur och intervallanropen avbryts, om inte fortsätter anropen som tidigare.

```
function checkTurn() {
    $.post("AJAXhandler.php", { functionCall: "checkTurn", player:
    player.playerName, game: player.gameId},
        function(data) {
            if(data) {
                player.myTurn = true;
                clearInterval(curInterval);
                getLetters(game);
            } else {
                player.myTurn = false;
            }
    });
}
```

Figur 1 Intervallanrop: AJAX-anrop från klienten

Hur AJAX-anropen hanteras på serversidan beror på den bifogade functionCall-variabeln. Variabeln beskriver för servern vilken information som skall returneras till klienten. Det är främst här som programkoden skiljer sig mellan högfrekventa intervallanrop och avvaktande AJAX-anrop. Figur 2 visar anropshanteringen då functionCall-variabeln är definierad som ”checkTurn”. Servern returnerar ”ok” om det är den anropande klientens tur, om inte returneras ett tomt svar.

```
// Check whos turn it is.
case "checkTurn":
    if (isset($_POST['game']) && isset($_POST['player'])) {
        if (checkTurn($_POST['game'], $_POST['player'])) {
            echo "ok";
        }
    }
break;
```

Figur 2 Intervallanrop: anropshantering på servern

Spelsessioner, spelare samt ordlistan lagras i en mySQL-databas. För att kontrollera om det är en spelares tur bifogas spelsessionens och spelarens unika id, se figur 2. Informationen jämförs med den som finns lagrad i databasen, sedan returneras ett svar till klienten, se figur 3. Processen, AJAX-anrop (figur 1), anropshantering (figur 2) och turkontroll (figur 3), pågår i intervaller fram tills dess att det är spelarens tur. För den fullständiga programkoden för klassiska intervallanrop, se appendix B.

```
function checkTurn($gameId, $playerName) {
    $pdo=getPDO();
    $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

    // Get player and gamesession turnId.
    $query="SELECT turn,turnId FROM playgames INNER JOIN games on
    gameId=id WHERE playerName = :USERNAME AND gameId = :GAMEID;";

    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':USERNAME', $playerName);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
    $result=$stmt->fetchAll();

    // If the turns match, it is the players turn.
    if($result[0]["turnId"] == $result[0]["turn"]) {
        return true;
    } else {
        return false;
    }
}
```

Figur 3 Intervallanrop: turkontroll på servern

5.2 Avvaktande AJAX-anrop

De avvaktande AJAX-anropen initieras, liksom intervallanropen, då en användare ansluter till en spelsession, samt när denne lämnar ifrån sig turen till en annan spelare. Hur sällan de avvaktande AJAX-anropen sker beror på serverkonfigurationen av PHP. Inställningen ”Max_execution_time” bestämmer hur länge ett script får köras innan det avbryts. Inställningens standardvärde är 30 sekunder. I arbetets serverkonfiguration är denna satt till 90 sekunder. Applikationen är dock oberoende av serverkonfigurationen och skulle tidsgränsen nås initieras ett nytt avvaktande AJAX-anrop, se figur 4.

```
function checkTurnLongpolling() {
   $.post("AJAXhandler.php", { functionCall: "checkTurnLongpolling",
   player: player.playerName, game: player.gameId},
   function(data) {
       if(data == "ok") {
           player.myTurn = true;
           getLetters(game);
       } else {
           checkTurnLongpolling();
       }
   });
}
```

Figur 4 Avvaktande AJAX-anrop: AJAX-anrop från klienten

Den primära skillnaden mellan intervallanrop och avvaktande AJAX-anrop ligger i anropshanteringen. Figur 5 visar hur scriptet inte returnerar något svar till användaren förrän dess att det är dennes tur. Istället väntar scriptet i ett förbestämt antal sekunder mellan varje kontroll innan det körs igen. Detta minskar slösad datatrafik då inga tomma svar skickas till klienten mellan varje kontroll. Turkontrollen är densamma och påverkas inte av anropstypen, se figur 3. För den fullständiga programkoden för avvaktande AJAX-anrop, se appendix C.

```
// Alternative method to check turn with long polling.
Case ”checkTurnLongpolling”:
    if (isset($_POST[’game’]) && isset($_POST[‘player’])) {
        $turn = false;
        while (!$turn) {
            if (checkTurn($_POST[‘game’], $_POST[‘player’])) {
                echo “ok”;
                $turn = true;
            } else {
                sleep($intervalTime);
            }
        }
    }
break;
```

Figur 5 Avvaktande AJAX-anrop: anropshantering på servern

6 Testupplägg

Under experimentet kommer testkandidaterna köra totalt fem spelsessioner, två sessioner per version med varierad anropsfrekvens. Anropsfrekvensen påverkar den mängd datatrafik som genereras, vilket gör det intressant att veta hur sällan anrop kan ske. Genom att under spelsessionerna använda en varierad anropsfrekvens, mellan en och tio sekunder, testas hur sällan anrop kan ske utan att användarens upplevelse av spelet påverkas.

Tabell 1 visar vilken anropsmetod samt anropsfrekvens testkandidaten skall använda under respektive spelomgång. Varje omgång, utom den femte, delar två testkandidater samma fördröjning vilket gör att likheter i svaren under intervjun bör finnas. Om det inte är fallet kan testkandidaten ha påverkats av en utomstående faktor exempelvis mobilenheten, mobiloperatören eller webbläsaren, alternativt att en fördröjning på tio sekunder är acceptabelt för denna typ av applikation.

Tabell 1 Spelsessioner med anropsfrekvens i sekunder.

Spelomgång	Testkandidat 1	Testkandidat 2	Testkandidat 3	Testkandidat 4
1	A:3	I:3	A:1	I:5
2	I:7	A:5	I:5	A:3
3	A:1	A:10	A:7	A:7
4	I:10	I:1	I:3	I:10
5	A:5	A:7	I:10	I:1

I = intervallanrop, A = avvaktande AJAX-anrop.

Syftet med intervjun är att få reda på under vilka omgångar testkandidaten uppfattade att anropsfrekvensen ändrades till det bättre respektive sämre. Intervjun efter experimentet kommer att innehålla följande frågor:

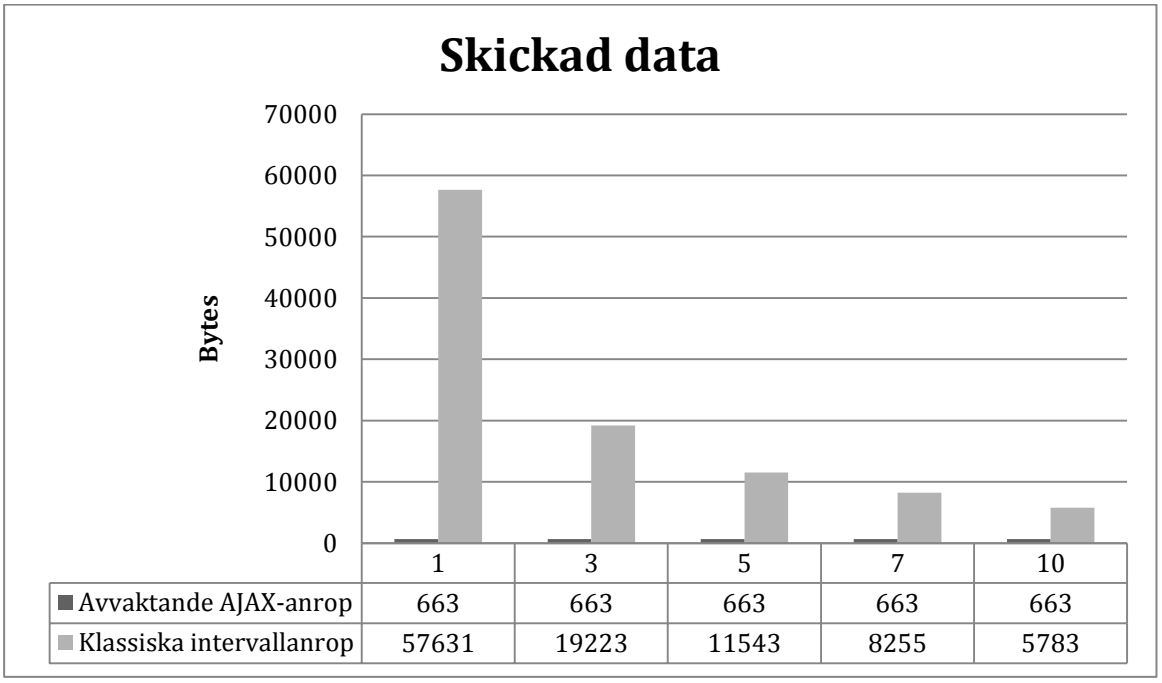
- Du har under experimentet spelat fem omgångar. Märkte du någon skillnad mellan dessa?
- Om ja, vilken spelomgång tyckte du fungerade bäst respektive sämst?
- Hur viktigt är det att ord som motståndare har spelat syns direkt?
- Vilken mobilenhet använde du under experimentet?
- Vilken webbläsare använde du under experimentet?
- Vilken mobiloperatör har du?

7 Analys

Experimentets primära syfte var att besvara användbarhetsrelaterade frågor kring fördröjning, men resulterade även i 20 stycken mätresultat. Respektive spelsession varade i 15minuter, vilket visade sig vara för lite för att presentera en korrekt bild av anropsmetodernas differens. Detta ledde till vidare testning med längre spelsessioner där speltiden istället varierade mellan 56 och 130minuter. Den fortsatta testningen ledde till 17 nya spelsessioner och resulterade i totalt 50 stycken nya mätresultat. Graferna är baserade på ett medelvärde av dessa resultat, se Appendix A.

Anropens differens beräknades först i en spelsession utan några klientgenererade uppdateringar, vilket då endast består av overheaddata, se figur 6. Informationen som bifogas är minimal och innehåller endast data för att identifiera en förändring i spelsessionen, vilket mäts till 4 byte data per anrop. Mätdata samlades in under två timmar där avvaktande AJAX-anrop, oavsett intervall, använder 663byte data. Under samma tid genererar de klassiska intervallanropen som minst 772,25% mer overheaddata än de avvaktande AJAX-anropen.

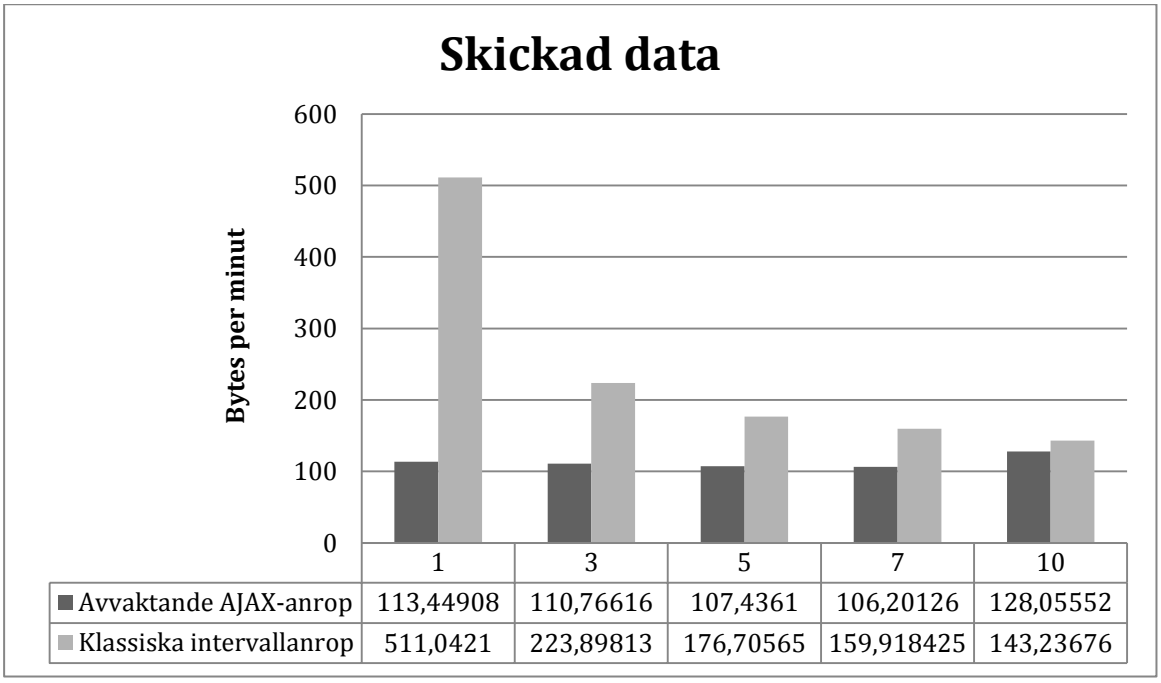
Sanajaya (2011) mäter i sitt arbete anropsmetodernas differens men separerar inte skickad och mottagen data. Då anropsmetoden inte påverkar mottagen data kan denna, beroende på storlek, komma att visa en felaktig bild av metodernas differens. Genom att isolera skickad data från klienten kunde en mer precis mätning av klientgenererad data genomföras.



Figur 6 Applikationsgenererad overheaddata i bytes

Under en spelsession adderas overheaddata med övrig klientgenererad data, exempelvis kontroll av ord och bokstavsbyte. Detta gör det möjligt att se om overheaddata utgör en markant del av applikationens klientgenererade bandbreddskonsumtion. Analysen är baserad på 50st mätresultat fördelat över två applikationer med olika anropsfrekvenser. Mätresultaten bekräftar minskningen av skickad data med avvaktande AJAX-anrop och beskriver hur applikationens datatrafik påverkas av olika anropsmetoder och frekvens.

Vid jämförelse av högfrekventa anrop med en sekunds intervaller genererar avvaktande AJAX-anrop 350% mindre skickad data än klassiska intervallanrop. Längre intervaller minskar stegvis klientgenererad data för de klassiska intervallanropen, se figur 7. Vid tio sekunders intervaller genererar avvaktande AJAX-anrop 10,6% mindre skickad data. Dock differerar det avvaktande AJAX-anropets mätdata från övriga anrop av samma metod med minst 11,4%. Under arbetets testsviter erhölls några mätresultat som presenterar en felaktig bild av anropens differens, exempelvis på avvaktande AJAX-anrop med tio sekunders intervallar. Dessa skulle jämnas ut genom att utföra fler tester. Tanken var att under arbetet erhålla minst 100 stycken mätresultat, men i och med den ökade speltiden var detta inte längre möjligt.



	1	3	5	7	10
Avvaktande AJAX-anrop	113,44908	110,76616	107,4361	106,20126	128,05552
Klassiska intervallanrop	511,0421	223,89813	176,70565	159,918425	143,23676

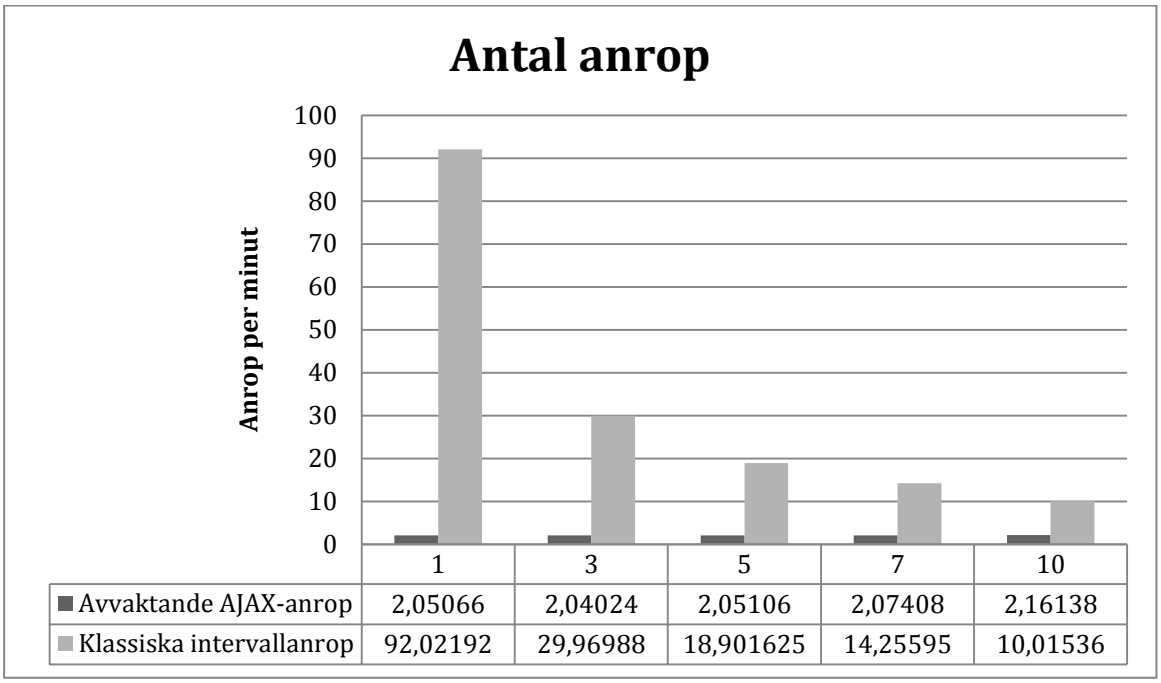
Figur 7 Skickad data i bytes per minut

Tidigare tester av avvaktande AJAX-anrop har främst implementerats i realtidsapplikationer där uppdateringar nödvändigtvis inte sker speciellt ofta, men vars realtidsfaktor är av större vikt. Dessa applikationer innefattar bland annat börskurslistor, auktionssajter och chattrum. Fokus i dessa arbeten har dock inte legat på differens av genererad datatrafik, utan serverlastning och ankomst av skickade paket under hög belastning (Bozdag, et al., 2007; Bozdag & van Deursen, 2008). Sanjaya (2011) såg däremot den minskade datatrafiken som en möjlighet för anropsmetoden att appliceras i applikationer för mobila enheter, vars bandbreddskonsumtion är begränsad.

Detta arbetets primära syfte är detsamma som Sanjaya's, att jämföra de olika anropsmetodernas differens i bandbreddskonsumtion. Skillnaden är att arbetets jämförelse av metoderna baseras på en realistisk implementation med användargenererade uppdateringar, vilket avlägsnar frågor kring validitet av slumpmässigt datorgenererade uppdateringsfrekvenser.

Förutom bandbreddskonsumtion är antal anrop en viktig faktor i utveckling för mobila enheter. Att minimera antalet anrop är en av huvudpunkterna i W3C rekommendation för utveckling till mobilplattformar (W3C, 2010). Avvaktande AJAX-anrop minskar inte bara

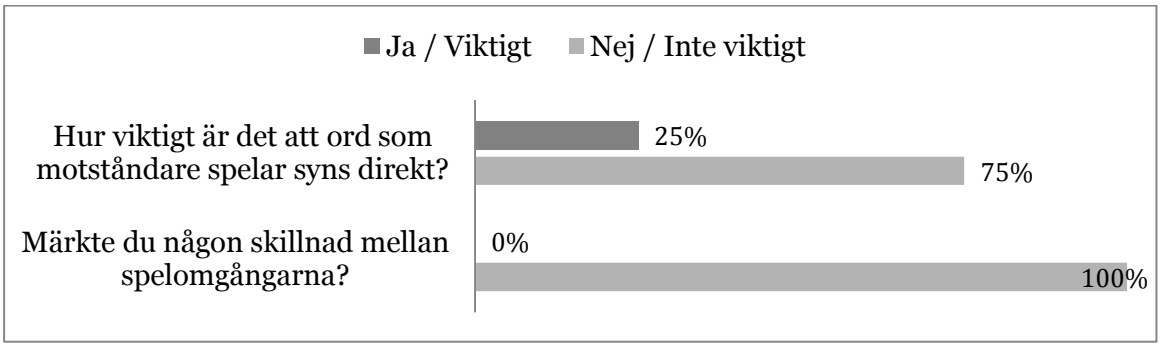
klientgenererad data utan även antalet server anrop. I denna implementation minskas antalet serveranrop med minst 363,4%, se figur 8.



	1	3	5	7	10
Avvaktande AJAX-anrop	2,05066	2,04024	2,05106	2,07408	2,16138
Klassiska intervallanrop	92,02192	29,96988	18,901625	14,25595	10,01536

Figur 8 Antal anrop per minut

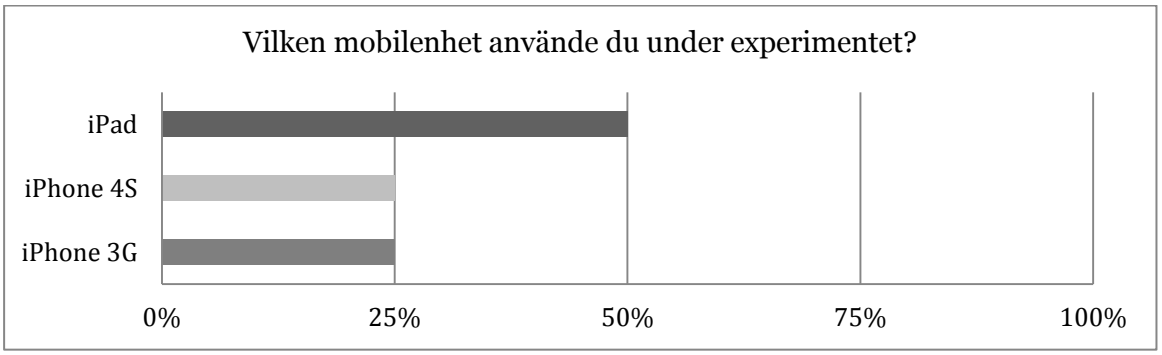
En delfråga i problemformuleringen var hur sällan intervallanrop kan ske utan att användaren uppfattar att applikationen inte sker i realtid. Denna fråga besvarades genom individuella intervjuer som gjordes efter experimentets genomförande. Ingen av testkandidaterna hade uppfattat någon skillnad mellan de olika applikationerna, vilket påvisar att tio sekunder är en acceptabel fördröjning för denna typ av applikationer. Den fullständiga dokumentationen för intervjuerna kan ses i appendix E, F, G och H.



Figur 9 Sammanställning av svaren från de användbarhetsrelaterade intervjufrågorna

När testkandidaterna fick frågan om hur viktigt det är att ord som motståndare spelar syns direkt var det mer delade svar, se figur 9. I en aktiv applikation, som spelas i en sittning, är realtid en viktigare faktor än i en passiv applikation som spelas över ett par dagar. Majoriteten tyckte att anropsfrekvensen var av mindre vikt medan andra tyckte att den var viktig om ett svar väntades inom kort, men inte om telefonen låg i fickan. Detta återkopplar till ett förlag som nämndes i W3Cs officiella rekommendation för utveckling till

mobilplattformar där användarens aktivitet monitoreras och uppdateringsfrekvensen bestäms därefter (W3C, 2010).



Figur 10 Sammanställning av de mobila enheter som användes under experimentet

Under experimentets genomförande användes tre olika typer av mobila enheter: iPhone 4S, iPhone 3G och iPad, se figur 10. Samtliga enheter hade operativsystemet iOS och använde webbläsaren Safari. Operatörerna som användes under experimentet var Tele2, Telia, Halebop och Telenor.

