Webes felületen játszható sakkjáték készítése

Rózsa Zoltán

2010

Forrai Magániskola Kéttannyelvű Középiskola

Nyilatkozat

Alulírott Rózsa Zoltán, a Forrai Magániskola Kéttannyelvű középiskola hallgatója kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és a szakdolgozatban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem

Aláírás

Tartalomjegyzék

I. Témakiírás

A szakdolgozat keretében fejlesszen ki egy jól játszható webes sakkjátékot.
Törekedjen rá, hogy alkalmazás figyelembe vegye a sakkjáték összes szabályát.
Nem cél, hogy az alkalmazás komoly játékportál, legyen, csak az a lényeg, hogy játszható legyen és kiindulási pont legyen egy ilyen irányú komolyabb fejlesztésnek.

Az alkalmazás felületén lehetőség legyen a regisztrációra és a bejelentkezésre, hogy a játékok folyamán az eredményeket nyilván lehessen tartani.
Bejelentkezés után a játékos vagy „nyit egy új asztalt”, beállítva, hogy milyen színnel szeretne játszani, vagy pedig „leül” egy már nyitott partihoz egy szabad helyre elfogadva ezzel a felkínált színt.

Játék közben a felhasználó egy sakktáblán követhesse vizuálisan az eseményeket. A táblán látja megjelenni az ellenfél lépését és neki is lehetősége van beírni a saját válaszlépését amely elküldés után szintén megjelenik a sakktáblán.
Az alkalmazás ismerje az összes lépésekre vonatkozó versenyszabályt, mint például a sáncolás, az en-passant stb. és ne engedje a játékosoknak szabálytalan lépés megtételét.
A játékosok bármelyike felajánlhatja a döntetlent egy gomb megnyomásával saját lépésének megtétele után és ha ezt az ellenfél elfogadja, akkor a játszma eredménye az állástól függetlenül döntetlen lesz.

Az alkalmazás a játszmákat tárolja el ezzel megteremtve a lehetőséget a későbbiekben létrehozandó szabadon böngészhető játszmaadatbázishoz.

II. Bevezetés

Napjainkban nagyon népszerűek a webes felületen, böngészőben játszható játékok. Számtalan oldalon találhatunk mindenféle kategóriában és stílusban unaloműzőnek szánt kis alkalmazásokat.
Nagy vonzereje ezeknek a kis webes játékoknak, hogy nem igényelnek hosszas előkészületeket, telepítést, nagyméretű fájlok letöltését.
Egyszerűen és gyorsan indíthatóak, azonnal játszhatóak és rövid időre kötik le a felhasználó figyelmét.

Hasonlatos ezekhez, de már egy magasabb színvonalat képező kategória, a weboldalon játszható társas, logikai, stratégiai játékok.
Ezek sem igényelnek, telepítést, de a felhasználónak tartósabb akár rendszeres kikapcsolódást nyújthatnak, köszönhetően az emberi ellenfeleknek és az összetettebb szabályoknak.
Példának hozható fel a nagyon népszerű Honfoglaló, a jojatek.hu és például létezik már a népszerű Catan telepesei című játék webes változata is.
Böngészőben, webes felületen játszható sakkjáték oldalak is léteznek már természetesen. A komolyabbak persze fizetősek, vagy korlátozott felhasználást nyújtanak csak ingyen, ilyenek például a playchess.com, vagy magyar nyelven a zebrachess.hu.

E szakdolgozat keretében szeretnék kifejleszteni egy ingyenes, de jól játszható webes sakkjátékot.
A cél az, hogy
Fontosnak tartom és törekedni fogok rá, hogy alkalmazásom figyelembe vegye a FIDE (**F**édération **I**nternationale **d**es **É**checs, a nemzetközi sakkszövetség) a sakkjátékra vonatkozó versenyszabályait, illetve igyekszem úgy tervezni a rendszert, hogy azok a szabályok, melyek bevezetése meghaladja e szakdolgozat keretét, könnyen beilleszthetők legyenek egy esetleges továbbfejlesztés alkalmával.