8 Slutsatser

8.1 Resultatsammanfattning

Arbetets syfte var att se hur en applikations datatrafik påverkas om högfrekventa intervallanrop ersätts av avvaktande AJAX-anrop, och hur sällan dessa anrop kan ske. För att undersöka detta utvecklades två nästintill identiska alfapetapplikationer, den ena använder avvaktande AJAX-anrop och den andra klassiska intervallanrop. Varje applikation finns i fem olika versioner med olika anropsfrekvens. Syftet med applikationen var att erhålla klientgenererade uppdateringar, samt en realistisk mängd overheaddata. Detta för att kunna mäta anropsmetodernas differens i använd datatrafik.

För att erhålla mätdata genomfördes ett experiment med fyra testkandidater. Varje kandidat spelade fem spelsessioner med olika anropsfrekvens. Under experimentet varade varje session i 15 minuter, vilket visade sig vara för lite för att presentera en korrekt bild av anropsmetorenas differens. Detta ledde till vidare testning efter experimentet med längre spelsessioner. Dessa tester resulterade i 50 stycken mätresultat och påvisar att avvaktande AJAX-anrop, i denna implementation, minskar klientgenererad datatrafik. Beroende på anropsfrekvens, minskar denna med minst 10,6% (tio sekunders intervaller) och som mest 350% (en sekunds intervaller).

Efter experimentet utfördes en intervju för att besvara arbetets användbarhetsrelaterade delfråga. För att se eventuella likheter i de individuella intervjuerna hade två användare samma anropsfrekvens i alla utom den sista spelsessionen. Detta för att eventuellt kunna se likheter i testkandidaternas svar under intervjun. Dock upplevde inte någon av testkandidaterna någon skillnad mellan de olika spelsessionerna, trots den varierande uppdateringsfrekvensen. Detta påvisar att en anropsfrekvens på tio sekunder är acceptabelt för denna typ av applikation. En fråga i intervjun pekade på hur viktigt det var att ord som spelas syns direkt. Majoriteten av testkandidaterna hade, i denna typ av applikation, godtagit en ännu längre anropsfrekvens.

8.2 Diskussion

För att erhålla ett verifierbart resultat är det viktigt att samtliga mätresultat och dess testplattform redovisas. Arbetet redovisar samtliga mätresultat, som även finns tillgängliga i appendix A, samt de mobila enheter som användes. Samtliga intervjuers dokumentation finns tillgänglig i appendix E, F, G och H, vilket gör dessa lätta att upprepa. Även testapplikationernas programkod, som användes för att erhålla mätresultaten, finns tillgängliga i appendix B och C. Strukturen för applikationens databas kan ses i appendix D. För att arbetet ska klassas som forskningsetiskt verifierbars redovisas alla moment som utförts för att nå detta resultat.

Som tidigare nämnts behövdes längre spelsessioner för att visa en korrekt bild av anropsmetodernas differens. Detta medförde relativt få testresultat vilket gör mätresultaten något osäkra. Ett exempel på detta är avvaktande AJAX-anrop med tio sekunders intervaller som avviker från övriga resultat av samma anropstyp med minst 11,4%. En faktor som kan ha påverkat mätresultaten är respektive klients spelstil, exempelvis antal testade ord per tur. En klient som testar fler ord per tur genererar givetvis mer data än en klient som spelar ut ett giltigt ord på första försöket.

Intervjuerna besvarade arbetets användbarhetsrelaterade delfråga om hur sällan anrop kan ske utan att användaren märker att dessa inte sker i realtid. Ytterligare intervjuer behöver dock utföras med en bredare grupp av deltagare. Detta då samtliga intervjudeltagare är av både lika kön och ålder, vilket leder till frågor kring hur faktorer som genus och ålder kan ha påverkat resultatet.

När arbetet påbörjades var tanken att ta en mobilapplikation, som i dagsläget genererar mycket datatrafik, och se hur olika anropsmetoder påverkar dess datatrafik. I detta steg var simulering av realtid en viktig faktor, denna har dock succesivt minskat under arbetets gång. Grundtanken var att använda en tidskvot för att stressa spelarna, vilket förväntades göra realtidsfaktorn mer essentiell. Detta visade sig dock inte vara fallet eftersom användarens spelupplevelse inte påverkas speciellt mycket av applikationens anropsfrekvens då denna i grund och botten är turbaserad. Bozdag och van Deursen (2008) samt Bozdag et al, (2007) nämner flera applikationer vars uppdateringar inte sker ofta men där realtidsfaktorn är desto viktigare; bland dessa finns börskurslistor, chattprogram och auktionssajter. Avvaktande AJAX-anrop skulle med stor sannorlikhet minska datatrafiken i dessa applikationer markant, vilket gör dem mindre intressanta att undersöka. Det som gjorde en vanlig applikation intressant var just att se om tekniken kunde implementeras i vilken applikation som helst för att spara datatrafik.

Det finns i dagsläget flera applikationer utvecklade för både iOS och Android där realtid är en viktig faktor. Bland dessa finns, som tidigare nämnt, chattprogram och auktionsapplikationer. För att uppnå realtid har det alltid varit naturligt att använda push-tekniker och hålla en öppen anslutning mellan klient och server (Bozdag, et al., 2007). Dessa applikationer tjänar troligtvis mer på att använda en implementation av WebSockets (Gutwin, Lippold & Graham, 2011). För andra applikationer, där realtid inte är av lika stor vikt, kan dock avvaktande AJAX-anrop vara ett lämpligt alternativ.

Datatrafiken i 3G-nätet kommer enligt Choi, et al. (2011) troligtvis att öka ytterligare mellan år 2009-2014. Då 3G-nätet enligt Choi, et al. (2011), Balasubramanian, et al. (2010), Ristanovic, et al. (2011), Stenquist (2011), Efendić (2011) samt Andersson och Åberg (2011) redan är överbelastat och ytterligare belastning är beräknad skulle avvaktande AJAX-anrop kunna fungera som en lämplig avlastningsmetod på applikationsnivå. Andra förslag för att avlasta 3G-nätet är främst riktade mot alternativa uppkopplingar, exempelvis trådlösa nätverk (Choi, et al., 2011; Balasubramanian, et al., 2010; Ristanovic, et al., 2011; Zhuo, et al., 2011). Sanjaya (2011) testar i sitt arbete avvaktande AJAX-anrop i avlastningssyfte med en automatiserad process. Detta arbete tillsammans med Sanjayas visar på en vinst i bandbreddskonsumtion vid användning av avvaktande AJAX-anrop med både automatiska och klientgenererade uppdateringar. Detta gör att avvaktande AJAX-anrop, tillsammans med utbyggnad av 3G-nätet och alternativa uppkopplingar, kan bidra med avlastning till den nuvarande och kommande belastningen i 3G-nätet.

8.3 Framtida arbete

För att presentera en mer korrekt bild av anropsmetodernas differens behövs mer data samlas in. Även algoritmen för att skicka ut uppdateringar från servern till klienten skulle behöva optimeras, förslagsvis genom att lagra alla uppdateringar i en numrerad lista och endast skicka ut de uppdateringar klienten saknar. Detta skulle göra applikationens

mottagna data jämförbar med dess skickade data. Just nu är applikationens mottagna data för stor för att dessa på ett rättvist sätt ska kunna jämföras.

Något som saknas i arbetet är en jämförelse med en ren push-teknik, exempelvis WebSockets (Gutwin, et al., 2011). Även om realtidsfaktorn succesivt blivit en mindre del av arbetet skulle undersökningen vara mer intressant om jämförelsen gjordes mellan en pull-, push- och en hybrid-teknik. Detta då respektive anropsmetod har olika för- och nackdelar vilket skulle göra det intressant att även monitorera servern hårdvara under testerna (Bozdag & van Deursen, 2008; Bozdag, et al., 2007). Ett naturligt steg vidare i arbetet vore att med Phonegap konvertera applikationen till en plattformspecifik applikation. Genom detta skulle fler faktorer som spelar roll vid utveckling för mobila enheter kunna jämföras, exempelvis batterianvändning och aktivitet. W3C (2010) ger exempelvis ett förslag att monitorera användarens aktivitet och därefter bestämma anropsintervall. Denna teknik skulle kunna implementeras i klassiska intervallanrop för att sedan jämföra med avvaktande AJAX-anrop.

Det största problemet med avvaktande AJAX-anrop är dess belastning på servern. För varje anrop som sker skapas en ny tråd i serverns processor, vilket innebär att servern har en tråd per ansluten klient. Detta leder i sin tur till att anropens exekveringstid måste begränsas för att kontrollera om klienten fortfarande är ansluten eller inte. Är klienten inte längre ansluten till spelsessionen kan tråden tas bort från processorn eftersom denna inte längre är aktiv. Om ingen maximal exekveringstid definieras kommer inte gamla trådar att tas bort från processorn. Den maximala exekveringstiden är den som bestämmer hur sällan anrop behöver ske, men är även den som bestämmer hur länge en död tråd får ligga kvar i processorn. Som det ser ut idag är den maximala exekveringstiden en statisk inställning i PHPs konfigurationsfil, vilket innebär att alla användare måste ha samma maximala exekveringstid. Ett alternativ vore att ge varje klient en individuell maximal exekveringstid. Denna skulle sedan kunna baseras på historik, exempelvis klientens genomsnittliga speltid. För en ny klient skulle den maximala exekveringstiden kunna börja på 90 sekunder men succesivt ökas allt eftersom klientens spelsession pågår.

Referenser

Andersson, G. & Åberg, R. (2011) *Telia och Tele2 tagna på sängen av mobilsurfandet | Feber / Mobil*. 2011. Tillgänglig på Internet: http://feber.se/mobil/art/224104/telia_och_tele2_tagna_p_sngen_/ [Hämtad 3 februari, 2012].

Balasubramanian, A., Mahajan, R. & Venkataramani, A. (2010) *Augmenting Mobile 3G Using WiFi: Measurement, System Design, and Implementation*.

Bozdag, E. & van Deursen, A. (2008) An Adaptive Push/Pull Algorithm for AJAX Applications. *2008 Eighth International Conference on Web Engineering, ICWE 2008*. 14 July Yorktown Heights, New York,. s. 95–100.

Bozdag, E., Mesbah, A. & van Deursen, A. (2007) A Comparison of Push and Pull Techniques for AJAX. *9th IEEE International Workshop on Web Site Evolution, 2007. WSE 2007*. 5 October IEEE. s. 15–22.

Böhmer, M., Hecht, B., Schöning, J., Krüger, A., Bauer, G. (2011) Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. MobileHCI '11. New York, NY, USA, ACM. s. 47–56.

Charland, A. & LeRoux, B. (2011) Mobile Application Development: Web vs. Native. *Queue*. 9 (4), s. 20:20–20:28.

Choi, Y., Ji, H.W., Park, J.Y., Kim, H., Silvester, J. (2011) A 3W network strategy for mobile data traffic offloading. *Communications Magazine, IEEE*. 49 (10), s. 118 –123.

Efendić, N. (2011) *3G-nät kraftigt överbelastat | IT | SvD*. 2011. Tillgänglig på Internet: http://www.svd.se/naringsliv/it/appar-som-wordfeud-pastas-sanka-3g-nat_6508520.svd [Hämtad 16 februari, 2012].

Google (2012) *3G Watchdog - Appar på Android Market*. 2012. Tillgänglig på Internet: https://market.android.com/details?id=net.rgruet.android.g3watchdog [Hämtad 3 februari, 2012].

Gutwin, C.A., Lippold, M. & Graham, T.C.N. (2011) Real-time groupware in the browser: testing the performance of web-based networking. *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. CSCW '11. New York, NY, USA, ACM. s. 167–176.

Haeran Kim, Haejin Kim, Moonsuk Jang, Soonhee Han, *et al.* (2010) A Comparison of Features of Ajax's Data Formats for the Medicinal Plants Application. *2010 2nd International Conference on Information Technology Convergence and Services (ITCS)*. 11 August IEEE. s. 1–6.

Hauswirth, M. & Jazayeri, M. (1999) A component and communication model for push systems. *SIGSOFT Softw. Eng. Notes*. 24 (6), s. 20–38.

Kasemvilas, S. & Firpo, D. (2009) Human Interface and the Management of Information. Information and Interaction. Lecture Notes in Computer Science. Springer Berlin / Heidelberg. s. 45–54.

Liew, J.S.Y., Kaziunas, E., Liu, J. & Zhuo, S. (2011) Socially-interactive dressing room: an iterative evaluation on interface design. *Proceedings of the 2011 annual conference extended*

abstracts on Human factors in computing systems. CHI EA '11. New York, NY, USA, ACM. s. 2023–2028.

Paulson, L.D. (2005) Building rich web applications with Ajax. *Computer*. 38 (10), s. 14– 17.

Ristanovic, N., Le Boudec, J., Chaintreau, A. & Erramilli, V. (2011) Energy Efficient Offloading of 3G Networks. *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*. October s. 202 –211.

Sallnäs, E.-L. (2007) *Beteendevetenskaplig metod Intervjuteknik och analys av intervjudata*. 2007. Tillgänglig på Internet: http://www.itn.liu.se/mit/education/courses/tnfl05-risk-och-olycksanalys/vecka-48/1.306165/Intervjuteknik07.pdf [Hämtad 21 mars, 2012].

Sanjaya, R. (2011) Mobile traffic evaluation for fuzzy-based application using Green Ajax. *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*. 27 May IEEE. s. 409–416.

Si Hai-ping, Qiao Yang, Fang Wei & Cao Yong-sheng (2010) Design of cross-domain query of crop germplasm resources investigation platform. *2010 International Conference on Educational and Information Technology (ICEIT)*. 17 September IEEE. s. V2–219-V2–221.

Stenquist, V. (2011) *3G-nätet sänkt – av Wordfeud | Nyheter | Aftonbladet*. 2011. Tillgänglig på Internet: http://www.aftonbladet.se/nyheter/article13698193.ab [Hämtad 16 februari, 2012].

van Velsen, L., van der Geest, T. & Klaassen, R. (2007) Testing the usability of a personalized system: comparing the use of interviews, questionnaires and thinking-aloud. *Professional Communication Conference, 2007. IPCC 2007. IEEE International*. 1 October IEEE. s. 1–8.

W3C (2010) *Mobile Web Application Best Practices*. 2010. Tillgänglig på Internet: http://www.w3.org/TR/mwabp/#bp-conserve-requests [Hämtad 7 februari, 2012].

W3C (2006) *The XMLHttpRequest Object*. 2006. Tillgänglig på Internet: http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060619/ [Hämtad 3 februari, 2012].

Xuejun Zhuo, Wei Gao, Guohong Cao & Yiqi Dai (2011) Win-Coupon: An incentive framework for 3G traffic offloading. *2011 19th IEEE International Conference on Network Protocols (ICNP)*. 17 October IEEE. s. 206–215.

Appendix A - Mätresultat

Anropsmetod och frekvens	Tid	Tur	Skickad data per minut	Mottagen data per minut	Anrop per minut	Skickad data	Mottagen data	Antal anrop
A10_#1	82	53	125.3049	14770.9024	2.2317	10275	1211214	183
A10_#2	78	49	104.1923	14374.3205	1.9744	8127	1121197	154
A10_#3	80	54	126.6625	15022.7625	2.1000	10133	1201821	168
A10_#4	82	51	125.4512	14071.1341	2.1220	10287	1153833	174
A10_#5	66	49	158.6667	16677.1970	2.3788	10472	1100695	157
A1_#1	130	66	142.3231	11394.0615	2.0231	18502	1481228	263
A1_#2	89	58	120.4607	13196.1573	2.3146	10721	1174458	206
A1_#3	76	48	106.4342	14227.4605	2.0526	8089	1081287	156
A1_#4	73	41	91.7945	12650.5479	1.8493	6701	923490	135
A1_#5	73	42	106.2329	12549.7671	2.0137	7755	916133	147
A3_#1	82	53	129.4146	14997.3537	2.0732	10612	1229783	170
A3_#2	78	49	84.8718	14374.1410	1.9615	6620	1121183	153
A3_#3	75	48	133.4667	14683.5200	2.1600	10010	1101264	162
A3_#4	73	41	96.6027	12647.8767	1.9315	7052	923295	141
A3_#5	80	54	109.4750	15359.7750	2.0750	8758	1228782	166
A5_#1	82	53	105.2927	14769.5488	2.2073	8634	1211103	181
A5_#2	78	49	88.2308	14624.7179	2.0128	6882	1140728	157
A5_#3	75	48	107.8133	14418.2000	2.0400	8086	1081365	153
A5_#4	73	41	134.3562	12649.4247	1.9452	9808	923408	142
A5_#5	80	54	101.4875	15353.6625	2.0500	8119	1228293	164
A7_#1	82	53	132.7195	14766.8659	2.2561	10883	1210883	185
A7_#2	78	49	97.0256	14373.2179	2.0000	7568	1121111	156

A7_#3	80	54	104.7125	15363.2250	2.1875	8377	1229058	175
A7_#4	82	51	103.4146	14070.2195	1.9878	8480	1153758	163
A7_#5	82	51	93.1341	14064.9024	1.9390	7637	1153322	159
I10_#1	66	49	141.7273	16675.3182	9.3030	9354	1100571	614
I10_#2	89	58	136.2135	13400.0674	11.0337	12123	1192606	982
I10_#3	76	48	160.4211	14237.3816	9.6579	12192	1082041	734
I10_#4	73	41	131.5068	12649.0000	10.0548	9600	923377	734
I10_#5	73	42	146.3151	12553.3288	10.0274	10681	916393	732
I1_#1	56	48	563.1429	24437.5357	91.8571	31536	1368502	5144
I1_#2	56	48	520.6429	24467.7321	87.5000	29156	1370193	4900
I1_#3	82	53	514.8659	18478.6585	97.0488	42219	1515250	7958
I1_#4	76	47	506.5263	17665.2763	96.3947	38496	1342561	7326
I1_#5	83	52	486.8554	17833.4096	89.0843	40409	1480173	7394
I1_#6	73	42	474.2192	15549.3973	90.2466	34618	1135106	6588
I3_#1	111	68	244.6937	13368.6306	23.4234	27161	1483918	2600
I3_#2	82	53	207.0854	14861.4024	30.6829	16981	1218635	2516
I3_#3	76	47	232.1053	14093.0526	31.5263	17640	1071072	2396
I3_#4	83	52	203.9880	14280.5663	30.4337	16931	1185287	2526
I3_#5	73	42	207.2740	12550.3973	30.3288	15131	916179	2214
I3_#6	66	49	248.2424	16670.2727	33.4242	16384	1100238	2206
I5_#1	82	53	159.2561	14862.2317	18.7805	13059	1218703	1540
I5_#2	76	47	173.0658	14088.7763	17.4211	13153	1070747	1324
I5_#3	83	52	180.8795	14282.6145	20.1928	15013	1185457	1676
I5_#4	66	49	193.6212	16675.5152	19.2121	12779	1100584	1268
I7_#1	89	58	168.4157	13422.6180	15.4382	14989	1194613	1374

I7_#2	82	53	169.7561	14536.4268	13.7561	13920	1191987	1128
I7_#3	76	47	146.2368	14087.1579	14.2632	11114	1070624	1084
I7_#4	83	52	155.2651	14280.1205	13.5663	12887	1185250	1126
overhead_A1	120	4	5.5250	141.6083	0.7167	663	16993	86
overhead_A10	120	4	5.5250	141.6083	0.7167	663	16993	86
overhead_A3	120	4	5.5250	141.6083	0.7167	663	16993	86
overhead_A5	120	4	5.5250	141.6083	0.7167	663	16993	86
overhead_A7	120	4	5.5250	141.6083	0.7167	663	16993	86
overhead_I1	120	4	493.3333	141.6250	120.3500	59200	16995	14442
overhead_I10	120	4	48.1917	141.5667	12.0500	5783	16988	1446
overhead_I3	120	4	160.1917	141.5750	40.0500	19223	16989	4806
overhead_I5	120	4	96.1917	141.6167	24.0500	11543	16994	2886
overhead_I7	120	4	68.7917	141.5833	17.2000	8255	16990	2064

Appendix B - Klassiska intervallanrop

Index.html

```
<html>
<head>
  <title>SpeedScrabble</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js
"></script>
  <script type="text/javascript" src="script.js"></script>
</head>
<body>
<div id="loginForm">
  <input type="text" id="playername" class="big" /><button
id="login" class="big">login</button>
</div>
<div id="gameList"></div>
<canvas id="app" width="320" height="373" tabindex="1">You no see
canvas?</canvas>
<div id="wrapper">
  <div id="score"></div>
  <div>
    <button id="menu" class="button">Menu</button>
    <button id="pass" class="button">Pass</button>
    <button id="swap" class="button">Swap</button>
    <button id="play" class="button">Play</button>
  </div>
  <div id="msg"></div>
</div>
</body>
</html>
```

Script.js

```
window.onload = function(){
  game = new gameBoard();
  player = new playerClass("",500,"");
  (typeof window.innerWidth == "number" && window.innerWidth > 500)
? sizeModifier = (window.innerHeight)/430: sizeModifier = 1;
  document.getElementById('login').addEventListener("click", login,
false);
  document.body.addEventListener('touchmove',function(event) { //
prevent elastic scrolling
    event.preventDefault();
  },false);  // end body:touchmove

  setInterval(draw, 30);
  $("#app").hide();
  $("#wrapper").hide();
  var dragok,                      // Enables letterdragging.
     canvas = document.getElementById('app'),     // Canvas element.
     context = canvas.getContext('2d'),         // Canvas context.
     request,                    // AJAX-call for abort().
     moveObject,                   // The letter that is being
dragged.
     mouseX = 0,                   // Global variable for current
mousePosition X.
     mouseY = 0,                   // Global variable for current
mousePosition Y.
     slotWidth = 21.3*sizeModifier,           // Width of tiles.
     slotHeight = 23.3*sizeModifier,         // Height of tiles.
     letterWidth = 20.3*sizeModifier,         // Width of letters.
     letterHeight = 22.3*sizeModifier,         // Height of letters.
     playerframeWidth = slotWidth*15,
     tileFont = 10*sizeModifier+"px Arial",
     letterFont = 14*sizeModifier+"px Arial",
     letterValFont = "bold "+9*sizeModifier+"px arial",
     swapFont = "bold "+12*sizeModifier+"px Arial",
     fontMod = 5*sizeModifier,
     fontModY = 15*sizeModifier,
    intervalTime = 1000;              // CheckTurn intervaltime.

   // Modify size of frames and fonts
   canvas.width = 320*sizeModifier;
   canvas.height = 373*sizeModifier;
   document.getElementById('wrapper').style.width =
320*sizeModifier;
   $("#score").css("font-size", Math.round(11*sizeModifier));
   $("#wrapper").css("height", Math.round(45*sizeModifier));
   $(".big").css("min-height", Math.round(40*sizeModifier));
   $(".big").css("width", playerframeWidth);
```

```
   $("#gameList").css("width", playerframeWidth);
   $(".button").css("font-size", Math.round(13*sizeModifier));
   $(".button").css("padding", Math.round(sizeModifier));
   $("#msg").css("font-size", Math.round(11*sizeModifier));

  // Prototype for rectangle with rounded corners.
  CanvasRenderingContext2D.prototype.roundRect = function (x, y, w,
h, r) {
      if (w < 2 * r) r = w / 2;
      if (h < 2 * r) r = h / 2;
      this.beginPath();
      this.moveTo(x+r, y);
      this.arcTo(x+w, y,   x+w, y+h, r);
      this.arcTo(x+w, y+h, x,   y+h, r);
      this.arcTo(x,   y+h, x,   y,   r);
      this.arcTo(x,   y,   x+w, y,   r);
      this.closePath();
      return this;
  }

  function checkProperty(x, y) {
    // All DW tiles.
    if ((x == 1 && y == 1) || (x == 2 && y == 2) || (x == 3 && y ==
3) || (x == 4 && y == 4
       || (x == 13 && y == 1) || (x == 12 && y == 2) || (x == 11 &&
y == 3) || (x == 10 && y == 4)
       || (x == 13 && y == 13) || (x == 12 && y == 12) || (x == 11
&& y == 11) || (x == 10 && y == 10)
       || (x == 1 && y == 13) || (x == 2 && y == 12) || (x == 3 && y
== 11) || (x == 4 && y == 10))){
      return "DW";

    // All TW tiles.
    } else if ((x == 0 && y == 0) || (x == 0 && y == 7) || (x == 0
&& y == 14) || (x == 7 && y == 0) ||
           (x == 14 && y == 0) || (x == 14 && y == 7) || (x == 14 &&
y == 14) || (x == 7 && y == 14)) {
      return "TW"
    }

    // All DL tiles.
    else if ((x == 3 && y == 0) || (x == 11 && y == 0) || (x == 6 &&
y == 2) || (x == 8 && y == 2
      || (x%7 == 0 && y == 3) || (x == 2 && y == 6)
       || (x == 12 && y == 6) || (x == 6 && y == 6) || (x == 8 && y
== 6) || (x == 3 && y == 7)
       || (x == 11 && y == 7) || (x == 2 && y == 8) || (x == 6 && y
== 8) || (x == 8 && y == 8)
```

```
       || (x == 12 && y == 8) || (x == 0 && y == 11) || (x == 7 && y
== 11) || (x == 14 && y == 11)
       || (x == 6 && y == 12) || (x == 8 && y == 12) || (x == 3 && y
== 14) || (x == 11 && y == 14))){
      return "DL";


    // All TL tiles.
    } else if ((x == 5 && y == 1) || (x == 9 && y == 1) || (x == 1
&& y == 5) || (x == 5 && y == 5)
      || (x == 9 && y == 5) || (x == 13 && y == 5) || (x == 1 && y
== 9) || (x == 5 && y == 9)
      || (x == 9 && y == 9) || (x == 13 && y == 9) || (x == 5 && y
== 13) || (x == 9 && y == 13)) {
      return "TL"
    } else {
      return "";
    }
  }

  function playerClass(name, timequota, gameId) {
    this.letters = [];       // Player letters.
    this.swapLetters = [];    // Letters to be swapped.
    this.timequota = timequota; // Player timequota.
    this.score = 0;       // Playerscore.
    this.playerName = name;   // Playername.
    this.gameId = gameId;     // Id of current game.
    this.myTurn = false;     // Is it my turn?
    this.turnCount = 0;      // What turn is it?
  }

  function getPlayerLetters() {
    $.post("AJAXhandler.php", { functionCall: "getPlayerLetters",
game: player.gameId, player: player.playerName},
    function(data) {
      if (data && data != "") {
        returnData = JSON.parse(data);
        for (i = 0; i < returnData.length; i++) {
          player.letters.push(returnData[i]);
        }
      } else {
        giveLetters(7);
      }
    });
  }

  function tile(rx, ry, x, y, p) {
    this.letter = null; // The letter.
```

```
    this.property = p;   // TW DW TL DL
    this.posX = x;    // To find words.
    this.posY = y;    // To find words.
    this.realX = rx;  // For drag & drop.
    this.realY = ry;  // For drag & drop.
  }

  function letter(letter, val, active) {
    this.letter = letter;
    this.val = val;
    (active) ? this.active = false : this.active = true;
    this.realX = 0;
    this.realY = 0;
  }

  function findWord(firstPos, lastPos, xPos, yPos, vertical,
horizontal, tile) {
    function sortXpos(a, b) { return a.posX - b.posX; }
    function sortYpos(a, b) { return a.posY - b.posY; }
    var activeLetters = [],
      vertical = vertical,
      horizontal = horizontal,
      firstPos = firstPos,
      lastPos = lastPos,
      yPos = yPos,
      xPos = xPos,
      findFirstLetter = false,
      findLastLetter = false,
      connected = true,
      word = "",
      points = 0,
      multiplier = 1;
    activeLetters.push(tile);

    if (horizontal) {
      while (!findFirstLetter || !findLastLetter) {
        for (var i = 0, len = game.tiles.length; i < len; i++) {
          if (firstPos == -1) findFirstLetter = true; // Dont look
for first letter is an activeletter is at end of board.
          if (game.tiles[i].posX == firstPos && game.tiles[i].posY
== yPos) {
            if (game.tiles[i].letter != null) {
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              firstPos--;
```

```
            } else {
              findFirstLetter = true;
            }
          }
          if (lastPos == 15) findLastLetter = true; // Dont look for
last letter is an activeletter is at end of board.
          if (game.tiles[i].posX == lastPos && game.tiles[i].posY ==
yPos) {
            if (game.tiles[i].letter != null) {
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              lastPos++;
            } else {
              findLastLetter = true;
            }
          }
        }
      }
      activeLetters.sort(sortXpos); // Sort array to create the word

    } else if (vertical) {
      findFirstLetter = false;
      findLastLetter = false;
      while (!findFirstLetter || !findLastLetter) {
        for (var i = 0, len = game.tiles.length; i < len; i++) {
        if (firstPos == -1) findFirstLetter = true; // Dont look for
first letter is an activeletter is at end of board.
          if (game.tiles[i].posY == firstPos && game.tiles[i].posX
== xPos) {
            if (game.tiles[i].letter != null) {
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              firstPos--;
            } else {
              findFirstLetter = true;
            }
          }
          if (lastPos == 15) findLastLetter = true; // Dont look for
last letter is an activeletter is at end of board.
          if (game.tiles[i].posY == lastPos && game.tiles[i].posX ==
xPos) {
            if (game.tiles[i].letter != null) {
```

```
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              lastPos++;
            } else {
              findLastLetter = true;
            }
          }
        }
      }
      activeLetters.sort(sortYpos); // Sort array to create the word
    }

    // Create word from letters.
    for (i = 0; i < activeLetters.length; i++) {
      if (connected) {
        word += activeLetters[i].letter.letter;
        if (activeLetters[i].letter.active) {
          switch(activeLetters[i].property) {
            case "TL":
              points += activeLetters[i].letter.val*3;
            break;
            case "DL":
              points += activeLetters[i].letter.val*2;
            break;
            case "TW":
              points += activeLetters[i].letter.val;
              multiplier += 2;
            break;
            case "DW":
              points += activeLetters[i].letter.val;
              multiplier += 1;
            break;
            case "":
              points += activeLetters[i].letter.val;
            break;
          }
        } else {
          points += activeLetters[i].letter.val;
        }
      }
    }

    // If the word is connected and contains more than 1 letter,
move on.
    if (connected && activeLetters.length > 1) {
```

```
      returnData = new Object();          // Return data as an
object.
      returnData.word = word;           // Add played word to object.
      returnData.points = points*multiplier;   // Add points to
object.
      returnData.letters = activeLetters;   // Add active letters to
object.
      return returnData;
    } else {
      return false;
    }
  }

  function findWords() {
    var activeLetters = [],
       vertical = true,
       horizontal = true,
       words = [],
       points = 0,
       allLetters = [];

    // Find the activeLetters.
    for (var i = 0; i < game.tiles.length; i++) {
      if(game.tiles[i].letter!= null && game.tiles[i].letter.active)
activeLetters.push(game.tiles[i]);
    }

    // Check if the word is vertical or horizontal.
    for (var i = 0, len = activeLetters.length; i < len; i++) {
      if (i > 0 && i <= len) {
        if (activeLetters[i].posX != activeLetters[i-1].posX)
vertical = false;
        if (activeLetters[i].posY != activeLetters[i-1].posY)
horizontal = false;
      }
    }

    if ((horizontal || vertical) && activeLetters.length > 0) {
      var yPos = 0,
         xPos = 0,
         done = 0,
         add = true,
         connected = true,
         returnData,
         passiveCount = 0;

      if (horizontal) {
        for (var i = 0, len = activeLetters.length; i < len; i++) {
```

```
          // Gather all vertical words connected to the
activeletters.
          returnData = findWord(activeLetters[i].posY,
activeLetters[i].posY, activeLetters[i].posX, 0, true, false,
activeLetters[i]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
          yPos = activeLetters[i].posY;
        }
        // Get the primary played word.
        if (activeLetters.length > 1) {
          returnData = findWord(activeLetters[0].posX,
activeLetters[0].posX, xPos, yPos, vertical, horizontal,
activeLetters[0]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
        }
        // Check that all activeLetters are included in the primary
played word.
        if (!vertical && horizontal) {
          var alCount = 0;
          if (returnData) {
            for (i = 0; i < returnData.letters.length; i++) {
              if (returnData.letters[i].letter.active) alCount++;
              if (i > 0 && returnData.letters[i].posX !=
returnData.letters[i-1].posX+1) connected = false;
            }
            if (alCount != activeLetters.length) connected = false;
            } else {
              connected = false;
            }
        }
      }

      if (vertical) {
        if (horizontal) done = 0; // If only one letter is played,
the done-variable is reset.
        for (var i = 0, len = activeLetters.length; i < len; i++) {
```

```
          // Gather all horizontal words connected to the
activeletters.
          returnData = findWord(activeLetters[i].posX,
activeLetters[i].posX, 0, activeLetters[i].posY, false, true,
activeLetters[i]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
          xPos = activeLetters[i].posX;
        }

        // Get the primary played word.
        if (vertical && !horizontal) {
          returnData = findWord(activeLetters[0].posY,
activeLetters[0].posY, xPos, yPos, vertical, horizontal,
activeLetters[0]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
        }

        // Check that all activeLetters are included in the primary
played word.
        if (vertical && !horizontal) {
          var alCount = 0;
          if (returnData) {
            for (i = 0; i < returnData.letters.length; i++) {
              if (returnData.letters[i].letter.active) alCount++;
              if (i > 0 && returnData.letters[i].posY !=
returnData.letters[i-1].posY+1) connected = false;
            }
            if (alCount != activeLetters.length) connected = false;
          } else {
            connected = false;
          }
        }
      }
```

```
      // Confirms that the word is connected to a previously played
word or first played word.
      for (i = 0; i < allLetters.length; i++) {
        if (!allLetters[i].letter.active || (allLetters[i].posX == 7
&& allLetters[i].posY == 7)) passiveCount++;
      }

      if (passiveCount == 0) connected = false;
      if (connected) {
        returnData = new Object();
        returnData.words = words;
        returnData.points = points;
        returnData.activeLetters = activeLetters;
        return returnData;
      }
    }
  }

  // Give val letters to the player from the letterpool.
  function giveLetters(val) {
    $.post("AJAXhandler.php", { functionCall: "giveLetters", game:
player.gameId, player: player.playerName, letters: val},
    function(data) {
      if (typeof data == "string") {
        returnData = JSON.parse(data);
        for (i = 0; i < returnData.length; i++) {
          player.letters.push(returnData[i]);
        }
      }
    });
  }

  function updatePlayerLetters() {
    $.post("AJAXhandler.php", { functionCall: "updatePlayerLetters",
game: player.gameId, player: player.playerName, letters:
JSON.stringify(player.letters)},
    function(data) {
      //player.letters = JSON.parse(data);
    });
  }

  function gameBoard() {
    this.tiles = []; // 15x15

    // Fills board with empty tiles for letters.
    this.fillTiles = function() {
      for (var i=0; i<15; i++) {
        for (var j=0; j<15; j++) {
```

```
          this.tiles.push(new tile(i*slotWidth, j*slotHeight, i, j,
checkProperty(j,i)));
        }
      }
    }

    // Draws board on canvas.
    this.draw = function() {
      // Draw player-frame.
      context.fillStyle = "4A4A4A";
      context.strokeStyle = "000";
      context.lineWidth = 1;
      context.fillRect(0, slotHeight*15, playerframeWidth,
slotHeight);

      // Draw swap-frame.
      context.strokeStyle = "000";
      context.fillStyle = "FFFDC4";
      context.roundRect(playerframeWidth-letterWidth*2,
slotHeight*15+1, letterWidth*2, letterHeight, 5).fill();
      context.roundRect(playerframeWidth-letterWidth*2,
slotHeight*15+1, letterWidth*2, letterHeight, 5).stroke();
      context.fillStyle = "4A4A4A";
      context.font = swapFont;
      context.fillText("SWAP",(playerframeWidth-
letterWidth*2+fontMod/2), letterHeight*16+fontModY/2);


      // Draw player-letters.
      for (var i=0; i < player.letters.length; i++) {
        drawLetter(player.letters[i], 1+i*letterWidth,
slotHeight*15, player.letters[i].active);
        player.letters[i].realX = 1+i*letterWidth;
        player.letters[i].realY = slotHeight*15;
      }

      // Draw player-letters to be swapped.
      for (var i=0; i < player.swapLetters.length; i++) {
        drawLetter(player.swapLetters[i],
(player.letters.length*letterWidth)+i*letterWidth, slotHeight*15,
false);
        player.swapLetters[i].realX =
(player.letters.length*letterWidth)+i*letterWidth;
        player.swapLetters[i].realY = slotHeight*15;
      }

      // Draw board-tiles.
      for (var i=0, len=15; i<len; i++) {
```

```
        for (var j=0; j<len; j++) {
          var color = "4A4A4A";
          context.fillStyle = color;
          context.fillRect(j*slotWidth, i*slotHeight, letterWidth,
letterHeight);
          for (var k=0; k < game.tiles.length; k++) {
            if (game.tiles[k].posX == i && game.tiles[k].posY == j)
{

              // Set backgroundcolor for each bonus-tile.
              switch(game.tiles[k].property) {
                case "DW":
                  color = "EB961E";
                  break;
                case "TW":
                  color = "D91E1E";
                  break;
                case "DL":
                  color = "63E080";
                  break;
                case "TL":
                  color = "49B9E6";
                  break;
                case "":
                  color = "4A4A4A";
              }

              /* Align to center depending on bonus-tile */
              (game.tiles[k].property == "TW" ||
game.tiles[k].property == "DW") ? modifier = 2 : modifier = +4;

              context.fillStyle = color;
              context.fillRect(j*slotWidth, i*slotHeight,
letterWidth, letterHeight);
              context.fillStyle = "FFF";
              context.font=tileFont;
              context.fillText(game.tiles[k].property,
(modifier+j*slotWidth), fontModY+i*slotHeight);
            }
          }
        }
      }

      // Draw letters on tiles.
      for (i = 0; i < game.tiles.length; i++) {
        if (game.tiles[i].letter != null) {
          drawLetter(game.tiles[i].letter, game.tiles[i].realX,
game.tiles[i].realY, game.tiles[i].letter.active);
```

```
        }
      }
    }
  }

  function checkDropTile() {
    for (var i = 0; i < game.tiles.length; i++) {
      if ((moveObject) && (mouseX >= Math.floor(game.tiles[i].realX)
&& mouseX <= Math.ceil(game.tiles[i].realX+letterWidth)) && (mouseY-
5 >= Math.floor(game.tiles[i].realY) && mouseY-5 <=
Math.ceil(game.tiles[i].realY+letterHeight))) {
        if (game.tiles[i].letter != null &&
game.tiles[i].letter.active) {
          player.letters.push(game.tiles[i].letter); // If theres is
an active letter on the tile, don't loose it.
          game.tiles[i].letter = moveObject;

        } else if (game.tiles[i].letter != null &&
!game.tiles[i].letter.active) {
          player.letters.push(moveObject);

        } else {
          game.tiles[i].letter = moveObject; // Move object to tile.
          game.tiles[i].letter.realX = game.tiles[i].realX; //
Change x-position.
          game.tiles[i].letter.realY = game.tiles[i].realY; //
Change y-position.
        }

      } else if (moveObject && mouseY > Math.floor(slotHeight*15) &&
mouseX <= Math.ceil(playerframeWidth-letterWidth*2)) { // Letter
dropped in player-frame.
        player.letters.push(moveObject);
        moveObject = null;
      } else if (moveObject && mouseY > Math.floor(slotHeight*15) &&
mouseX > Math.ceil(playerframeWidth-letterWidth*2)) { // Letter is
to be swapped!
        player.swapLetters.push(moveObject);
        moveObject = null;
      }
    }
  }

  function mouseMove(e) {
    if (dragok){
      if (e.clientX && e.clientY) {   //Handle mouse-events
        mouseX = e.clientX;
        mouseY = e.clientY;
```

```
      } else {              // Handle touch-events
        mouseX = e.targetTouches[0].pageX;
        mouseY = e.targetTouches[0].pageY;
      }
    }
  }

  function mouseDown(e) {
    dragok = true;
    if (e.clientX && e.clientY) {   //Handle mouse-events
      mouseX = e.clientX;
      mouseY = e.clientY;
    } else {              // Handle touch-events
      mouseX = e.targetTouches[0].pageX;
      mouseY = e.targetTouches[0].pageY;
    }
    canvas.addEventListener('mousemove', mouseMove, false); //Handle
mouse-events
    canvas.addEventListener('touchmove', mouseMove, false); //
Handle touch-events

    // Check if player is dragging letters from panel.
    for (var i = 0; i < player.letters.length; i++) {
      if((mouseX-5 > player.letters[i].realX && mouseX-5 <
player.letters[i].realX+letterWidth) && (mouseY-5 >
player.letters[i].realY && mouseY-5 <
player.letters[i].realY+letterHeight)) {
        moveObject = player.letters[i];
        player.letters.splice(i,1);
      }
    }

    for (var i = 0; i < player.swapLetters.length; i++) {
      if((mouseX-5 > player.swapLetters[i].realX && mouseX-5 <
player.swapLetters[i].realX+letterWidth) && (mouseY-5 >
player.swapLetters[i].realY && mouseY-5 <
player.swapLetters[i].realY+letterHeight)) {
        moveObject = player.swapLetters[i];
        player.swapLetters.splice(i,1);
      }
    }

    // Check if player is dragging letters from gameboard.
    for (i = 0; i < game.tiles.length; i++) {
      if (game.tiles[i].letter) {
        if((mouseX-5 > game.tiles[i].realX && mouseX-5 <
game.tiles[i].realX+letterWidth) && (mouseY-5 > game.tiles[i].realY
```

```
&& mouseY-5 < game.tiles[i].realY+letterHeight) &&
(game.tiles[i].letter.active)) {
          moveObject = game.tiles[i].letter;
          game.tiles[i].letter = null;
        }
      }
    }
  }

  function mouseUp() {
    checkDropTile();
    dragok = false;
    moveObject = null;

    canvas.onmousemove = null;
  }

  function clear() {
     context.clearRect(0, 0, 320*sizeModifier, 373*sizeModifier);
  }

  function drawLetter(letter, x, y, active) {
    (active) ? context.fillStyle = "FFF" : context.fillStyle =
"FFFDC4";

    context.strokeStyle = "000";
    context.lineWidth   = 1;
    context.roundRect(x, y, letterWidth, letterHeight, 5).fill();
    context.roundRect(x, y, letterWidth, letterHeight, 5).stroke();

    context.fillStyle = "4A4A4A";
    context.font = letterFont;
    context.fillText(letter.letter, x+fontMod, y+fontModY, 20);

    context.font = letterValFont;
    (letter.val >= 10) ? modifier = 1*sizeModifier : modifier =
4*sizeModifier;
    context.fillText(letter.val, x+modifier+(fontMod*2),
y+fontModY/2+1, 20);
  }

  function draw() {
     clear();
     game.draw();
     if (dragok && moveObject) {
       drawLetter(moveObject, mouseX-letterWidth/2, mouseY-
letterHeight/2, moveObject.active);
     }
```

```
  }

  function init() {
    game.fillTiles();
    getPlayerLetters();
    //checkTurnLongpolling();
    checkTurn();
    curInterval = setInterval(checkTurn, intervalTime);

    // Add events for canvas.
    canvas.addEventListener("mousedown", mouseDown, false);
    canvas.addEventListener("mouseup", mouseUp, false);
    canvas.addEventListener("touchstart", mouseDown, false);
    canvas.addEventListener("touchend", mouseUp, false);

    // Add events for buttons.
    document.getElementById('swap').addEventListener("click", swap,
false);
    document.getElementById('play').addEventListener("click", play,
false);
    document.getElementById('pass').addEventListener("click", pass,
false);
    document.getElementById('menu').addEventListener("click", menu,
false);
  }

  function swap() {
    if (player.myTurn && player.swapLetters.length > 0) {
      $.post("AJAXhandler.php", { functionCall: "swapLetters",
letters: JSON.stringify(player.swapLetters), player:
player.playerName, game: player.gameId},
      function(data) {
        if (data && data.charAt(0) != " ") {
          returnData = JSON.parse(data);
          if(returnData[0].letter) {
            // Push all new letters
            for (i = 0; i < returnData.length; i++) {
              player.letters.push(returnData[i])
            }

            player.swapLetters = []; // Empty player swapLetters.
            player.MyTurn = false;
            setTimeout(checkTurn, 1000);
            curInterval = setInterval(checkTurn, intervalTime);
          }
        } else {
          $("#msg").empty();
          $("#msg").append(data);
```

```
        }

      });
    }
  }

  function menu() {
    game.tiles = [];      // Clear gameboard from previously viewed
game.
    player.letters = [];     // Clear letters from previously viewed
game.
    player.swapLetters = [];  // Clear swapLetters from previously
viewed game.
    player.turnCount = 0;    // Reset turnCount.
    //request.abort();      // Abort current AJAX-call.
    clearInterval(curInterval); // Abort intervals.
    $("#gameList").show();     // Show gamelist
    $("#app").hide();       // Hide canvas
    $("#wrapper").hide();     // Hide canvasrelated elements
  }

  function play() {
    data = findWords();
    if (player.myTurn) {
      $.post("AJAXhandler.php", { functionCall: "checkWords", words:
JSON.stringify(data), player: player.playerName, game:
player.gameId},
      function(data) {
        if (data && typeof data == "string" && data.charAt(0) != "
") {
          returnData = JSON.parse(data);

          // Deactivate all current letters on playBoard
          for (i = 0; i < game.tiles.length; i++) {
            if (game.tiles[i].letter && game.tiles[i].letter.active)
{
              game.tiles[i].letter.active = false;
            }
          }
          if(returnData.letters != 0) {
            // Push all new letters
            for (i = 0; i < returnData.letters.length; i++) {
              player.letters.push(returnData.letters[i])
            }

            // Set turn to false.
            player.myTurn = false;
            setTimeout(checkTurn, 1000);
```

```
            curInterval = setInterval(checkTurn, intervalTime);
          } else if (returnData.letters == 0){
            player.myTurn = false;
            setTimeout(checkTurn, 1000);
            curInterval = setInterval(checkTurn, intervalTime);
          }
        } else {
          $("#msg").empty();
          $("#msg").append(data+" is no word!");
        }
      });
    }
  }

  function changeButtons() {
    if (player.myTurn) {
      document.getElementById('swap').disabled = false;
      document.getElementById('play').disabled = false;
      document.getElementById('pass').disabled = false;
    } else {
      document.getElementById('swap').disabled = true;
      document.getElementById('play').disabled = true;
      document.getElementById('pass').disabled = true;
    }
    $("#msg").empty();
  }

  function pass() {
    $.post("AJAXhandler.php", { functionCall: "pass", game:
player.gameId},
    function(data) {
      player.myTurn = false;
      changeButtons();
    });
    curInterval = setInterval(checkTurn, intervalTime);
    setTimeout(checkTurn, 1000);

  }

  function login() {
    $.post("AJAXhandler.php", { functionCall: "login", player:
$("#playername").val()},
      function(data) {
        if(typeof data == "string") {
          returnData = JSON.parse(data);
          player.playerName = $("#playername").val();
          $("#loginForm").hide();
```

```
          for (i = 0; i < returnData.length; i++) {
            if (returnData[i].over == 1) {
              $("#gameList").append("<div style='background-color:
#63E080;' class='game' id='"+returnData[i].game+"'>GameId:
"+returnData[i].game+"</div>");
            } else {
              $("#gameList").append("<div class='game'
id='"+returnData[i].game+"'>GameId: "+returnData[i].game+"</div>");
            }
          }

          $(".game").click(function() {
            showGame(this.id);
          });
          $(".game").css("width", playerframeWidth);
          $(".game").css("height", 60*sizeModifier);
          $(".game").css("font-size", 48*sizeModifier);
          //$(".game").css("padding-top", (60*sizeModifier)/2);
        }
      });
  }

  function showGame(game) {
    player.gameId = game;
    if(player.playerName && player.gameId) {
      $("#gameList").hide();
      $("#app").show();
      $("#wrapper").show();
      init();
    }
  }

  function checkTurn() {
    $.post("AJAXhandler.php", { functionCall: "checkTurn", player:
player.playerName, game: player.gameId, turnCount:
player.turnCount},
    function(data) {
      if(typeof data == "string" && data != "") {
        returnData = JSON.parse(data);
        //Update score, turn and poolsize.
        $("#score").empty();
        $("#score").append("Turn: <span
style='color:#63E080;'>"+returnData.turnCount+"</span>");
        $("#score").append(" Letters left: <span
style='color:#63E080;'>"+returnData.lettersLeft+"</span><br />");
        $("#score").append(returnData.score);

        //Update gameBoard.
```

```
        for (i = 0; i < returnData.gameBoard.length; i++) {
          if (returnData.gameBoard[i].letter) {
            if (game.tiles[i].letter && game.tiles[i].letter.active)
{
              player.letters.push(game.tiles[i].letter);
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            } else {
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            }
          }
        }
        player.turnCount = returnData.turnCount;
        if (returnData.gameOver) {
          player.myTurn = false;
          clearInterval(curInterval);
          changeButtons();
        } else if (returnData.myTurn) {
          player.myTurn = true;
          clearInterval(curInterval);
        } else {
          player.myTurn = false;
        }
      } else {
        player.myTurn = false;
      }
      changeButtons();
    });
  }

  function checkTurnLongpolling() {
    request = $.post("AJAXhandler.php", { functionCall:
"checkTurnLongpolling", player: player.playerName, game:
player.gameId, turnCount: player.turnCount},
    function(data) {
      if(typeof data == "string" && data != "" && data.charAt(0) !=
"<") {
        returnData = JSON.parse(data);
        //Update score, turn and poolsize.
        $("#score").empty();
        $("#score").append("Turn: <span
style='color:#63E080;'>"+returnData.turnCount+"</span>");
        $("#score").append(" Letters left: <span
style='color:#63E080;'>"+returnData.lettersLeft+"</span><br />");
        $("#score").append(returnData.score);

        //Update gameBoard.
        for (i = 0; i < returnData.gameBoard.length; i++) {
          if (returnData.gameBoard[i].letter != "") {
```

```
            if (game.tiles[i].letter && game.tiles[i].letter.active)
{
              player.letters.push(game.tiles[i].letter);
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            } else {
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            }
          }
        }
        player.turnCount = returnData.turnCount;
        if (returnData.gameOver) {
          player.myTurn = false;
          //clearInterval(curInterval);
          changeButtons();
        } else if (returnData.myTurn) {
          player.myTurn = true;
          //clearInterval(curInterval);
        } else {
          player.myTurn = false;
          checkTurnLongpolling();
        }
      } else {
        player.myTurn = false;
        checkTurnLongpolling();
      }
      changeButtons();
    });
  }
};
```

Serverscript.php

```
<?php
$game = new game();
class letter {
  public  $letter = "",
      $realX = 0,
      $val = 0,
      $realY = 0,
      $active = true;

  function __construct($letter, $val) {
    $this->letter = $letter;
    $this->val = $val;
    return 0;
  }
}

class game {
  public $letterPool = array();

  public function getLetterPool($gameId) {
    $pdo=getPDO();
    $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

    $query="SELECT letterPool FROM games WHERE id = :GAMEID;";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
    $result=$stmt->fetchColumn(0);
    if ($result) {
      $letters = json_decode($result);
      $this->letterPool = array();
      foreach ($letters as $letter) {
        array_push($this->letterPool, $letter);
      }
    }
  }

  public function giveLetters($gameId, $playerName, $val) {
    $this->getLetterPool($gameId);
    if (count($this->letterPool) > 0) {
      $min = 0;
      $max = count($this->letterPool)-1;
      $returnData = array();

      // Get playerletters from database.
      $pdo=getPDO();
      $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );
```

```
      $query="SELECT * FROM playgames WHERE playerName = :USERNAME
AND gameId = :GAMEID;";
      $stmt = $pdo->prepare($query);
      $stmt->bindParam(':USERNAME', $playerName);
      $stmt->bindParam(':GAMEID', $gameId);
      $stmt->execute();
      $result=$stmt->fetchColumn(2);
      $playerLetters = json_decode($result);
      if (!$playerLetters) $playerLetters = array(); // If the
player doesn't have any letters, create an array instead.

      for ($i = 0; $i < $val; $i++) {
        $index = rand($min, $max);

        /* Checks if the letter still is exists in pool */
        if(isset($this->letterPool[$index])) {
          array_push($returnData, $this->letterPool[$index]);
          array_push($playerLetters, $this->letterPool[$index]);
          unset($this->letterPool[$index]);
        } else {
          $i--;
        }
      }

      // Update playerletters in database
      $query="UPDATE playgames SET playerLetters = :LETTERS WHERE
playerName = :USERNAME AND gameId = :GAMEID;";
      $stmt = $pdo->prepare($query);
      $stmt->bindParam(':USERNAME', $playerName);
      $stmt->bindParam(':GAMEID', $gameId);
      $stmt->bindParam(':LETTERS', json_encode($playerLetters));
      $stmt->execute();

      // Update letterpool
      $query="UPDATE games SET letterPool = :LETTERPOOL WHERE id =
:GAMEID;";
      $stmt = $pdo->prepare($query);
      $stmt->bindParam(':GAMEID', $gameId);
      $stmt->bindParam(':LETTERPOOL', json_encode($this-
>letterPool));
      $stmt->execute();
    } else {
      $returnData = "empty";
    }

    return $returnData;
  }
```

```
  public function removeLetters($gameId, $playerName, $letters) {
    $pdo=getPDO();
    $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

    $newPlayerLetters = array();
    $lettersToRemove = array();
    $playerLetters = json_decode(getPlayerLetters($gameId,
$playerName));

    // Get letters to be removed.
    foreach ($letters as $letter) {
      array_push($lettersToRemove, $letter->letter->letter);
    }

    // Remove letters to playerLetters.
    foreach ($lettersToRemove as $letterToRemove) {
      $index = 0;
      foreach ($playerLetters as $key => $letter) {
        if($letter->letter == $letterToRemove) {
          unset($playerLetters[$key]);
          break;
        }
        $index++;
      }
    }

    // Push letter to new array for correct format
    foreach ($playerLetters as $letter) {
      array_push($newPlayerLetters, $letter);
    }

    // Push new letters to database.
    $query="UPDATE playgames SET playerLetters = :LETTERS WHERE
playerName = :USERNAME AND gameId = :GAMEID;";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':LETTERS', json_encode($newPlayerLetters));
    $stmt->bindParam(':USERNAME', $playerName);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
  }

  public function addLetters($gameId, $letters) {
    $pdo=getPDO();
    $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

    $query="SELECT letterPool FROM games WHERE id = :GAMEID;";
    $stmt = $pdo->prepare($query);
```

```
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
    $result=$stmt->fetchAll();
    $letterPool = json_decode($result[0]["letterPool"]);
    $newLetterPool = array();

    foreach ($letterPool as $letter) {
      array_push($newLetterPool, $letter);
    }

    foreach ($letters as $letter) {
      array_push($newLetterPool, $letter);
    }

    $query="UPDATE games SET letterPool = :LETTERPOOL WHERE id =
:GAMEID;";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':LETTERPOOL', json_encode($newLetterPool));
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
  }
}

/* Returns db connection */
function getPDO() {
  $pdo = new PDO('mysql:dbname=scrabble;host=localhost', 'scrabble',
'xqQNTBsvxdtf7XQ6');
  return $pdo;
}

/* Returns TRUE if successful */
function login($username) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  $query="SELECT * FROM players WHERE name = :USERNAME;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':USERNAME', $username);
  $stmt->execute();
  $result=$stmt->fetchColumn(0);

  if ($result == $username) {
    $query="UPDATE players SET status = 1 WHERE name = :USERNAME";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':USERNAME', $username);
    $stmt->execute();
    return TRUE;
  } else {
```

```
    return FALSE;
  }
}

function getPlayerGames($player) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  $query="SELECT * FROM playGames INNER JOIN games on gameId=id
WHERE playerName = :USERNAME;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':USERNAME', $player);
  $stmt->execute();
  $result=$stmt->fetchAll();
  $returnData = array();

  for ($i = 0; $i < count($result); $i++) {
    $object = new stdClass;
    $object->game = $result[$i]["gameId"];
    $object->over = $result[$i]["gameOver"];
    array_push($returnData,$object);
  }

  return json_encode($returnData);
}

function logout($username) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  $query="UPDATE players SET status = 0 WHERE name = :USERNAME";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':USERNAME', $username);
  if($stmt->execute()) {
    return TRUE;
  } else {
    return FALSE;
  }
}

function checkWords($words, $playerName, $gameId) {
  $json = json_decode($words);
  $noWord = "";
  $points = 0;
  $pdo=getPDO();
  $pdo->exec('SET NAMES UTF8');
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );
```

```
  /* Check all words with dictionary */
  foreach ($json->words as $word) {
    $utf8proofWord = mb_strtolower($word,
mb_detect_encoding($word));
    $query="SELECT * FROM se_dictionary WHERE word = :WORD";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':WORD', $utf8proofWord);
    $stmt->execute();
    $row = $stmt->fetch();
    if(count($row) <= 1) {
      $noWord = $noWord." ".$utf8proofWord;
    }
  }

  if ($noWord) { // Return the words that are declined.
    return $noWord;
  } else {
    // Add points to the player.
    $query="UPDATE playgames SET score = score+:POINTS WHERE
playerName = :PLAYERNAME AND gameId = :GAMEID";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':PLAYERNAME', $playerName);
    $stmt->bindParam(':POINTS', $json->points);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();

    // Get gameBoard from database.
    $query="SELECT * FROM games WHERE id = :GAMEID";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
    $row = $stmt->fetchColumn(6);
    $gameBoard = json_decode($row);

    // Update gameBoard with new tiles.
    foreach ($json->activeLetters as $letter) {
      $letter->letter->active = 0; // Set letter to inactive before
push.
      for ($i = 0; $i < count($gameBoard); $i++) {
        if ($gameBoard[$i]->posX == $letter->posX && $gameBoard[$i]-
>posY == $letter->posY) {
          $gameBoard[$i] = $letter;
        }
      }
    }

    // Push new gameBoard to database.
```

```
    $query="UPDATE games SET gameBoard = :GAMEBOARD, passCount = 0
WHERE id = :GAMEID";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':GAMEBOARD', json_encode($gameBoard));
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();

    // Return 2 for success.
    return 2;
  }
}

function getTiles($gameId) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get gameBoard from database.
  $query="SELECT * FROM games WHERE id = :GAMEID";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $row = $stmt->fetchColumn(6);
  return json_decode($row);
}

function getPlayerLetters($gameId, $playerName) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get gameBoard from database.
  $query="SELECT playerLetters, started FROM playgames WHERE
playerName = :USERNAME AND gameId = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':USERNAME', $playerName);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();

  // Set starttime for game.
  if ($result[0][1] == "0000-00-00 00:00:00") {
    $query="UPDATE playgames SET started = NOW() WHERE playerName =
:USERNAME AND gameId = :GAMEID;";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':USERNAME', $playerName);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->execute();
  }
```

```
  return $result[0][0];
}

function checkTurn($gameId, $playerName) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get player turnId and the current turn from game.
  $query="SELECT turn,turnId FROM playgames INNER JOIN games on
gameId=id WHERE playerName = :USERNAME AND gameId = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':USERNAME', $playerName);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();

  // If the turns match, it is the players turn.
  if($result[0]["turnId"] == $result[0]["turn"]) {
    return true;
  } else {
    return false;
  }
}

function changeTurn($gameId, $playerName) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get current turn and playerCount.
  $query="SELECT turn,playercount FROM games WHERE id = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();

  // If it is the last players turn, the next turn will go to the
first player.
  if($result[0]["turn"] == $result[0]["playercount"]) {
    $turn = 1;
  } else {
    $turn = $result[0]["turn"]+1;
  }

  // Update database.
  $query="UPDATE games SET turn = :TURN, turnCount = turnCount + 1
WHERE id = :GAMEID";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
```

```
  $stmt->bindParam(':TURN', $turn);
  $stmt->execute();
}

function getScore($gameId) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  $query="SELECT playerName,score,turnId,turn FROM playgames INNER
JOIN games on gameId=id WHERE gameId = :GAMEID ORDER BY score
desc;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();
  $returnString = "";
  $printCount = 0;

  foreach ($result as $player) {
    if ($player["turnId"] == $player["turn"]) {
      $returnString = $returnString."<span
class='current'>".$player["playerName"]." - ".$player["score"]."p
</span>";
    } else {
      $returnString = $returnString."<span>".$player["playerName"]."
- ".$player["score"]."p </span >";
    }
    ($printCount % 2) ? $returnString = $returnString."<br />" :
false;
    $printCount++;
  }

  return $returnString;
}

function getTurnCount($gameId) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get current turnCount.
  $query="SELECT turnCount FROM games WHERE id = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();

  return $result[0]["turnCount"];
}
```

```
function pass($gameId) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  $query="UPDATE games SET passCount = passCount+1 WHERE id =
:GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();

  $query="SELECT passCount FROM games WHERE id = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();

  return $result[0]["passCount"];
}

function checkGameOver($gameId) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get current turnCount.
  $query="SELECT gameOver FROM games WHERE id = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();

  ($result[0]["gameOver"] == 1) ? $returndata = true : $returndata =
false;
  return $returndata;
}

function endGame($gameId) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Get current turnCount.
  $query="SELECT playerName,playerLetters,score FROM playgames WHERE
gameId = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
  $result=$stmt->fetchAll();
  $players = array();
```

```
  foreach ($result as $player) {
    $newPlayerData = new stdClass;
    $newPlayerData->playerName = $player["playerName"];
    $newPlayerData->score = $player["score"];
    $letters = json_decode($player["playerLetters"]);

    // Remove points for remaining letters.
    foreach ($letters as $letter) {
      $newPlayerData->score -= $letter->val;
    }

    array_push($players, $newPlayerData);
  }

  // Push new data to database.
  foreach ($players as $player) {
    $query="UPDATE playgames SET score = :SCORE, ended = NOW() WHERE
gameId = :GAMEID AND playerName = :PLAYERNAME;";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':GAMEID', $gameId);
    $stmt->bindParam(':SCORE', $player->score);
    $stmt->bindParam(':PLAYERNAME', $player->playerName);
    $stmt->execute();
  }

  // End game by setting gameOver to 1.
  $query="UPDATE games SET gameOver = 1 WHERE id = :GAMEID;";
  $stmt = $pdo->prepare($query);
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->execute();
}

function dataUsage($gameId, $playerName, $dataSize, $direction) {
  $pdo=getPDO();
  $pdo->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );

  // Update column based on sent or received amount of data.
  if ($direction == "sent") {
    $query="UPDATE playgames SET sent = sent+:DATASIZE, callCount =
callCount+1 WHERE gameId = :GAMEID AND playerName = :PLAYERNAME;";
  } else {
    $query="UPDATE playgames SET received = received+:DATASIZE,
callCount = callCount+1 WHERE gameId = :GAMEID AND playerName =
:PLAYERNAME;";
  }

  $stmt = $pdo->prepare($query);
```

```
  $stmt->bindParam(':GAMEID', $gameId);
  $stmt->bindParam(':PLAYERNAME', $playerName);
  $stmt->bindParam(':DATASIZE', $dataSize);
  $stmt->execute();
}
?>
```

AJAXhandler.php

```
<?php
$intervalTime = 10;
include("serverScript.php");

switch($_POST['functionCall']) {

  // Check if the played word exists or not.
  case "checkWords":
    if (isset($_POST['words']) && isset($_POST['player']) &&
isset($_POST['game'])) {

      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['words'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      if (checkWords($_POST['words'], $_POST['player'],
$_POST['game']) == 2) {
        // Return new letters to the player
        $game->getLetterPool($_POST['game']);              // Get
letterpool for game.
        $activeLetters = json_decode($_POST['words'])-
>activeLetters;  // Get played letters.
        $letterPool_size = count($game->letterPool);          // Get
size of letterpool.
        $activeLetters_size = 0;                    // Create
variable for size of letters.
        $returnData = new stdClass;                    // Create
object to return with JSON.
        $returnData->letters = 0;                    // Create
object-variable letters.
        $returnData->score = getScore($_POST['game']);          //
Create object-variable score.

        // Cant use count sinze it contains both array and objects.
        foreach ($activeLetters as $letter) {
          $activeLetters_size++;
        }

        // There are letters to return.
        if ($letterPool_size > 0 && $letterPool_size >=
$activeLetters_size) {
          $returnData->letters = $game->giveLetters($_POST['game'],
$_POST['player'], $activeLetters_size);
```

```
        // There are not enough letters to return.
        } else if ($letterPool_size > 0 && $letterPool_size <
$activeLetters_size) {
          $returnData->letters = $game->giveLetters($_POST['game'],
$_POST['player'], $letterPool_size);
        }
        $returnData = json_encode($returnData);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData; // Return object to client.
        $game->removeLetters($_POST['game'], $_POST['player'],
$activeLetters); // Remove letters from player.
        changeTurn($_POST['game'], $_POST['player']); // Change
turn.

        // Check if the game is over.
        (count(json_decode(getPlayerLetters($_POST['game'],
$_POST['player']))) == 0) ? endgame($_POST['game']) : false;
      } else {
        $returnData = checkWords($_POST['words'], $_POST['player'],
$_POST['game']);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;
      }
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Swap letters for player.
  case "swapLetters":
    if (isset($_POST['letters']) && isset($_POST['player']) &&
isset($_POST['game'])) {
      $game->getLetterPool($_POST['game']);

      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['letters'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      // Dont allow player to swap letters if there are 10 or less
letters left in the pool.
      if (count($game->letterPool) >= 10) {
        $letters = json_decode($_POST['letters']);
```

```
        // Push letters to correct format for removeLetters-
function.
        $lettersToRemove = array();
        foreach ($letters as $letter) {
          $object = new stdClass;
          $object->letter = "";
          $object->letter->letter = $letter->letter;
          array_push($lettersToRemove, $object);
        }

        $game->removeLetters($_POST['game'], $_POST['player'],
$lettersToRemove);
        $newLetters = $game->giveLetters($_POST['game'],
$_POST['player'], count($letters));
        $game->addLetters($_POST['game'], $letters);
        $returnData = json_encode($newLetters);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;

        changeTurn($_POST['game'], $_POST['player']);
      } else {
        echo " Swapping is no longer allowed!";
      }
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Give letters to a player in a game
  case "giveLetters":
    if (isset($_POST['game']) && isset($_POST['player']) &&
isset($_POST['letters'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['letters'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      $returnData = json_encode($game->giveLetters($_POST['game'],
$_POST['player'], $_POST['letters']));
      echo $returnData;

      $dataSize += mb_strlen($returnData, 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
```

```
  break;

  // Get players current letters from a game.
  case "getPlayerLetters":
    if (isset($_POST['game']) && isset($_POST['player'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      $returnData = getPlayerLetters($_POST['game'],
$_POST['player']);
      echo $returnData;

      $dataSize += mb_strlen($returnData, 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Login player.
  case "login":
    if (isset($_POST['player'])) {
      if(login($_POST['player'])) {
        echo getPlayerGames($_POST['player']);
      } else {
        echo "";
      }
    }
  break;

  // Check whos turn it is.
  case "checkTurn":
    if (isset($_POST['game']) && isset($_POST['player']) &&
isset($_POST['turnCount'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['turnCount'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      if (checkTurn($_POST['game'], $_POST['player'])) {
        $game->getLetterPool($_POST['game']);
        $returnData = new stdClass;
```

```
        $returnData->gameBoard = getTiles($_POST['game']);
        $returnData->score = getScore($_POST['game']);
        $returnData->myTurn = true;
        $returnData->turnCount = getTurnCount($_POST['game']);
        $returnData->lettersLeft = count($game->letterPool);
        $returnData->gameOver = checkGameOver($_POST['game']);
        $returnData = json_encode($returnData);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;
      } else if (getTurnCount($_POST['game']) !=
$_POST['turnCount']) {
        $game->getLetterPool($_POST['game']);
        $returnData = new stdClass;
        $returnData->gameBoard = getTiles($_POST['game']);
        $returnData->score = getScore($_POST['game']);
        $returnData->myTurn = false;
        $returnData->turnCount = getTurnCount($_POST['game']);
        $returnData->lettersLeft = count($game->letterPool);
        $returnData->gameOver = checkGameOver($_POST['game']);
        $returnData = json_encode($returnData);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;
      }
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Alternative method to check turn with long polling.
  case "checkTurnLongpolling":
    if (isset($_POST['game']) && isset($_POST['player']) &&
isset($_POST['turnCount'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['turnCount'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.
      $turn = false;

      while (!$turn) {
        if (checkTurn($_POST['game'], $_POST['player'])) {
          $game->getLetterPool($_POST['game']);
          $returnData = new stdClass;
          $returnData->gameBoard = getTiles($_POST['game']);
          $returnData->score = getScore($_POST['game']);
          $returnData->myTurn = true;
```

```
          $returnData->turnCount = getTurnCount($_POST['game']);
          $returnData->lettersLeft = count($game->letterPool);
          $returnData->gameOver = checkGameOver($_POST['game']);
          $returnData = json_encode($returnData);
          $dataSize += mb_strlen($returnData, 'latin1');

          dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
          echo $returnData;
          $turn = true;
        } else if (getTurnCount($_POST['game']) !=
$_POST['turnCount']) {
          $game->getLetterPool($_POST['game']);
          $returnData = new stdClass;
          $returnData->gameBoard = getTiles($_POST['game']);
          $returnData->score = getScore($_POST['game']);
          $returnData->myTurn = false;
          $returnData->turnCount = getTurnCount($_POST['game']);
          $returnData->lettersLeft = count($game->letterPool);
          $returnData->gameOver = checkGameOver($_POST['game']);
          $returnData = json_encode($returnData);
          $dataSize += mb_strlen($returnData, 'latin1');

          dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
          echo $returnData;
          $turn = true;
        } else {
          sleep($intervalTime);
        }
      }
    }
  break;

  case "pass":
    if (isset($_POST['game'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['game'], 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      if (pass($_POST['game']) >= 4) {
        endGame($_POST['game']);
      } else {
        changeTurn($_POST['game'], $_POST['player']);
      }
    }
  break;
}
```

```
?>
```

Style.css

```
body {
  margin: 0;
  padding: 0;
}
.big {
  width: 320px;
  min-height: 40px;
  font-family: arial, verdana, sans-serif;
  font-size: 48px;
}
#gameList {
  width: 320px;
  background-color: #4A4A4A;
  font-family: arial, verdana, sans-serif;
  font-size: 48px;
  color: #FFF;
}
#wrapper {
  background-color: #4A4A4A;
  width: 320px;
  height: 45px;
  margin-top: 1px;
}
.game {
  width: 320px;
  height: 60px;
  border-top: 1px solid;
  text-align: center;
}
#score {
  float: left;
  padding-left: 5px;
  min-width: 140px;
  font-size: 11px;
  color: #FFF;
  font-family: arial, verdana, sans-serif;
}
#msg {
  float: right;
  min-width: 50px;
  font-size: 11px;
  font-family: arial, verdana, sans-serif;
  color: #FFF;
}
.button {
  font-size: 13px;
  float: right;
```

```
  margin: 0px;
  padding: 0px;
  font-family: arial, verdana, sans-serif;
  color: #363636;
}
.current {
  color: #49B9E6;
  padding: 0;
  margin: 0;
}
```

Appendix C - Avvaktande AJAX-anrop

Index.html

```
<html>
<head>
  <title>SpeedScrabble</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js
"></script>
  <script type="text/javascript" src="script.js"></script>
</head>
<body>
<div id="loginForm">
  <input type="text" id="playername" class="big" /><button
id="login" class="big">login</button>
</div>
<div id="gameList"></div>
<canvas id="app" width="320" height="373" tabindex="1">You no see
canvas?</canvas>
<div id="wrapper">
  <div id="score"></div>
  <div>
    <button id="menu" class="button">Menu</button>
    <button id="pass" class="button">Pass</button>
    <button id="swap" class="button">Swap</button>
    <button id="play" class="button">Play</button>
  </div>
  <div id="msg"></div>
</div>
</body>
</html>
```

Script.js

```
window.onload = function(){
  game = new gameBoard();
  player = new playerClass("",500,"");
  (typeof window.innerWidth == "number" && window.innerWidth > 500)
? sizeModifier = (window.innerHeight)/430: sizeModifier = 1;
  document.getElementById('login').addEventListener("click", login,
false);
  document.body.addEventListener('touchmove',function(event) { //
prevent elastic scrolling
    event.preventDefault();
  },false); // end body:touchmove

  setInterval(draw, 30);
  $("#app").hide();
  $("#wrapper").hide();
  var dragok,                      // Enables letterdragging.
    canvas = document.getElementById('app'),    // Canvas element.
    context = canvas.getContext('2d'),        // Canvas context.
    request,                    // AJAX-call for abort().
    moveObject,                   // The letter that is being
dragged.
    mouseX = 0,                   // Global variable for current
mousePosition X.
    mouseY = 0,                   // Global variable for current
mousePosition Y.
    slotWidth = 21.3*sizeModifier,          // Width of tiles.
    slotHeight = 23.3*sizeModifier,         // Height of tiles.
    letterWidth = 20.3*sizeModifier,        // Width of letters.
    letterHeight = 22.3*sizeModifier,         // Height of letters.
    playerframeWidth = slotWidth*15,
    tileFont = 10*sizeModifier+"px Arial",
    letterFont = 14*sizeModifier+"px Arial",
    letterValFont = "bold "+9*sizeModifier+"px arial",
    swapFont = "bold "+12*sizeModifier+"px Arial",
    fontMod = 5*sizeModifier,
    fontModY = 15*sizeModifier,
    intervalTime = 3000;              // CheckTurn intervaltime.

  // Modify size of frames and fonts
  canvas.width = 320*sizeModifier;
  canvas.height = 373*sizeModifier;
  document.getElementById('wrapper').style.width = 320*sizeModifier;
  $("#score").css("font-size", Math.round(11*sizeModifier));
  $("#wrapper").css("height", Math.round(45*sizeModifier));
  $(".big").css("min-height", Math.round(40*sizeModifier));
  $(".big").css("width", playerframeWidth);
  $("#gameList").css("width", playerframeWidth);
```

```
  $(".button").css("font-size", Math.round(13*sizeModifier));
  $(".button").css("padding", Math.round(sizeModifier));
  $("#msg").css("font-size", Math.round(11*sizeModifier));

  // Prototype for rectangle with rounded corners.
  CanvasRenderingContext2D.prototype.roundRect = function (x, y, w,
h, r) {
      if (w < 2 * r) r = w / 2;
      if (h < 2 * r) r = h / 2;
      this.beginPath();
      this.moveTo(x+r, y);
      this.arcTo(x+w, y,   x+w, y+h, r);
      this.arcTo(x+w, y+h, x,   y+h, r);
      this.arcTo(x,   y+h, x,   y,   r);
      this.arcTo(x,   y,   x+w, y,   r);
      this.closePath();
      return this;
  }

  function checkProperty(x, y) {
    // All DW tiles.
    if ((x == 1 && y == 1) || (x == 2 && y == 2) || (x == 3 && y ==
3) || (x == 4 && y == 4
      || (x == 13 && y == 1) || (x == 12 && y == 2) || (x == 11 && y
== 3) || (x == 10 && y == 4)
      || (x == 13 && y == 13) || (x == 12 && y == 12) || (x == 11 &&
y == 11) || (x == 10 && y == 10)
      || (x == 1 && y == 13) || (x == 2 && y == 12) || (x == 3 && y
== 11) || (x == 4 && y == 10))){
      return "DW";

    // All TW tiles.
    } else if ((x == 0 && y == 0) || (x == 0 && y == 7) || (x == 0
&& y == 14) || (x == 7 && y == 0) ||
           (x == 14 && y == 0) || (x == 14 && y == 7) || (x == 14 &&
y == 14) || (x == 7 && y == 14)) {
      return "TW"
    }

    // All DL tiles.
    else if ((x == 3 && y == 0) || (x == 11 && y == 0) || (x == 6 &&
y == 2) || (x == 8 && y == 2
      || (x%7 == 0 && y == 3) || (x == 2 && y == 6)
      || (x == 12 && y == 6) || (x == 6 && y == 6) || (x == 8 && y
== 6) || (x == 3 && y == 7)
      || (x == 11 && y == 7) || (x == 2 && y == 8) || (x == 6 && y
== 8) || (x == 8 && y == 8)
```

```
      || (x == 12 && y == 8) || (x == 0 && y == 11) || (x == 7 && y
== 11) || (x == 14 && y == 11)
      || (x == 6 && y == 12) || (x == 8 && y == 12) || (x == 3 && y
== 14) || (x == 11 && y == 14))){
      return "DL";


    // All TL tiles.
    } else if ((x == 5 && y == 1) || (x == 9 && y == 1) || (x == 1
&& y == 5) || (x == 5 && y == 5)
      || (x == 9 && y == 5) || (x == 13 && y == 5) || (x == 1 && y
== 9) || (x == 5 && y == 9)
      || (x == 9 && y == 9) || (x == 13 && y == 9) || (x == 5 && y
== 13) || (x == 9 && y == 13)) {
      return "TL"
    } else {
      return "";
    }
  }

  function playerClass(name, timequota, gameId) {
    this.letters = [];      // Player letters.
    this.swapLetters = [];    // Letters to be swapped.
    this.timequota = timequota; // Player timequota.
    this.score = 0;       // Playerscore.
    this.playerName = name;   // Playername.
    this.gameId = gameId;     // Id of current game.
    this.myTurn = false;    // Is it my turn?
    this.turnCount = 0;     // What turn is it?
  }

  function getPlayerLetters() {
    $.post("AJAXhandler.php", { functionCall: "getPlayerLetters",
game: player.gameId, player: player.playerName},
    function(data) {
      if (data && data != "") {
        returnData = JSON.parse(data);
        for (i = 0; i < returnData.length; i++) {
          player.letters.push(returnData[i]);
        }
      } else {
        giveLetters(7);
      }
    });
  }

  function tile(rx, ry, x, y, p) {
    this.letter = null; // The letter.
```

```
    this.property = p;  // TW DW TL DL
    this.posX = x;    // To find words.
    this.posY = y;    // To find words.
    this.realX = rx;  // For drag & drop.
    this.realY = ry;  // For drag & drop.
  }

  function letter(letter, val, active) {
    this.letter = letter;
    this.val = val;
    (active) ? this.active = false : this.active = true;
    this.realX = 0;
    this.realY = 0;
  }

  function findWord(firstPos, lastPos, xPos, yPos, vertical,
horizontal, tile) {
    function sortXpos(a, b) { return a.posX - b.posX; }
    function sortYpos(a, b) { return a.posY - b.posY; }
    var activeLetters = [],
      vertical = vertical,
      horizontal = horizontal,
      firstPos = firstPos,
      lastPos = lastPos,
      yPos = yPos,
      xPos = xPos,
      findFirstLetter = false,
      findLastLetter = false,
      connected = true,
      word = "",
      points = 0,
      multiplier = 1;
    activeLetters.push(tile);

    if (horizontal) {
      while (!findFirstLetter || !findLastLetter) {
        for (var i = 0, len = game.tiles.length; i < len; i++) {
          if (firstPos == -1) findFirstLetter = true; // Dont look
for first letter is an activeletter is at end of board.
          if (game.tiles[i].posX == firstPos && game.tiles[i].posY
== yPos) {
            if (game.tiles[i].letter != null) {
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              firstPos--;
```

```
            } else {
              findFirstLetter = true;
            }
          }
          if (lastPos == 15) findLastLetter = true; // Dont look for
last letter is an activeletter is at end of board.
          if (game.tiles[i].posX == lastPos && game.tiles[i].posY ==
yPos) {
            if (game.tiles[i].letter != null) {
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              lastPos++;
            } else {
              findLastLetter = true;
            }
          }
        }
      }
      activeLetters.sort(sortXpos); // Sort array to create the word

    } else if (vertical) {
      findFirstLetter = false;
      findLastLetter = false;
      while (!findFirstLetter || !findLastLetter) {
        for (var i = 0, len = game.tiles.length; i < len; i++) {
        if (firstPos == -1) findFirstLetter = true; // Dont look for
first letter is an activeletter is at end of board.
          if (game.tiles[i].posY == firstPos && game.tiles[i].posX
== xPos) {
            if (game.tiles[i].letter != null) {
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              firstPos--;
            } else {
              findFirstLetter = true;
            }
          }
          if (lastPos == 15) findLastLetter = true; // Dont look for
last letter is an activeletter is at end of board.
          if (game.tiles[i].posY == lastPos && game.tiles[i].posX ==
xPos) {
            if (game.tiles[i].letter != null) {
```

```
              add = true;
              for (var j = 0; j < activeLetters.length; j++) {
                if (activeLetters[j] == game.tiles[i]) add = false;
              }
              if (add) activeLetters.push(game.tiles[i]);
              lastPos++;
            } else {
              findLastLetter = true;
            }
          }
        }
      }
      activeLetters.sort(sortYpos); // Sort array to create the word
    }

    // Create word from letters.
    for (i = 0; i < activeLetters.length; i++) {
      if (connected) {
        word += activeLetters[i].letter.letter;
        if (activeLetters[i].letter.active) {
          switch(activeLetters[i].property) {
            case "TL":
              points += activeLetters[i].letter.val*3;
            break;
            case "DL":
              points += activeLetters[i].letter.val*2;
            break;
            case "TW":
              points += activeLetters[i].letter.val;
              multiplier += 2;
            break;
            case "DW":
              points += activeLetters[i].letter.val;
              multiplier += 1;
            break;
            case "":
              points += activeLetters[i].letter.val;
            break;
          }
        } else {
          points += activeLetters[i].letter.val;
        }
      }
    }

    // If the word is connected and contains more than 1 letter,
move on.
    if (connected && activeLetters.length > 1) {
```

```
      returnData = new Object();        // Return data as an object.
      returnData.word = word;         // Add played word to object.
      returnData.points = points*multiplier;  // Add points to
object.
      returnData.letters = activeLetters;   // Add active letters to
object.
      return returnData;
    } else {
      return false;
    }
  }

  function findWords() {
    var activeLetters = [],
      vertical = true,
      horizontal = true,
      words = [],
      points = 0,
      allLetters = [];

    // Find the activeLetters.
    for (var i = 0; i < game.tiles.length; i++) {
      if(game.tiles[i].letter!= null && game.tiles[i].letter.active)
activeLetters.push(game.tiles[i]);
    }

    // Check if the word is vertical or horizontal.
    for (var i = 0, len = activeLetters.length; i < len; i++) {
      if (i > 0 && i <= len) {
        if (activeLetters[i].posX != activeLetters[i-1].posX)
vertical = false;
        if (activeLetters[i].posY != activeLetters[i-1].posY)
horizontal = false;
      }
    }

    if ((horizontal || vertical) && activeLetters.length > 0) {
      var yPos = 0,
        xPos = 0,
        done = 0,
        add = true,
        connected = true,
        returnData,
        passiveCount = 0;

      if (horizontal) {
        for (var i = 0, len = activeLetters.length; i < len; i++) {
```

```
          // Gather all vertical words connected to the
activeletters.
          returnData = findWord(activeLetters[i].posY,
activeLetters[i].posY, activeLetters[i].posX, 0, true, false,
activeLetters[i]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
          yPos = activeLetters[i].posY;
        }
        // Get the primary played word.
        if (activeLetters.length > 1) {
          returnData = findWord(activeLetters[0].posX,
activeLetters[0].posX, xPos, yPos, vertical, horizontal,
activeLetters[0]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
        }
        // Check that all activeLetters are included in the primary
played word.
        if (!vertical && horizontal) {
          var alCount = 0;
          if (returnData) {
            for (i = 0; i < returnData.letters.length; i++) {
              if (returnData.letters[i].letter.active) alCount++;
              if (i > 0 && returnData.letters[i].posX !=
returnData.letters[i-1].posX+1) connected = false;
            }
            if (alCount != activeLetters.length) connected = false;
            } else {
              connected = false;
            }
        }
      }

      if (vertical) {
        if (horizontal) done = 0; // If only one letter is played,
the done-variable is reset.
        for (var i = 0, len = activeLetters.length; i < len; i++) {
```

```
          // Gather all horizontal words connected to the
activeletters.
          returnData = findWord(activeLetters[i].posX,
activeLetters[i].posX, 0, activeLetters[i].posY, false, true,
activeLetters[i]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
          xPos = activeLetters[i].posX;
        }

        // Get the primary played word.
        if (vertical && !horizontal) {
          returnData = findWord(activeLetters[0].posY,
activeLetters[0].posY, xPos, yPos, vertical, horizontal,
activeLetters[0]);
          if (returnData) {
            words.push(returnData.word);
            points += returnData.points;
            for (j = 0; j < returnData.letters.length; j++) {
              allLetters.push(returnData.letters[j]);
            }
          }
        }

        // Check that all activeLetters are included in the primary
played word.
        if (vertical && !horizontal) {
          var alCount = 0;
          if (returnData) {
            for (i = 0; i < returnData.letters.length; i++) {
              if (returnData.letters[i].letter.active) alCount++;
              if (i > 0 && returnData.letters[i].posY !=
returnData.letters[i-1].posY+1) connected = false;
            }
            if (alCount != activeLetters.length) connected = false;
          } else {
            connected = false;
          }
        }
      }
```

```
      // Confirms that the word is connected to a previously played
word or first played word.
      for (i = 0; i < allLetters.length; i++) {
        if (!allLetters[i].letter.active || (allLetters[i].posX == 7
&& allLetters[i].posY == 7)) passiveCount++;
      }

      if (passiveCount == 0) connected = false;
      if (connected) {
        returnData = new Object();
        returnData.words = words;
        returnData.points = points;
        returnData.activeLetters = activeLetters;
        return returnData;
      }
    }
  }

  // Give val letters to the player from the letterpool.
  function giveLetters(val) {
    $.post("AJAXhandler.php", { functionCall: "giveLetters", game:
player.gameId, player: player.playerName, letters: val},
    function(data) {
      if (typeof data == "string") {
        returnData = JSON.parse(data);
        for (i = 0; i < returnData.length; i++) {
          player.letters.push(returnData[i]);
        }
      }
    });
  }

  function updatePlayerLetters() {
    $.post("AJAXhandler.php", { functionCall: "updatePlayerLetters",
game: player.gameId, player: player.playerName, letters:
JSON.stringify(player.letters)},
    function(data) {
      //player.letters = JSON.parse(data);
    });
  }

  function gameBoard() {
    this.tiles = []; // 15x15

    // Fills board with empty tiles for letters.
    this.fillTiles = function() {
      for (var i=0; i<15; i++) {
        for (var j=0; j<15; j++) {
```

```
          this.tiles.push(new tile(i*slotWidth, j*slotHeight, i, j,
checkProperty(j,i)));
        }
      }
    }

    // Draws board on canvas.
    this.draw = function() {
      // Draw player-frame.
      context.fillStyle = "4A4A4A";
      context.strokeStyle = "000";
      context.lineWidth = 1;
      context.fillRect(0, slotHeight*15, playerframeWidth,
slotHeight);

      // Draw swap-frame.
      context.strokeStyle = "000";
      context.fillStyle = "FFFDC4";
      context.roundRect(playerframeWidth-letterWidth*2,
slotHeight*15+1, letterWidth*2, letterHeight, 5).fill();
      context.roundRect(playerframeWidth-letterWidth*2,
slotHeight*15+1, letterWidth*2, letterHeight, 5).stroke();
      context.fillStyle = "4A4A4A";
      context.font = swapFont;
      context.fillText("SWAP",(playerframeWidth-
letterWidth*2+fontMod/2), letterHeight*16+fontModY/2);


      // Draw player-letters.
      for (var i=0; i < player.letters.length; i++) {
        drawLetter(player.letters[i], 1+i*letterWidth,
slotHeight*15, player.letters[i].active);
        player.letters[i].realX = 1+i*letterWidth;
        player.letters[i].realY = slotHeight*15;
      }

      // Draw player-letters to be swapped.
      for (var i=0; i < player.swapLetters.length; i++) {
        drawLetter(player.swapLetters[i],
(player.letters.length*letterWidth)+i*letterWidth, slotHeight*15,
false);
        player.swapLetters[i].realX =
(player.letters.length*letterWidth)+i*letterWidth;
        player.swapLetters[i].realY = slotHeight*15;
      }

      // Draw board-tiles.
      for (var i=0, len=15; i<len; i++) {
```

```
        for (var j=0; j<len; j++) {
          var color = "4A4A4A";
          context.fillStyle = color;
          context.fillRect(j*slotWidth, i*slotHeight, letterWidth,
letterHeight);
          for (var k=0; k < game.tiles.length; k++) {
            if (game.tiles[k].posX == i && game.tiles[k].posY == j)
{

              // Set backgroundcolor for each bonus-tile.
              switch(game.tiles[k].property) {
                case "DW":
                  color = "EB961E";
                  break;
                case "TW":
                  color = "D91E1E";
                  break;
                case "DL":
                  color = "63E080";
                  break;
                case "TL":
                  color = "49B9E6";
                  break;
                case "":
                  color = "4A4A4A";
              }

              /* Align to center depending on bonus-tile */
              (game.tiles[k].property == "TW" ||
game.tiles[k].property == "DW") ? modifier = 2 : modifier = +4;

              context.fillStyle = color;
              context.fillRect(j*slotWidth, i*slotHeight,
letterWidth, letterHeight);
              context.fillStyle = "FFF";
              context.font=tileFont;
              context.fillText(game.tiles[k].property,
(modifier+j*slotWidth), fontModY+i*slotHeight);
            }
          }
        }
      }

      // Draw letters on tiles.
      for (i = 0; i < game.tiles.length; i++) {
        if (game.tiles[i].letter != null) {
          drawLetter(game.tiles[i].letter, game.tiles[i].realX,
game.tiles[i].realY, game.tiles[i].letter.active);
```

```
        }
      }
    }
  }

  function checkDropTile() {
    for (var i = 0; i < game.tiles.length; i++) {
      if ((moveObject) && (mouseX >= Math.floor(game.tiles[i].realX)
&& mouseX <= Math.ceil(game.tiles[i].realX+letterWidth)) && (mouseY-
5 >= Math.floor(game.tiles[i].realY) && mouseY-5 <=
Math.ceil(game.tiles[i].realY+letterHeight))) {
        if (game.tiles[i].letter != null &&
game.tiles[i].letter.active) {
          player.letters.push(game.tiles[i].letter); // If theres is
an active letter on the tile, don't loose it.
          game.tiles[i].letter = moveObject;

        } else if (game.tiles[i].letter != null &&
!game.tiles[i].letter.active) {
          player.letters.push(moveObject);

        } else {
          game.tiles[i].letter = moveObject; // Move object to tile.
          game.tiles[i].letter.realX = game.tiles[i].realX; //
Change x-position.
          game.tiles[i].letter.realY = game.tiles[i].realY; //
Change y-position.
        }

      } else if (moveObject && mouseY > Math.floor(slotHeight*15) &&
mouseX <= Math.ceil(playerframeWidth-letterWidth*2)) { // Letter
dropped in player-frame.
        player.letters.push(moveObject);
        moveObject = null;
      } else if (moveObject && mouseY > Math.floor(slotHeight*15) &&
mouseX > Math.ceil(playerframeWidth-letterWidth*2)) { // Letter is
to be swapped!
        player.swapLetters.push(moveObject);
        moveObject = null;
      }
    }
  }

  function mouseMove(e) {
    if (dragok){
      if (e.clientX && e.clientY) {   //Handle mouse-events
        mouseX = e.clientX;
        mouseY = e.clientY;
```

```
      } else {            // Handle touch-events
        mouseX = e.targetTouches[0].pageX;
        mouseY = e.targetTouches[0].pageY;
      }
    }
  }

  function mouseDown(e) {
    dragok = true;
    if (e.clientX && e.clientY) {   //Handle mouse-events
      mouseX = e.clientX;
      mouseY = e.clientY;
    } else {            // Handle touch-events
      mouseX = e.targetTouches[0].pageX;
      mouseY = e.targetTouches[0].pageY;
    }
    canvas.addEventListener('mousemove', mouseMove, false); //Handle
mouse-events
    canvas.addEventListener('touchmove', mouseMove, false); //
Handle touch-events

    // Check if player is dragging letters from panel.
    for (var i = 0; i < player.letters.length; i++) {
      if((mouseX-5 > player.letters[i].realX && mouseX-5 <
player.letters[i].realX+letterWidth) && (mouseY-5 >
player.letters[i].realY && mouseY-5 <
player.letters[i].realY+letterHeight)) {
        moveObject = player.letters[i];
        player.letters.splice(i,1);
      }
    }

    for (var i = 0; i < player.swapLetters.length; i++) {
      if((mouseX-5 > player.swapLetters[i].realX && mouseX-5 <
player.swapLetters[i].realX+letterWidth) && (mouseY-5 >
player.swapLetters[i].realY && mouseY-5 <
player.swapLetters[i].realY+letterHeight)) {
        moveObject = player.swapLetters[i];
        player.swapLetters.splice(i,1);
      }
    }

    // Check if player is dragging letters from gameboard.
    for (i = 0; i < game.tiles.length; i++) {
      if (game.tiles[i].letter) {
        if((mouseX-5 > game.tiles[i].realX && mouseX-5 <
game.tiles[i].realX+letterWidth) && (mouseY-5 > game.tiles[i].realY
```

```
&& mouseY-5 < game.tiles[i].realY+letterHeight) &&
(game.tiles[i].letter.active)) {
          moveObject = game.tiles[i].letter;
          game.tiles[i].letter = null;
        }
      }
    }
  }

  function mouseUp() {
    checkDropTile();
    dragok = false;
    moveObject = null;

    canvas.onmousemove = null;
  }

  function clear() {
    context.clearRect(0, 0, 320*sizeModifier, 373*sizeModifier);
  }

  function drawLetter(letter, x, y, active) {
    (active) ? context.fillStyle = "FFF" : context.fillStyle =
"FFFDC4";

    context.strokeStyle = "000";
    context.lineWidth   = 1;
    context.roundRect(x, y, letterWidth, letterHeight, 5).fill();
    context.roundRect(x, y, letterWidth, letterHeight, 5).stroke();

    context.fillStyle = "4A4A4A";
    context.font = letterFont;
    context.fillText(letter.letter, x+fontMod, y+fontModY, 20);

    context.font = letterValFont;
    (letter.val >= 10) ? modifier = 1*sizeModifier : modifier =
4*sizeModifier;
    context.fillText(letter.val, x+modifier+(fontMod*2),
y+fontModY/2+1, 20);
  }

  function draw() {
    clear();
    game.draw();
    if (dragok && moveObject) {
      drawLetter(moveObject, mouseX-letterWidth/2, mouseY-
letterHeight/2, moveObject.active);
    }
```

```
  }

  function init() {
    game.fillTiles();
    getPlayerLetters();
    //checkTurn();
    checkTurnLongpolling();
    //curInterval = setInterval(checkTurn, intervalTime);

    // Add events for canvas.
    canvas.addEventListener("mousedown", mouseDown, false);
    canvas.addEventListener("mouseup", mouseUp, false);
    canvas.addEventListener("touchstart", mouseDown, false);
    canvas.addEventListener("touchend", mouseUp, false);

    // Add events for buttons.
    document.getElementById('swap').addEventListener("click", swap,
false);
    document.getElementById('play').addEventListener("click", play,
false);
    document.getElementById('pass').addEventListener("click", pass,
false);
    document.getElementById('menu').addEventListener("click", menu,
false);
  }

  function swap() {
    if (player.myTurn && player.swapLetters.length > 0) {
      $.post("AJAXhandler.php", { functionCall: "swapLetters",
letters: JSON.stringify(player.swapLetters), player:
player.playerName, game: player.gameId},
      function(data) {
        if (data && data.charAt(0) != " ") {
          returnData = JSON.parse(data);
          if(returnData[0].letter) {
            // Push all new letters
            for (i = 0; i < returnData.length; i++) {
              player.letters.push(returnData[i])
            }

            player.swapLetters = []; // Empty player swapLetters.
            player.MyTurn = false;
            setTimeout(checkTurnLongpolling, 1000);
            //curInterval = setInterval(checkTurn, intervalTime);
          }
        } else {
          $("#msg").empty();
          $("#msg").append(data);
```

```
        }

      });
    }
  }

  function menu() {
    game.tiles = [];      // Clear gameboard from previously viewed
game.
    player.letters = [];    // Clear letters from previously viewed
game.
    player.swapLetters = [];  // Clear swapLetters from previously
viewed game.
    player.turnCount = 0;   // Reset turnCount.
    request.abort();      // Abort current AJAX-call.
    //clearInterval(curInterval); // Abort intervals.
    $("#gameList").show();    // Show gamelist
    $("#app").hide();       // Hide canvas
    $("#wrapper").hide();     // Hide canvasrelated elements
  }

  function play() {
    data = findWords();
    if (player.myTurn) {
      $.post("AJAXhandler.php", { functionCall: "checkWords", words:
JSON.stringify(data), player: player.playerName, game:
player.gameId},
      function(data) {
        if (data && typeof data == "string" && data.charAt(0) != "
") {
          returnData = JSON.parse(data);

          // Deactivate all current letters on playBoard
          for (i = 0; i < game.tiles.length; i++) {
            if (game.tiles[i].letter && game.tiles[i].letter.active)
{
              game.tiles[i].letter.active = false;
            }
          }
          if(returnData.letters != 0) {
            // Push all new letters
            for (i = 0; i < returnData.letters.length; i++) {
              player.letters.push(returnData.letters[i])
            }

            // Set turn to false.
            player.myTurn = false;
            setTimeout(checkTurnLongpolling, 1000);
```

```
            //curInterval = setInterval(checkTurn, intervalTime);
          } else if (returnData.letters == 0){
            player.myTurn = false;
            setTimeout(checkTurnLongpolling, 1000);
            //curInterval = setInterval(checkTurn, intervalTime);
          }
        } else {
          $("#msg").empty();
          $("#msg").append(data+" is no word!");
        }
      });
    }
  }

  function changeButtons() {
    if (player.myTurn) {
      document.getElementById('swap').disabled = false;
      document.getElementById('play').disabled = false;
      document.getElementById('pass').disabled = false;
    } else {
      document.getElementById('swap').disabled = true;
      document.getElementById('play').disabled = true;
      document.getElementById('pass').disabled = true;
    }
    $("#msg").empty();
  }

  function pass() {
    $.post("AJAXhandler.php", { functionCall: "pass", game:
player.gameId},
    function(data) {
      player.myTurn = false;
      changeButtons();
    });
    //checkTurn();
    //curInterval = setInterval(checkTurn, intervalTime);
    setTimeout(checkTurnLongpolling, 1000);

  }

  function login() {
    $.post("AJAXhandler.php", { functionCall: "login", player:
$("#playername").val()},
      function(data) {
        if(typeof data == "string") {
          returnData = JSON.parse(data);
          player.playerName = $("#playername").val();
          $("#loginForm").hide();
```

```
          for (i = 0; i < returnData.length; i++) {
            if (returnData[i].over == 1) {
              $("#gameList").append("<div style='background-color:
#63E080;' class='game' id='"+returnData[i].game+"'>GameId:
"+returnData[i].game+"</div>");
            } else {
              $("#gameList").append("<div class='game'
id='"+returnData[i].game+"'>GameId: "+returnData[i].game+"</div>");
            }
          }

          $(".game").click(function() {
            showGame(this.id);
          });
          $(".game").css("width", playerframeWidth);
          $(".game").css("height", 60*sizeModifier);
          $(".game").css("font-size", 48*sizeModifier);
          //$(".game").css("padding-top", (60*sizeModifier)/2);
        }
      });
  }

  function showGame(game) {
    player.gameId = game;
    if(player.playerName && player.gameId) {
      $("#gameList").hide();
      $("#app").show();
      $("#wrapper").show();
      init();
    }
  }

  function checkTurn() {
    $.post("AJAXhandler.php", { functionCall: "checkTurn", player:
player.playerName, game: player.gameId, turnCount:
player.turnCount},
    function(data) {
      if(typeof data == "string" && data != "") {
        returnData = JSON.parse(data);
        //Update score, turn and poolsize.
        $("#score").empty();
        $("#score").append("Turn: <span
style='color:#63E080;'>"+returnData.turnCount+"</span>");
        $("#score").append(" Letters left: <span
style='color:#63E080;'>"+returnData.lettersLeft+"</span><br />");
        $("#score").append(returnData.score);
```

```
        //Update gameBoard.
        for (i = 0; i < returnData.gameBoard.length; i++) {
          if (returnData.gameBoard[i].letter != "") {
            if (game.tiles[i].letter && game.tiles[i].letter.active)
{
              player.letters.push(game.tiles[i].letter);
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            } else {
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            }
          }
        }
        player.turnCount = returnData.turnCount;
        if (returnData.gameOver) {
          player.myTurn = false;
          clearInterval(curInterval);
          changeButtons();
        } else if (returnData.myTurn) {
          player.myTurn = true;
          clearInterval(curInterval);
        } else {
          player.myTurn = false;
        }
      } else {
        player.myTurn = false;
      }
      changeButtons();
    });
  }

  function checkTurnLongpolling() {
    request = $.post("AJAXhandler.php", { functionCall:
"checkTurnLongpolling", player: player.playerName, game:
player.gameId, turnCount: player.turnCount},
    function(data) {
      if(typeof data == "string" && data != "" && data.charAt(0) !=
"<") {
        returnData = JSON.parse(data);
        //Update score, turn and poolsize.
        $("#score").empty();
        $("#score").append("Turn: <span
style='color:#63E080;'>"+returnData.turnCount+"</span>");
        $("#score").append(" Letters left: <span
style='color:#63E080;'>"+returnData.lettersLeft+"</span><br />");
        $("#score").append(returnData.score);

        //Update gameBoard.
        for (i = 0; i < returnData.gameBoard.length; i++) {
```

```
          if (returnData.gameBoard[i].letter) {
            if (game.tiles[i].letter && game.tiles[i].letter.active)
{
              player.letters.push(game.tiles[i].letter);
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            } else {
              game.tiles[i].letter = returnData.gameBoard[i].letter;
            }
          }
        }
        player.turnCount = returnData.turnCount;
        if (returnData.gameOver) {
          player.myTurn = false;
          //clearInterval(curInterval);
          changeButtons();
        } else if (returnData.myTurn) {
          player.myTurn = true;
          //clearInterval(curInterval);
        } else {
          player.myTurn = false;
          checkTurnLongpolling();
        }
      } else {
        player.myTurn = false;
        checkTurnLongpolling();
      }
      changeButtons();
    });
  }
};
```

Serverscript.php

```
<?php
$game = new game();
class letter {
          public  $letter = "",
                              $realX = 0,
                              $val = 0,
                              $realY = 0,
                              $active = true;

          function __construct($letter, $val) {
                    $this->letter = $letter;
                    $this->val = $val;
                    return 0;
          }
}

class game {
          public $letterPool = array();

          public function getLetterPool($gameId) {
                    $pdo=getPDO();
                    $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

                    $query="SELECT letterPool FROM games WHERE id =
:GAMEID;";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();
                    $result=$stmt->fetchColumn(0);
                    if ($result) {
                              $letters = json_decode($result);
                              $this->letterPool = array();
                              foreach ($letters as $letter) {
                                        array_push($this->letterPool,
$letter);
                              }
                    }
          }

          public function giveLetters($gameId, $playerName, $val) {
                    $this->getLetterPool($gameId);
                    if (count($this->letterPool) > 0) {
                              $min = 0;
                              $max = count($this->letterPool)-1;
                              $returnData = array();
```

```
                                // Get playerletters from database.
                                $pdo=getPDO();
                                $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

                                $query="SELECT * FROM playgames WHERE
playerName = :USERNAME AND gameId = :GAMEID;";
                                $stmt = $pdo->prepare($query);
                                $stmt->bindParam(':USERNAME',
$playerName);
                                $stmt->bindParam(':GAMEID', $gameId);
                                $stmt->execute();
                                $result=$stmt->fetchColumn(2);
                                $playerLetters = json_decode($result);
                                if (!$playerLetters) $playerLetters =
array(); // If the player doesn't have any letters, create an array
instead.

                                for ($i = 0; $i < $val; $i++) {
                                        $index = rand($min, $max);

                                        /* Checks if the letter still
is exists in pool */
                                        if(isset($this-
>letterPool[$index])) {

          array_push($returnData, $this->letterPool[$index]);

          array_push($playerLetters, $this->letterPool[$index]);
                                                unset($this-
>letterPool[$index]);
                                        } else {
                                                $i--;
                                        }
                                }

                                // Update playerletters in database

                                $query="UPDATE playgames SET
playerLetters = :LETTERS WHERE playerName = :USERNAME AND gameId =
:GAMEID;";
                                $stmt = $pdo->prepare($query);
                                $stmt->bindParam(':USERNAME',
$playerName);
                                $stmt->bindParam(':GAMEID', $gameId);
                                $stmt->bindParam(':LETTERS',
json_encode($playerLetters));
                                $stmt->execute();
```

```
                                // Update letterpool
                                $query="UPDATE games SET letterPool =
:LETTERPOOL WHERE id = :GAMEID;";
                                $stmt = $pdo->prepare($query);
                                $stmt->bindParam(':GAMEID', $gameId);
                                $stmt->bindParam(':LETTERPOOL',
json_encode($this->letterPool));
                                $stmt->execute();
                        } else {
                                $returnData = "empty";
                        }

                        return $returnData;
                }

                public function removeLetters($gameId, $playerName,
$letters) {
                        $pdo=getPDO();
                        $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

                        $newPlayerLetters = array();
                        $lettersToRemove = array();
                        $playerLetters =
json_decode(getPlayerLetters($gameId, $playerName));

                        // Get letters to be removed.
                        foreach ($letters as $letter) {
                                array_push($lettersToRemove, $letter-
>letter->letter);
                        }

                        // Remove letters to playerLetters.
                        foreach ($lettersToRemove as $letterToRemove) {
                                $index = 0;

                                foreach ($playerLetters as $key =>
$letter) {
                                        if($letter->letter ==
$letterToRemove) {

                unset($playerLetters[$key]);

                                                break;
                                        }
                                        $index++;
                                }
```

```
                    }

                    // Push letter to new array for correct format
                    foreach ($playerLetters as $letter) {
                              array_push($newPlayerLetters, $letter);
                    }

                    // Push new letters to database.
                    $query="UPDATE playgames SET playerLetters =
:LETTERS WHERE playerName = :USERNAME AND gameId = :GAMEID;";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':LETTERS',
json_encode($newPlayerLetters));
                    $stmt->bindParam(':USERNAME', $playerName);
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();
          }

          public function addLetters($gameId, $letters) {
                    $pdo=getPDO();
                    $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

                    $query="SELECT letterPool FROM games WHERE id =
:GAMEID;";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();
                    $result=$stmt->fetchAll();
                    $letterPool =
json_decode($result[0]["letterPool"]);
                    $newLetterPool = array();

                    foreach ($letterPool as $letter) {
                              array_push($newLetterPool, $letter);
                    }

                    foreach ($letters as $letter) {
                              array_push($newLetterPool, $letter);
                    }

                    $query="UPDATE games SET letterPool =
:LETTERPOOL WHERE id = :GAMEID;";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':LETTERPOOL',
json_encode($newLetterPool));
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();
```

```
        }
}

/* Returns db connection */
function getPDO() {
        $pdo = new PDO('mysql:dbname=scrabble;host=localhost',
'scrabble', 'xqQNTBsvxdtf7XQ6');
        return $pdo;
}

/* Returns TRUE if successful */
function login($username) {
        $pdo=getPDO();
        $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

        $query="SELECT * FROM players WHERE name = :USERNAME;";
        $stmt = $pdo->prepare($query);
        $stmt->bindParam(':USERNAME', $username);
        $stmt->execute();
        $result=$stmt->fetchColumn(0);

        if ($result == $username) {
                $query="UPDATE players SET status = 1 WHERE name
= :USERNAME";
                $stmt = $pdo->prepare($query);
                $stmt->bindParam(':USERNAME', $username);
                $stmt->execute();

                return TRUE;
        } else {
                return FALSE;
        }
}

function getPlayerGames($player) {
        $pdo=getPDO();
        $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

        $query="SELECT * FROM playGames INNER JOIN games on
gameId=id WHERE playerName = :USERNAME;";
        $stmt = $pdo->prepare($query);
        $stmt->bindParam(':USERNAME', $player);
        $stmt->execute();
        $result=$stmt->fetchAll();
        $returnData = array();
```

```
          for ($i = 0; $i < count($result); $i++) {
                    $object = new stdClass;
                    $object->game = $result[$i]["gameId"];
                    $object->over = $result[$i]["gameOver"];
                    array_push($returnData,$object);
          }

          return json_encode($returnData);
}

function logout($username) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          $query="UPDATE players SET status = 0 WHERE name =
:USERNAME";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':USERNAME', $username);
          if($stmt->execute()) {
                    return TRUE;
          } else {
                    return FALSE;
          }
}

function checkWords($words, $playerName, $gameId) {
          $json = json_decode($words);
          $noWord = "";
          $points = 0;
          $pdo=getPDO();
          $pdo->exec('SET NAMES UTF8');
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          /* Check all words with dictionary */
          foreach ($json->words as $word) {
                    $utf8proofWord = mb_strtolower($word,
mb_detect_encoding($word));
                    $query="SELECT * FROM se_dictionary WHERE word =
:WORD";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':WORD', $utf8proofWord);

                    $stmt->execute();
                    $row = $stmt->fetch();
                    if(count($row) <= 1) {
                              $noWord = $noWord." ".$utf8proofWord;
```

```
                    }
          }

          if ($noWord) { // Return the words that are declined.
                    return $noWord;
          } else {
                    // Add points to the player.
                    $query="UPDATE playgames SET score =
score+:POINTS WHERE playerName = :PLAYERNAME AND gameId = :GAMEID";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':PLAYERNAME', $playerName);
                    $stmt->bindParam(':POINTS', $json->points);
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();

                    // Get gameBoard from database.
                    $query="SELECT * FROM games WHERE id = :GAMEID";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();
                    $row = $stmt->fetchColumn(6);
                    $gameBoard = json_decode($row);

                    // Update gameBoard with new tiles.
                    foreach ($json->activeLetters as $letter) {
                              $letter->letter->active = 0; // Set
letter to inactive before push.
                              for ($i = 0; $i < count($gameBoard);
$i++) {
                                        if ($gameBoard[$i]->posX ==
$letter->posX && $gameBoard[$i]->posY == $letter->posY) {
                                                  $gameBoard[$i] =
$letter;
                                        }
                              }
                    }

                    // Push new gameBoard to database.
                    $query="UPDATE games SET gameBoard = :GAMEBOARD,
passCount = 0 WHERE id = :GAMEID";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':GAMEBOARD',
json_encode($gameBoard));
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->execute();

                    // Return 2 for success.
                    return 2;
```

```
        }
}

function getTiles($gameId) {
        $pdo=getPDO();
        $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

        // Get gameBoard from database.
        $query="SELECT * FROM games WHERE id = :GAMEID";
        $stmt = $pdo->prepare($query);
        $stmt->bindParam(':GAMEID', $gameId);
        $stmt->execute();
        $row = $stmt->fetchColumn(6);
        return json_decode($row);
}

function getPlayerLetters($gameId, $playerName) {
        $pdo=getPDO();
        $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

        // Get gameBoard from database.
        $query="SELECT playerLetters, started FROM playgames WHERE
playerName = :USERNAME AND gameId = :GAMEID;";
        $stmt = $pdo->prepare($query);
        $stmt->bindParam(':USERNAME', $playerName);
        $stmt->bindParam(':GAMEID', $gameId);
        $stmt->execute();
        $result=$stmt->fetchAll();

        // Set starttime for game.
        if ($result[0][1] == "0000-00-00 00:00:00") {
                $query="UPDATE playgames SET started = NOW()
WHERE playerName = :USERNAME AND gameId = :GAMEID;";
                $stmt = $pdo->prepare($query);
                $stmt->bindParam(':USERNAME', $playerName);
                $stmt->bindParam(':GAMEID', $gameId);
                $stmt->execute();
        }

        return $result[0][0];
}

function checkTurn($gameId, $playerName) {
        $pdo=getPDO();
        $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );
```

```
          // Get player turnId and the current turn from game.
          $query="SELECT turn,turnId FROM playgames INNER JOIN games
on gameId=id WHERE playerName = :USERNAME AND gameId = :GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':USERNAME', $playerName);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
          $result=$stmt->fetchAll();

          // If the turns match, it is the players turn.
          if($result[0]["turnId"] == $result[0]["turn"]) {
                    return true;
          } else {
                    return false;
          }
}

function changeTurn($gameId, $playerName) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          // Get current turn and playerCount.
          $query="SELECT turn,playercount FROM games WHERE id =
:GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
          $result=$stmt->fetchAll();

          // If it is the last players turn, the next turn will go
to the first player.
          if($result[0]["turn"] == $result[0]["playercount"]) {
                    $turn = 1;
          } else {
                    $turn = $result[0]["turn"]+1;
          }

          // Update database.
          $query="UPDATE games SET turn = :TURN, turnCount =
turnCount + 1 WHERE id = :GAMEID";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->bindParam(':TURN', $turn);
          $stmt->execute();
}
```

```
function getScore($gameId) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          $query="SELECT playerName,score,turnId,turn FROM playgames
INNER JOIN games on gameId=id WHERE gameId = :GAMEID ORDER BY score
desc;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
          $result=$stmt->fetchAll();
          $returnString = "";
          $printCount = 0;

          foreach ($result as $player) {
                    if ($player["turnId"] == $player["turn"]) {
                              $returnString = $returnString."<span
class='current'>".$player["playerName"]." - ".$player["score"]."p
</span>";
                    } else {
                              $returnString =
$returnString."<span>".$player["playerName"]." -
".$player["score"]."p </span >";
                    }
                    ($printCount % 2) ? $returnString =
$returnString."<br />" : false;
                    $printCount++;
          }

          return $returnString;
}

function getTurnCount($gameId) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          // Get current turnCount.
          $query="SELECT turnCount FROM games WHERE id = :GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
          $result=$stmt->fetchAll();

          return $result[0]["turnCount"];
}
```

```
function pass($gameId) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          $query="UPDATE games SET passCount = passCount+1 WHERE id
= :GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();

          $query="SELECT passCount FROM games WHERE id = :GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
          $result=$stmt->fetchAll();

          return $result[0]["passCount"];
}

function checkGameOver($gameId) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          // Get current turnCount.
          $query="SELECT gameOver FROM games WHERE id = :GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
          $result=$stmt->fetchAll();

          ($result[0]["gameOver"] == 1) ? $returndata = true :
$returndata = false;
          return $returndata;
}

function endGame($gameId) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          // Get current turnCount.
          $query="SELECT playerName,playerLetters,score FROM
playgames WHERE gameId = :GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
```

```
          $result=$stmt->fetchAll();
          $players = array();

          foreach ($result as $player) {
                    $newPlayerData = new stdClass;
                    $newPlayerData->playerName =
$player["playerName"];
                    $newPlayerData->score = $player["score"];
                    $letters =
json_decode($player["playerLetters"]);

                    // Remove points for remaining letters.
                    foreach ($letters as $letter) {
                              $newPlayerData->score -= $letter->val;
                    }

                    array_push($players, $newPlayerData);
          }

          // Push new data to database.
          foreach ($players as $player) {
                    $query="UPDATE playgames SET score = :SCORE,
ended = NOW() WHERE gameId = :GAMEID AND playerName = :PLAYERNAME;";
                    $stmt = $pdo->prepare($query);
                    $stmt->bindParam(':GAMEID', $gameId);
                    $stmt->bindParam(':SCORE', $player->score);
                    $stmt->bindParam(':PLAYERNAME', $player-
>playerName);
                    $stmt->execute();
          }

          // End game by setting gameOver to 1.
          $query="UPDATE games SET gameOver = 1 WHERE id =
:GAMEID;";
          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->execute();
}

function dataUsage($gameId, $playerName, $dataSize, $direction) {
          $pdo=getPDO();
          $pdo->setAttribute( PDO::ATTR_ERRMODE,
PDO::ERRMODE_WARNING );

          // Update column based on sent or received amount of data.
          if ($direction == "sent") {
```

```
                    $query="UPDATE playgames SET sent =
sent+:DATASIZE, callCount = callCount+1 WHERE gameId = :GAMEID AND
playerName = :PLAYERNAME;";
          } else {
                    $query="UPDATE playgames SET received =
received+:DATASIZE, callCount = callCount+1 WHERE gameId = :GAMEID
AND playerName = :PLAYERNAME;";
          }

          $stmt = $pdo->prepare($query);
          $stmt->bindParam(':GAMEID', $gameId);
          $stmt->bindParam(':PLAYERNAME', $playerName);
          $stmt->bindParam(':DATASIZE', $dataSize);
          $stmt->execute();
}
?>
```

AJAXhandler.php

```
<?php
$intervalTime = 10;
include("serverScript.php");

switch($_POST['functionCall']) {

  // Check if the played word exists or not.
  case "checkWords":
    if (isset($_POST['words']) && isset($_POST['player']) &&
isset($_POST['game'])) {

      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['words'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      if (checkWords($_POST['words'], $_POST['player'],
$_POST['game']) == 2) {
        // Return new letters to the player
        $game->getLetterPool($_POST['game']);             // Get
letterpool for game.
        $activeLetters = json_decode($_POST['words'])-
>activeLetters; // Get played letters.
        $letterPool_size = count($game->letterPool);          // Get
size of letterpool.
        $activeLetters_size = 0;                    // Create
variable for size of letters.
        $returnData = new stdClass;                   // Create
object to return with JSON.
        $returnData->letters = 0;                   // Create
object-variable letters.
        $returnData->score = getScore($_POST['game']);          //
Create object-variable score.

        // Cant use count sinze it contains both array and objects.
        foreach ($activeLetters as $letter) {
          $activeLetters_size++;
        }

        // There are letters to return.
        if ($letterPool_size > 0 && $letterPool_size >=
$activeLetters_size) {
          $returnData->letters = $game->giveLetters($_POST['game'],
$_POST['player'], $activeLetters_size);
```

```
        // There are not enough letters to return.
        } else if ($letterPool_size > 0 && $letterPool_size <
$activeLetters_size) {
          $returnData->letters = $game->giveLetters($_POST['game'],
$_POST['player'], $letterPool_size);
        }
        $returnData = json_encode($returnData);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData; // Return object to client.
        $game->removeLetters($_POST['game'], $_POST['player'],
$activeLetters); // Remove letters from player.
        changeTurn($_POST['game'], $_POST['player']); // Change
turn.

        // Check if the game is over.
        (count(json_decode(getPlayerLetters($_POST['game'],
$_POST['player']))) == 0) ? endgame($_POST['game']) : false;
      } else {
        $returnData = checkWords($_POST['words'], $_POST['player'],
$_POST['game']);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;
      }
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Swap letters for player.
  case "swapLetters":
    if (isset($_POST['letters']) && isset($_POST['player']) &&
isset($_POST['game'])) {
      $game->getLetterPool($_POST['game']);

      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['letters'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      // Dont allow player to swap letters if there are 10 or less
letters left in the pool.
      if (count($game->letterPool) >= 10) {
        $letters = json_decode($_POST['letters']);
```

```
        // Push letters to correct format for removeLetters-
function.
        $lettersToRemove = array();
        foreach ($letters as $letter) {
          $object = new stdClass;
          $object->letter = "";
          $object->letter->letter = $letter->letter;
          array_push($lettersToRemove, $object);
        }

        $game->removeLetters($_POST['game'], $_POST['player'],
$lettersToRemove);
        $newLetters = $game->giveLetters($_POST['game'],
$_POST['player'], count($letters));
        $game->addLetters($_POST['game'], $letters);
        $returnData = json_encode($newLetters);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;

        changeTurn($_POST['game'], $_POST['player']);
      } else {
        echo " Swapping is no longer allowed!";
      }
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Give letters to a player in a game
  case "giveLetters":
    if (isset($_POST['game']) && isset($_POST['player']) &&
isset($_POST['letters'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['letters'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      $returnData = json_encode($game->giveLetters($_POST['game'],
$_POST['player'], $_POST['letters']));
      echo $returnData;

      $dataSize += mb_strlen($returnData, 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
```

```
  break;

  // Get players current letters from a game.
  case "getPlayerLetters":
    if (isset($_POST['game']) && isset($_POST['player'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      $returnData = getPlayerLetters($_POST['game'],
$_POST['player']);
      echo $returnData;

      $dataSize += mb_strlen($returnData, 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Login player.
  case "login":
    if (isset($_POST['player'])) {
      if(login($_POST['player'])) {
        echo getPlayerGames($_POST['player']);
      } else {
        echo "";
      }
    }
  break;

  // Check whos turn it is.
  case "checkTurn":
    if (isset($_POST['game']) && isset($_POST['player']) &&
isset($_POST['turnCount'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['turnCount'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.

      if (checkTurn($_POST['game'], $_POST['player'])) {
        $game->getLetterPool($_POST['game']);
        $returnData = new stdClass;
```

```
        $returnData->gameBoard = getTiles($_POST['game']);
        $returnData->score = getScore($_POST['game']);
        $returnData->myTurn = true;
        $returnData->turnCount = getTurnCount($_POST['game']);
        $returnData->lettersLeft = count($game->letterPool);
        $returnData->gameOver = checkGameOver($_POST['game']);
        $returnData = json_encode($returnData);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;
      } else if (getTurnCount($_POST['game']) !=
$_POST['turnCount']) {
        $game->getLetterPool($_POST['game']);
        $returnData = new stdClass;
        $returnData->gameBoard = getTiles($_POST['game']);
        $returnData->score = getScore($_POST['game']);
        $returnData->myTurn = false;
        $returnData->turnCount = getTurnCount($_POST['game']);
        $returnData->lettersLeft = count($game->letterPool);
        $returnData->gameOver = checkGameOver($_POST['game']);
        $returnData = json_encode($returnData);
        $dataSize += mb_strlen($returnData, 'latin1');
        echo $returnData;
      }
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
    }
  break;

  // Alternative method to check turn with long polling.
  case "checkTurnLongpolling":
    if (isset($_POST['game']) && isset($_POST['player']) &&
isset($_POST['turnCount'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['player'], 'latin1') +
mb_strlen($_POST['game'], 'latin1') + mb_strlen($_POST['turnCount'],
'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      $dataSize = 0; // Reset for received data.
      $turn = false;

      while (!$turn) {
        if (checkTurn($_POST['game'], $_POST['player'])) {
          $game->getLetterPool($_POST['game']);
          $returnData = new stdClass;
          $returnData->gameBoard = getTiles($_POST['game']);
          $returnData->score = getScore($_POST['game']);
          $returnData->myTurn = true;
```

```
          $returnData->turnCount = getTurnCount($_POST['game']);
          $returnData->lettersLeft = count($game->letterPool);
          $returnData->gameOver = checkGameOver($_POST['game']);
          $returnData = json_encode($returnData);
          $dataSize += mb_strlen($returnData, 'latin1');

          dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
          echo $returnData;
          $turn = true;
        } else if (getTurnCount($_POST['game']) !=
$_POST['turnCount']) {
          $game->getLetterPool($_POST['game']);
          $returnData = new stdClass;
          $returnData->gameBoard = getTiles($_POST['game']);
          $returnData->score = getScore($_POST['game']);
          $returnData->myTurn = false;
          $returnData->turnCount = getTurnCount($_POST['game']);
          $returnData->lettersLeft = count($game->letterPool);
          $returnData->gameOver = checkGameOver($_POST['game']);
          $returnData = json_encode($returnData);
          $dataSize += mb_strlen($returnData, 'latin1');

          dataUsage($_POST['game'], $_POST['player'], $dataSize,
"received");
          echo $returnData;
          $turn = true;
        } else {
          sleep($intervalTime);
        }
      }
    }
  break;

  case "pass":
    if (isset($_POST['game'])) {
      // Update sent-data in bytes for player.
      $dataSize = 0 + mb_strlen($_POST['game'], 'latin1');
      dataUsage($_POST['game'], $_POST['player'], $dataSize,
"sent");
      if (pass($_POST['game']) >= 4) {
        endGame($_POST['game']);
      } else {
        changeTurn($_POST['game'], $_POST['player']);
      }
    }
  break;
}
```

```
?>
```

Style.css

```
body {
  margin: 0;
  padding: 0;
}
.big {
  width: 320px;
  min-height: 40px;
  font-family: arial, verdana, sans-serif;
  font-size: 48px;
}
#gameList {
  width: 320px;
  background-color: #4A4A4A;
  font-family: arial, verdana, sans-serif;
  font-size: 48px;
  color: #FFF;
}
#wrapper {
  background-color: #4A4A4A;
  width: 320px;
  height: 45px;
  margin-top: 1px;
}
.game {
  width: 320px;
  height: 60px;
  border-top: 1px solid;
  text-align: center;
}
#score {
  float: left;
  padding-left: 5px;
  min-width: 140px;
  font-size: 11px;
  color: #FFF;
  font-family: arial, verdana, sans-serif;
}
#msg {
  float: right;
  min-width: 50px;
  font-size: 11px;
  font-family: arial, verdana, sans-serif;
  color: #FFF;
}
.button {
  font-size: 13px;
  float: right;
```

```
  margin: 0px;
  padding: 0px;
  font-family: arial, verdana, sans-serif;
  color: #363636;
}
.current {
  color: #49B9E6;
  padding: 0;
  margin: 0;
}
```

Appendix D - MySQL-databas struktur

```
-- phpMyAdmin SQL Dump
-- version 3.4.5
-- http://www.phpmyadmin.net
--
-- Host: localhost
-- Generation Time: May 25, 2012 at 07:41 AM
-- Server version: 5.5.16
-- PHP Version: 5.3.8

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Database: `scrabble`
--

-- --------------------------------------------------------

--
-- Table structure for table `games`
--

CREATE TABLE IF NOT EXISTS `games` (
  `id` smallint(11) NOT NULL AUTO_INCREMENT,
  `passCount` tinyint(1) NOT NULL DEFAULT '0',
  `turn` tinyint(1) NOT NULL DEFAULT '1',
  `playerCount` tinyint(1) NOT NULL,
  `turnCount` tinyint(3) NOT NULL DEFAULT '1',
  `letterPool` varchar(7000) NOT NULL DEFAULT
    '[{"letter":"X","realX":0,"val":10,"realY":0,"active":true},{"letter
    ":"Z","realX":0,"val":10,"realY":0,"active":true},{"letter":"Q","rea
    lX":0,"val":10,"realY":0,"active":true},{"letter":"J","realX":0,"val
    ":8,"realY":0,"active":true},{"letter":"\\u00c5","realX":0,"val":4,"
    realY":0,"active":true},{"letter":"\\u00c4","realX":0,"val":4,"realY
    ":0,"active":true},{"letter":"F","realX":0,"val":4,"realY":0,"active
    ":true},{"letter":"\\u00d6","realX":0,"val":4,"realY":0,"active":tru
    e},{"letter":"Y","realX":0,"val":8,"realY":0,"active":true},{"letter
    ":"B","realX":0,"val":4,"realY":0,"active":true},{"letter":"C","real
    X":0,"val":8,"realY":0,"active":true},{"letter":"V","realX":0,"val":
    4,"realY":0,"active":true},{"letter":"\\u00c5","realX":0,"val":4,"re
    alY":0,"active":true},{"letter":"\\u00c4","realX":0,"val":4,"realY":
    0,"active":true},{"letter":"F","realX":0,"val":4,"realY":0,"active":
    true},{"letter":"\\u00d6","realX":0,"val":4,"realY":0,"active":true}
    ,{"letter":"Y","realX":0,"val":8,"realY":0,"active":true},{"letter":
    "B","realX":0,"val":4,"realY":0,"active":true},{"letter":"C","realX"
    :0,"val":8,"realY":0,"active":true},{"letter":"V","realX":0,"val":4,
    "realY":0,"active":true},{"letter":"H","realX":0,"val":3,"realY":0,"
```

```
active":true},{"letter":"M","realX":0,"val":3,"realY":0,"active":tru
e},{"letter":"K","realX":0,"val":3,"realY":0,"active":true},{"letter
":"P","realX":0,"val":3,"realY":0,"active":true},{"letter":"U","real
X":0,"val":3,"realY":0,"active":true},{"letter":"H","realX":0,"val":
3,"realY":0,"active":true},{"letter":"M","realX":0,"val":3,"realY":0
,"active":true},{"letter":"K","realX":0,"val":3,"realY":0,"active":t
rue},{"letter":"P","realX":0,"val":3,"realY":0,"active":true},{"lett
er":"U","realX":0,"val":3,"realY":0,"active":true},{"letter":"H","re
alX":0,"val":3,"realY":0,"active":true},{"letter":"M","realX":0,"val
":3,"realY":0,"active":true},{"letter":"K","realX":0,"val":3,"realY"
:0,"active":true},{"letter":"P","realX":0,"val":3,"realY":0,"active"
:true},{"letter":"U","realX":0,"val":3,"realY":0,"active":true},{"le
tter":"G","realX":0,"val":2,"realY":0,"active":true},{"letter":"G","
realX":0,"val":2,"realY":0,"active":true},{"letter":"G","realX":0,"v
al":2,"realY":0,"active":true},{"letter":"G","realX":0,"val":2,"real
Y":0,"active":true},{"letter":"O","realX":0,"val":2,"realY":0,"activ
e":true},{"letter":"O","realX":0,"val":2,"realY":0,"active":true},{"
letter":"O","realX":0,"val":2,"realY":0,"active":true},{"letter":"O"
,"realX":0,"val":2,"realY":0,"active":true},{"letter":"O","realX":0,
"val":2,"realY":0,"active":true},{"letter":"I","realX":0,"val":1,"re
alY":0,"active":true},{"letter":"I","realX":0,"val":1,"realY":0,"act
ive":true},{"letter":"I","realX":0,"val":1,"realY":0,"active":true},
{"letter":"I","realX":0,"val":1,"realY":0,"active":true},{"letter":"
I","realX":0,"val":1,"realY":0,"active":true},{"letter":"I","realX":
0,"val":1,"realY":0,"active":true},{"letter":"D","realX":0,"val":1,"
realY":0,"active":true},{"letter":"T","realX":0,"val":1,"realY":0,"a
ctive":true},{"letter":"L","realX":0,"val":1,"realY":0,"active":true
},{"letter":"N","realX":0,"val":1,"realY":0,"active":true},{"letter"
:"D","realX":0,"val":1,"realY":0,"active":true},{"letter":"T","realX
":0,"val":1,"realY":0,"active":true},{"letter":"L","realX":0,"val":1
,"realY":0,"active":true},{"letter":"N","realX":0,"val":1,"realY":0,
"active":true},{"letter":"D","realX":0,"val":1,"realY":0,"active":tr
ue},{"letter":"T","realX":0,"val":1,"realY":0,"active":true},{"lette
r":"L","realX":0,"val":1,"realY":0,"active":true},{"letter":"N","rea
lX":0,"val":1,"realY":0,"active":true},{"letter":"D","realX":0,"val"
:1,"realY":0,"active":true},{"letter":"T","realX":0,"val":1,"realY":
0,"active":true},{"letter":"L","realX":0,"val":1,"realY":0,"active":
true},{"letter":"N","realX":0,"val":1,"realY":0,"active":true},{"let
ter":"D","realX":0,"val":1,"realY":0,"active":true},{"letter":"T","r
ealX":0,"val":1,"realY":0,"active":true},{"letter":"L","realX":0,"va
l":1,"realY":0,"active":true},{"letter":"N","realX":0,"val":1,"realY
":0,"active":true},{"letter":"D","realX":0,"val":1,"realY":0,"active
":true},{"letter":"T","realX":0,"val":1,"realY":0,"active":true},{"l
etter":"L","realX":0,"val":1,"realY":0,"active":true},{"letter":"N",
"realX":0,"val":1,"realY":0,"active":true},{"letter":"D","realX":0,"
val":1,"realY":0,"active":true},{"letter":"T","realX":0,"val":1,"rea
lY":0,"active":true},{"letter":"L","realX":0,"val":1,"realY":0,"acti
ve":true},{"letter":"N","realX":0,"val":1,"realY":0,"active":true},{
"letter":"S","realX":0,"val":1,"realY":0,"active":true},{"letter":"E
","realX":0,"val":1,"realY":0,"active":true},{"letter":"S","realX":0
,"val":1,"realY":0,"active":true},{"letter":"E","realX":0,"val":1,"r
ealY":0,"active":true},{"letter":"S","realX":0,"val":1,"realY":0,"ac
tive":true},{"letter":"E","realX":0,"val":1,"realY":0,"active":true}
,{"letter":"S","realX":0,"val":1,"realY":0,"active":true},{"letter":
"E","realX":0,"val":1,"realY":0,"active":true},{"letter":"S","realX"
:0,"val":1,"realY":0,"active":true},{"letter":"E","realX":0,"val":1,
```

```
  "realY":0,"active":true},{"letter":"S","realX":0,"val":1,"realY":0,"
  active":true},{"letter":"E","realX":0,"val":1,"realY":0,"active":tru
  e},{"letter":"S","realX":0,"val":1,"realY":0,"active":true},{"letter
  ":"E","realX":0,"val":1,"realY":0,"active":true},{"letter":"S","real
  X":0,"val":1,"realY":0,"active":true},{"letter":"E","realX":0,"val":
  1,"realY":0,"active":true},{"letter":"R","realX":0,"val":1,"realY":0
  ,"active":true},{"letter":"A","realX":0,"val":1,"realY":0,"active":t
  rue},{"letter":"R","realX":0,"val":1,"realY":0,"active":true},{"lett
  er":"A","realX":0,"val":1,"realY":0,"active":true},{"letter":"R","re
  alX":0,"val":1,"realY":0,"active":true},{"letter":"A","realX":0,"val
  ":1,"realY":0,"active":true},{"letter":"R","realX":0,"val":1,"realY"
  :0,"active":true},{"letter":"A","realX":0,"val":1,"realY":0,"active"
  :true},{"letter":"R","realX":0,"val":1,"realY":0,"active":true},{"le
  tter":"A","realX":0,"val":1,"realY":0,"active":true},{"letter":"R","
  realX":0,"val":1,"realY":0,"active":true},{"letter":"A","realX":0,"v
  al":1,"realY":0,"active":true},{"letter":"R","realX":0,"val":1,"real
  Y":0,"active":true},{"letter":"A","realX":0,"val":1,"realY":0,"activ
  e":true},{"letter":"R","realX":0,"val":1,"realY":0,"active":true},{"
  letter":"A","realX":0,"val":1,"realY":0,"active":true},{"letter":"R"
  ,"realX":0,"val":1,"realY":0,"active":true},{"letter":"A","realX":0,
  "val":1,"realY":0,"active":true}]',
`gameBoard` varchar(30000) NOT NULL DEFAULT
  '[{"letter":null,"property":"TW","posX":0,"posY":0,"realX":0,"realY"
  :0},{"letter":null,"property":"","posX":0,"posY":1,"realX":0,"realY"
  :31},{"letter":null,"property":"","posX":0,"posY":2,"realX":0,"realY
  ":62},{"letter":null,"property":"DL","posX":0,"posY":3,"realX":0,"re
  alY":93},{"letter":null,"property":"","posX":0,"posY":4,"realX":0,"r
  ealY":124},{"letter":null,"property":"","posX":0,"posY":5,"realX":0,
  "realY":155},{"letter":null,"property":"","posX":0,"posY":6,"realX":
  0,"realY":186},{"letter":null,"property":"TW","posX":0,"posY":7,"rea
  lX":0,"realY":217},{"letter":null,"property":"","posX":0,"posY":8,"r
  ealX":0,"realY":248},{"letter":null,"property":"","posX":0,"posY":9,
  "realX":0,"realY":279},{"letter":null,"property":"","posX":0,"posY":
  10,"realX":0,"realY":310},{"letter":null,"property":"DL","posX":0,"p
  osY":11,"realX":0,"realY":341},{"letter":null,"property":"","posX":0
  ,"posY":12,"realX":0,"realY":372},{"letter":null,"property":"","posX
  ":0,"posY":13,"realX":0,"realY":403},{"letter":null,"property":"TW",
  "posX":0,"posY":14,"realX":0,"realY":434},{"letter":null,"property":
  "","posX":1,"posY":0,"realX":31,"realY":0},{"letter":null,"property"
  :"DW","posX":1,"posY":1,"realX":31,"realY":31},{"letter":null,"prope
  rty":"","posX":1,"posY":2,"realX":31,"realY":62},{"letter":null,"pro
  perty":"","posX":1,"posY":3,"realX":31,"realY":93},{"letter":null,"p
  roperty":"","posX":1,"posY":4,"realX":31,"realY":124},{"letter":null
  ,"property":"TL","posX":1,"posY":5,"realX":31,"realY":155},{"letter"
  :null,"property":"","posX":1,"posY":6,"realX":31,"realY":186},{"lett
  er":null,"property":"","posX":1,"posY":7,"realX":31,"realY":217},{"l
  etter":null,"property":"","posX":1,"posY":8,"realX":31,"realY":248},
  {"letter":null,"property":"TL","posX":1,"posY":9,"realX":31,"realY":
  279},{"letter":null,"property":"","posX":1,"posY":10,"realX":31,"rea
  lY":310},{"letter":null,"property":"","posX":1,"posY":11,"realX":31,
  "realY":341},{"letter":null,"property":"","posX":1,"posY":12,"realX"
  :31,"realY":372},{"letter":null,"property":"DW","posX":1,"posY":13,"
  realX":31,"realY":403},{"letter":null,"property":"","posX":1,"posY":
  14,"realX":31,"realY":434},{"letter":null,"property":"","posX":2,"po
  sY":0,"realX":62,"realY":0},{"letter":null,"property":"","posX":2,"p
  osY":1,"realX":62,"realY":31},{"letter":null,"property":"DW","posX":
```

```
2,"posY":2,"realX":62,"realY":62},{"letter":null,"property":"","posX
":2,"posY":3,"realX":62,"realY":93},{"letter":null,"property":"","po
sX":2,"posY":4,"realX":62,"realY":124},{"letter":null,"property":"",
"posX":2,"posY":5,"realX":62,"realY":155},{"letter":null,"property":
"DL","posX":2,"posY":6,"realX":62,"realY":186},{"letter":null,"prope
rty":"","posX":2,"posY":7,"realX":62,"realY":217},{"letter":null,"pr
operty":"DL","posX":2,"posY":8,"realX":62,"realY":248},{"letter":nul
l,"property":"","posX":2,"posY":9,"realX":62,"realY":279},{"letter":
null,"property":"","posX":2,"posY":10,"realX":62,"realY":310},{"lett
er":null,"property":"","posX":2,"posY":11,"realX":62,"realY":341},{"
letter":null,"property":"DW","posX":2,"posY":12,"realX":62,"realY":3
72},{"letter":null,"property":"","posX":2,"posY":13,"realX":62,"real
Y":403},{"letter":null,"property":"","posX":2,"posY":14,"realX":62,"
realY":434},{"letter":null,"property":"DL","posX":3,"posY":0,"realX"
:93,"realY":0},{"letter":null,"property":"","posX":3,"posY":1,"realX
":93,"realY":31},{"letter":null,"property":"","posX":3,"posY":2,"rea
lX":93,"realY":62},{"letter":null,"property":"DW","posX":3,"posY":3,
"realX":93,"realY":93},{"letter":null,"property":"","posX":3,"posY":
4,"realX":93,"realY":124},{"letter":null,"property":"","posX":3,"pos
Y":5,"realX":93,"realY":155},{"letter":null,"property":"","posX":3,"
posY":6,"realX":93,"realY":186},{"letter":null,"property":"DL","posX
":3,"posY":7,"realX":93,"realY":217},{"letter":null,"property":"","p
osX":3,"posY":8,"realX":93,"realY":248},{"letter":null,"property":""
,"posX":3,"posY":9,"realX":93,"realY":279},{"letter":null,"property"
:"","posX":3,"posY":10,"realX":93,"realY":310},{"letter":null,"prope
rty":"DW","posX":3,"posY":11,"realX":93,"realY":341},{"letter":null,
"property":"","posX":3,"posY":12,"realX":93,"realY":372},{"letter":n
ull,"property":"","posX":3,"posY":13,"realX":93,"realY":403},{"lette
r":null,"property":"DL","posX":3,"posY":14,"realX":93,"realY":434},{
"letter":null,"property":"","posX":4,"posY":0,"realX":124,"realY":0}
,{"letter":null,"property":"","posX":4,"posY":1,"realX":124,"realY":
31},{"letter":null,"property":"","posX":4,"posY":2,"realX":124,"real
Y":62},{"letter":null,"property":"","posX":4,"posY":3,"realX":124,"r
ealY":93},{"letter":null,"property":"DW","posX":4,"posY":4,"realX":1
24,"realY":124},{"letter":null,"property":"","posX":4,"posY":5,"real
X":124,"realY":155},{"letter":null,"property":"","posX":4,"posY":6,"
realX":124,"realY":186},{"letter":null,"property":"","posX":4,"posY"
:7,"realX":124,"realY":217},{"letter":null,"property":"","posX":4,"p
osY":8,"realX":124,"realY":248},{"letter":null,"property":"","posX":
4,"posY":9,"realX":124,"realY":279},{"letter":null,"property":"DW","
posX":4,"posY":10,"realX":124,"realY":310},{"letter":null,"property"
:"","posX":4,"posY":11,"realX":124,"realY":341},{"letter":null,"prop
erty":"","posX":4,"posY":12,"realX":124,"realY":372},{"letter":null,
"property":"","posX":4,"posY":13,"realX":124,"realY":403},{"letter":
null,"property":"","posX":4,"posY":14,"realX":124,"realY":434},{"let
ter":null,"property":"","posX":5,"posY":0,"realX":155,"realY":0},{"l
etter":null,"property":"TL","posX":5,"posY":1,"realX":155,"realY":31
},{"letter":null,"property":"","posX":5,"posY":2,"realX":155,"realY"
:62},{"letter":null,"property":"","posX":5,"posY":3,"realX":155,"rea
lY":93},{"letter":null,"property":"","posX":5,"posY":4,"realX":155,"
realY":124},{"letter":null,"property":"TL","posX":5,"posY":5,"realX"
:155,"realY":155},{"letter":null,"property":"","posX":5,"posY":6,"re
alX":155,"realY":186},{"letter":null,"property":"","posX":5,"posY":7
,"realX":155,"realY":217},{"letter":null,"property":"","posX":5,"pos
Y":8,"realX":155,"realY":248},{"letter":null,"property":"TL","posX":
5,"posY":9,"realX":155,"realY":279},{"letter":null,"property":"","po
```

```
sX":5,"posY":10,"realX":155,"realY":310},{"letter":null,"property":"
","posX":5,"posY":11,"realX":155,"realY":341},{"letter":null,"proper
ty":"","posX":5,"posY":12,"realX":155,"realY":372},{"letter":null,"p
roperty":"TL","posX":5,"posY":13,"realX":155,"realY":403},{"letter":
null,"property":"","posX":5,"posY":14,"realX":155,"realY":434},{"let
ter":null,"property":"","posX":6,"posY":0,"realX":186,"realY":0},{"l
etter":null,"property":"","posX":6,"posY":1,"realX":186,"realY":31},
{"letter":null,"property":"DL","posX":6,"posY":2,"realX":186,"realY"
:62},{"letter":null,"property":"","posX":6,"posY":3,"realX":186,"rea
lY":93},{"letter":null,"property":"","posX":6,"posY":4,"realX":186,"
realY":124},{"letter":null,"property":"","posX":6,"posY":5,"realX":1
86,"realY":155},{"letter":null,"property":"DL","posX":6,"posY":6,"re
alX":186,"realY":186},{"letter":null,"property":"","posX":6,"posY":7
,"realX":186,"realY":217},{"letter":null,"property":"DL","posX":6,"p
osY":8,"realX":186,"realY":248},{"letter":null,"property":"","posX":
6,"posY":9,"realX":186,"realY":279},{"letter":null,"property":"","po
sX":6,"posY":10,"realX":186,"realY":310},{"letter":null,"property":"
","posX":6,"posY":11,"realX":186,"realY":341},{"letter":null,"proper
ty":"DL","posX":6,"posY":12,"realX":186,"realY":372},{"letter":null,
"property":"","posX":6,"posY":13,"realX":186,"realY":403},{"letter":
null,"property":"","posX":6,"posY":14,"realX":186,"realY":434},{"let
ter":null,"property":"TW","posX":7,"posY":0,"realX":217,"realY":0},{
"letter":null,"property":"","posX":7,"posY":1,"realX":217,"realY":31
},{"letter":null,"property":"","posX":7,"posY":2,"realX":217,"realY"
:62},{"letter":null,"property":"DL","posX":7,"posY":3,"realX":217,"r
ealY":93},{"letter":null,"property":"","posX":7,"posY":4,"realX":217
,"realY":124},{"letter":null,"property":"","posX":7,"posY":5,"realX"
:217,"realY":155},{"letter":null,"property":"","posX":7,"posY":6,"re
alX":217,"realY":186},{"letter":null,"property":"","posX":7,"posY":7
,"realX":217,"realY":217},{"letter":null,"property":"","posX":7,"pos
Y":8,"realX":217,"realY":248},{"letter":null,"property":"","posX":7,
"posY":9,"realX":217,"realY":279},{"letter":null,"property":"","posX
":7,"posY":10,"realX":217,"realY":310},{"letter":null,"property":"DL
","posX":7,"posY":11,"realX":217,"realY":341},{"letter":null,"proper
ty":"","posX":7,"posY":12,"realX":217,"realY":372},{"letter":null,"p
roperty":"","posX":7,"posY":13,"realX":217,"realY":403},{"letter":nu
ll,"property":"TW","posX":7,"posY":14,"realX":217,"realY":434},{"let
ter":null,"property":"","posX":8,"posY":0,"realX":248,"realY":0},{"l
etter":null,"property":"","posX":8,"posY":1,"realX":248,"realY":31},
{"letter":null,"property":"DL","posX":8,"posY":2,"realX":248,"realY"
:62},{"letter":null,"property":"","posX":8,"posY":3,"realX":248,"rea
lY":93},{"letter":null,"property":"","posX":8,"posY":4,"realX":248,"
realY":124},{"letter":null,"property":"","posX":8,"posY":5,"realX":2
48,"realY":155},{"letter":null,"property":"DL","posX":8,"posY":6,"re
alX":248,"realY":186},{"letter":null,"property":"","posX":8,"posY":7
,"realX":248,"realY":217},{"letter":null,"property":"DL","posX":8,"p
osY":8,"realX":248,"realY":248},{"letter":null,"property":"","posX":
8,"posY":9,"realX":248,"realY":279},{"letter":null,"property":"","po
sX":8,"posY":10,"realX":248,"realY":310},{"letter":null,"property":"
","posX":8,"posY":11,"realX":248,"realY":341},{"letter":null,"proper
ty":"DL","posX":8,"posY":12,"realX":248,"realY":372},{"letter":null,
"property":"","posX":8,"posY":13,"realX":248,"realY":403},{"letter":
null,"property":"","posX":8,"posY":14,"realX":248,"realY":434},{"let
ter":null,"property":"","posX":9,"posY":0,"realX":279,"realY":0},{"l
etter":null,"property":"TL","posX":9,"posY":1,"realX":279,"realY":31
},{"letter":null,"property":"","posX":9,"posY":2,"realX":279,"realY"
```

```
:62},{"letter":null,"property":"","posX":9,"posY":3,"realX":279,"rea
lY":93},{"letter":null,"property":"","posX":9,"posY":4,"realX":279,"
realY":124},{"letter":null,"property":"TL","posX":9,"posY":5,"realX"
:279,"realY":155},{"letter":null,"property":"","posX":9,"posY":6,"re
alX":279,"realY":186},{"letter":null,"property":"","posX":9,"posY":7
,"realX":279,"realY":217},{"letter":null,"property":"","posX":9,"pos
Y":8,"realX":279,"realY":248},{"letter":null,"property":"TL","posX":
9,"posY":9,"realX":279,"realY":279},{"letter":null,"property":"","po
sX":9,"posY":10,"realX":279,"realY":310},{"letter":null,"property":"
","posX":9,"posY":11,"realX":279,"realY":341},{"letter":null,"proper
ty":"","posX":9,"posY":12,"realX":279,"realY":372},{"letter":null,"p
roperty":"TL","posX":9,"posY":13,"realX":279,"realY":403},{"letter":
null,"property":"","posX":9,"posY":14,"realX":279,"realY":434},{"let
ter":null,"property":"","posX":10,"posY":0,"realX":310,"realY":0},{"
letter":null,"property":"","posX":10,"posY":1,"realX":310,"realY":31
},{"letter":null,"property":"","posX":10,"posY":2,"realX":310,"realY
":62},{"letter":null,"property":"","posX":10,"posY":3,"realX":310,"r
ealY":93},{"letter":null,"property":"DW","posX":10,"posY":4,"realX":
310,"realY":124},{"letter":null,"property":"","posX":10,"posY":5,"re
alX":310,"realY":155},{"letter":null,"property":"","posX":10,"posY":
6,"realX":310,"realY":186},{"letter":null,"property":"","posX":10,"p
osY":7,"realX":310,"realY":217},{"letter":null,"property":"","posX":
10,"posY":8,"realX":310,"realY":248},{"letter":null,"property":"","p
osX":10,"posY":9,"realX":310,"realY":279},{"letter":null,"property":
"DW","posX":10,"posY":10,"realX":310,"realY":310},{"letter":null,"pr
operty":"","posX":10,"posY":11,"realX":310,"realY":341},{"letter":nu
ll,"property":"","posX":10,"posY":12,"realX":310,"realY":372},{"lett
er":null,"property":"","posX":10,"posY":13,"realX":310,"realY":403},
{"letter":null,"property":"","posX":10,"posY":14,"realX":310,"realY"
:434},{"letter":null,"property":"DL","posX":11,"posY":0,"realX":341,
"realY":0},{"letter":null,"property":"","posX":11,"posY":1,"realX":3
41,"realY":31},{"letter":null,"property":"","posX":11,"posY":2,"real
X":341,"realY":62},{"letter":null,"property":"DW","posX":11,"posY":3
,"realX":341,"realY":93},{"letter":null,"property":"","posX":11,"pos
Y":4,"realX":341,"realY":124},{"letter":null,"property":"","posX":11
,"posY":5,"realX":341,"realY":155},{"letter":null,"property":"","pos
X":11,"posY":6,"realX":341,"realY":186},{"letter":null,"property":"D
L","posX":11,"posY":7,"realX":341,"realY":217},{"letter":null,"prope
rty":"","posX":11,"posY":8,"realX":341,"realY":248},{"letter":null,"
property":"","posX":11,"posY":9,"realX":341,"realY":279},{"letter":n
ull,"property":"","posX":11,"posY":10,"realX":341,"realY":310},{"let
ter":null,"property":"DW","posX":11,"posY":11,"realX":341,"realY":34
1},{"letter":null,"property":"","posX":11,"posY":12,"realX":341,"rea
lY":372},{"letter":null,"property":"","posX":11,"posY":13,"realX":34
1,"realY":403},{"letter":null,"property":"DL","posX":11,"posY":14,"r
ealX":341,"realY":434},{"letter":null,"property":"","posX":12,"posY"
:0,"realX":372,"realY":0},{"letter":null,"property":"","posX":12,"po
sY":1,"realX":372,"realY":31},{"letter":null,"property":"DW","posX":
12,"posY":2,"realX":372,"realY":62},{"letter":null,"property":"","po
sX":12,"posY":3,"realX":372,"realY":93},{"letter":null,"property":""
,"posX":12,"posY":4,"realX":372,"realY":124},{"letter":null,"propert
y":"","posX":12,"posY":5,"realX":372,"realY":155},{"letter":null,"pr
operty":"DL","posX":12,"posY":6,"realX":372,"realY":186},{"letter":n
ull,"property":"","posX":12,"posY":7,"realX":372,"realY":217},{"lett
er":null,"property":"DL","posX":12,"posY":8,"realX":372,"realY":248}
,{"letter":null,"property":"","posX":12,"posY":9,"realX":372,"realY"
```

```
    :279},{"letter":null,"property":"","posX":12,"posY":10,"realX":372,"
    realY":310},{"letter":null,"property":"","posX":12,"posY":11,"realX"
    :372,"realY":341},{"letter":null,"property":"DW","posX":12,"posY":12
    ,"realX":372,"realY":372},{"letter":null,"property":"","posX":12,"po
    sY":13,"realX":372,"realY":403},{"letter":null,"property":"","posX":
    12,"posY":14,"realX":372,"realY":434},{"letter":null,"property":"","
    posX":13,"posY":0,"realX":403,"realY":0},{"letter":null,"property":"
    DW","posX":13,"posY":1,"realX":403,"realY":31},{"letter":null,"prope
    rty":"","posX":13,"posY":2,"realX":403,"realY":62},{"letter":null,"p
    roperty":"","posX":13,"posY":3,"realX":403,"realY":93},{"letter":nul
    l,"property":"","posX":13,"posY":4,"realX":403,"realY":124},{"letter
    ":null,"property":"TL","posX":13,"posY":5,"realX":403,"realY":155},{
    "letter":null,"property":"","posX":13,"posY":6,"realX":403,"realY":1
    86},{"letter":null,"property":"","posX":13,"posY":7,"realX":403,"rea
    lY":217},{"letter":null,"property":"","posX":13,"posY":8,"realX":403
    ,"realY":248},{"letter":null,"property":"TL","posX":13,"posY":9,"rea
    lX":403,"realY":279},{"letter":null,"property":"","posX":13,"posY":1
    0,"realX":403,"realY":310},{"letter":null,"property":"","posX":13,"p
    osY":11,"realX":403,"realY":341},{"letter":null,"property":"","posX"
    :13,"posY":12,"realX":403,"realY":372},{"letter":null,"property":"DW
    ","posX":13,"posY":13,"realX":403,"realY":403},{"letter":null,"prope
    rty":"","posX":13,"posY":14,"realX":403,"realY":434},{"letter":null,
    "property":"TW","posX":14,"posY":0,"realX":434,"realY":0},{"letter":
    null,"property":"","posX":14,"posY":1,"realX":434,"realY":31},{"lett
    er":null,"property":"","posX":14,"posY":2,"realX":434,"realY":62},{"
    letter":null,"property":"DL","posX":14,"posY":3,"realX":434,"realY":
    93},{"letter":null,"property":"","posX":14,"posY":4,"realX":434,"rea
    lY":124},{"letter":null,"property":"","posX":14,"posY":5,"realX":434
    ,"realY":155},{"letter":null,"property":"","posX":14,"posY":6,"realX
    ":434,"realY":186},{"letter":null,"property":"TW","posX":14,"posY":7
    ,"realX":434,"realY":217},{"letter":null,"property":"","posX":14,"po
    sY":8,"realX":434,"realY":248},{"letter":null,"property":"","posX":1
    4,"posY":9,"realX":434,"realY":279},{"letter":null,"property":"","po
    sX":14,"posY":10,"realX":434,"realY":310},{"letter":null,"property":
    "DL","posX":14,"posY":11,"realX":434,"realY":341},{"letter":null,"pr
    operty":"","posX":14,"posY":12,"realX":434,"realY":372},{"letter":nu
    ll,"property":"","posX":14,"posY":13,"realX":434,"realY":403},{"lett
    er":null,"property":"TW","posX":14,"posY":14,"realX":434,"realY":434
    }]',
  `gameOver` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=29 ;

-- --------------------------------------------------------

--
-- Table structure for table `players`
--

CREATE TABLE IF NOT EXISTS `players` (
  `name` varchar(20) NOT NULL,
  `wins` tinyint(4) NOT NULL DEFAULT '0',
  `losses` tinyint(4) NOT NULL DEFAULT '0',
  `status` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- --------------------------------------------------------

--
-- Table structure for table `playgames`
--

CREATE TABLE IF NOT EXISTS `playgames` (
  `gameId` smallint(11) NOT NULL,
  `playerName` varchar(20) NOT NULL,
  `playerLetters` varchar(5000) NOT NULL,
  `timeQuota` smallint(6) NOT NULL DEFAULT '0',
  `score` smallint(5) NOT NULL DEFAULT '0',
  `turnId` tinyint(1) NOT NULL DEFAULT '0',
  `sent` int(11) NOT NULL DEFAULT '0',
  `received` int(11) NOT NULL DEFAULT '0',
  `callCount` smallint(4) NOT NULL DEFAULT '0',
  `started` datetime NOT NULL,
  `ended` datetime NOT NULL,
  PRIMARY KEY (`gameId`,`playerName`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `se_dictionary`
--

CREATE TABLE IF NOT EXISTS `se_dictionary` (
  `word` varchar(40) COLLATE utf8_bin NOT NULL,
  `wordLength` tinyint(2) NOT NULL,
  PRIMARY KEY (`word`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_bin;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Appendix E - Intervjuprotokoll - testkandidat 1

Kön: Man
Ålder: 20-25
Datum: 2011-05-13

Du har under experimentet spelat fyra omgångar. Märkte du någon skillnad mellan dessa?
Nej.

Om ja, vilken spelomgång tyckte du fungerade bäst respektive sämst?

Hur viktigt är det att ord som motståndare har spelat syns direkt?
Inte viktigt alls. När jag spelade Wordfeud kollade jag telefonen lite då och då men jag brukade inte sitta och vänta på att någon skulle svara.

Vilken mobilenhet använde du under experimentet?
iPhone 4S.

Vilken webbläsare använde du under experimentet?
Safari.

Vilken mobiloperatör har du?
Halebop.

Appendix F - Intervjuprotokoll - testkandidat 2

Kön: Man
Ålder: 20-25
Datum: 2011-05-13

Du har under experimentet spelat fyra omgångar. Märkte du någon skillnad mellan dessa?
Nej.

Om ja, vilken spelomgång tyckte du fungerade bäst respektive sämst?

Hur viktigt är det att ord som motståndare har spelat syns direkt?
Inte så viktigt. Ibland när man spelar mot varandra på tåget kan det vara lite segt men annars brukar det inte spela någon roll.

Vilken mobilenhet använde du under experimentet?
iPad.

Vilken webbläsare använde du under experimentet?
Safari.

Vilken mobiloperatör har du?
Ingen.

Under experimentet erhölls internetanslutning genom WIFI hotspot från HTC Desire med operatören Tele2.

Appendix G - Intervjuprotokoll - testkandidat 3

Kön: Man
Ålder: 20-25
Datum: 2011-05-13

Du har under experimentet spelat fyra omgångar. Märkte du någon skillnad mellan dessa?
Nej.

Om ja, vilken spelomgång tyckte du fungerade bäst respektive sämst?

Hur viktigt är det att ord som motståndare har spelat syns direkt?
Ganska viktigt, det beror på hur intensivt det är. Det kan bli segt om man väntar på att någon ska svara. Spelar man inte aktivt gör det ju inget eftersom telefonen ligger i fickan.

Vilken mobilenhet använde du under experimentet?
iPhone 3G.

Vilken webbläsare använde du under experimentet?
Safari.

Vilken mobiloperatör har du?
Telia.

Appendix H - Intervjuprotokoll - testkandidat 4

Kön: Man
Ålder: 20-25
Datum: 2011-05-13

Du har under experimentet spelat fyra omgångar. Märkte du någon skillnad mellan dessa?
Nej.

Om ja, vilken spelomgång tyckte du fungerade bäst respektive sämst?

Hur viktigt är det att ord som motståndare har spelat syns direkt?
Jag vet inte riktigt. Det spelar nog ingen större roll. Det blir ju att man väntar ändå.

Vilken mobilenhet använde du under experimentet?
iPad.

Vilken webbläsare använde du under experimentet?
Safari.

Vilken mobiloperatör har du?
Telenor.