A rendszerrel szembeni elvárások nagy vonalakban az alábbiak.
Az alkalmazás felületén lehetőség legyen a regisztrációra és a bejelentkezésre, hogy a játékok folyamán az eredményeket nyilván lehessen tartani.
Bejelentkezés után a játékos vagy „nyit egy új asztalt”, beállítva, hogy milyen színnel szeretne játszani, vagy pedig „leül” egy már nyitott partihoz egy szabad helyre elfogadva ezzel a felkínált színt.
Játék közben a felhasználó egy sakktáblán követhesse vizuálisan az eseményeket. A táblán látja megjelenni az ellenfél lépését és neki is lehetősége van beírni a saját

válaszlépését amely elküldés után szintén megjelenik a sakktáblán.
A gondolkodási idő tekintetében nem cél, hogy igazi verseny körülményeket teremtsünk. Időt jelenleg nem kell mérnie a programnak viszont az esetleges továbbfejlesztésnél ez legyen könnyen megvalósítható.
Az alkalmazás ismerje az összes lépésekre vonatkozó versenyszabályt és ne engedje a játékosoknak szabálytalan lépés megtételét. A játékosok bármelyike felajánlhatja a döntetlent egy gomb megnyomásával saját lépésének megtétele után és ha ezt az ellenfél elfogadja, akkor a játszma eredménye az állástól függetlenül döntetlen lesz. Jelen szakdolgozat kereteit meghaladja, hogy a döntetlen igénylés szabályait is figyelembe vegye (háromszori tükörállás, 50 lépés ütés és gyaloglépés nélkül), ez szintén egy esetleges továbbfejlesztés alkalmával kerülhet megvalósításra.
A játékosoknak egy gomb megnyomásával lehetőségük legyen a kilátástalannak gondolt játszma feladására.
A játszmák eredményét tárolja le a rendszer de a pontszámítás egyenlőre nem megvalósítandó feladat.

Az alkalmazás a játszmákat eltárolja ezzel létrehozva egy szabadon böngészhető játszmaadatbázist, de a játszmák visszajátszásának megoldása illetve a játszmák kimentése PGN formátumba már túlhaladja jelen dolgozat kereteit.

A fentebb vázolt websakk alkalmazás PHP nyelven készül, figyelembe véve, hogy talán ehhez a nyelvhez találni a legszélesebb körű ingyenes támogatást, ingyenes tárhelyet a kipróbáláshoz, működtetéshez.
Az alkalmazás MySQL adatbázist használ, valamint a böngészőoldalon JavaScriptet, és Ajax technológiát.

III. Fejlesztői dokumentáció

1. Követelményelemzés

1.1. Részletes feladatspecifikáció

A feladat tehát egy webes felületen játszható sakkjáték készítése.
A játékot személyek játszák, tehát gépi intelligencia beépítése nem szükséges, a rendszer csak a játék kereteit adja, meg, a szabályok betartását ellenőrzi, a játékot vizuálisan megjeleníti, illetve játszmákat tárol.
Most röviden ismertetem a sakkjáték azon szabályait, melyeknek a betartására ügyel a program.

A sakkot két játékos játsza, világos illetve sötét színnel. A játékot világos kezdi és ezután felváltva lépnek. Lépni kötelező. Mindkét játékosnak 16-16 figurája van (1 király, 1 vezér, 2 bástya, 2 futó, 2 huszár, 8 gyalog), melyek különböző lépésmóddal rendelkeznek. A legfontosabb báb a király. A királyt nem lehet leütni, ha a királyt támadják (sakk) azt ki kell védeni.
Ha ezt nem lehet megtenni, akkor mondjuk, hogy az adott játékos mattot kapott, tulajdonképpen ez a játék végcélja.
Ha valamelyik fél lépésen van ugyan, de nem tud szabályosat lépni, és a királya sincs sakkban, akkor azt mondjuk, hogy patt. Patt esetén a játszma döntetlen eredménnyel ér véget.

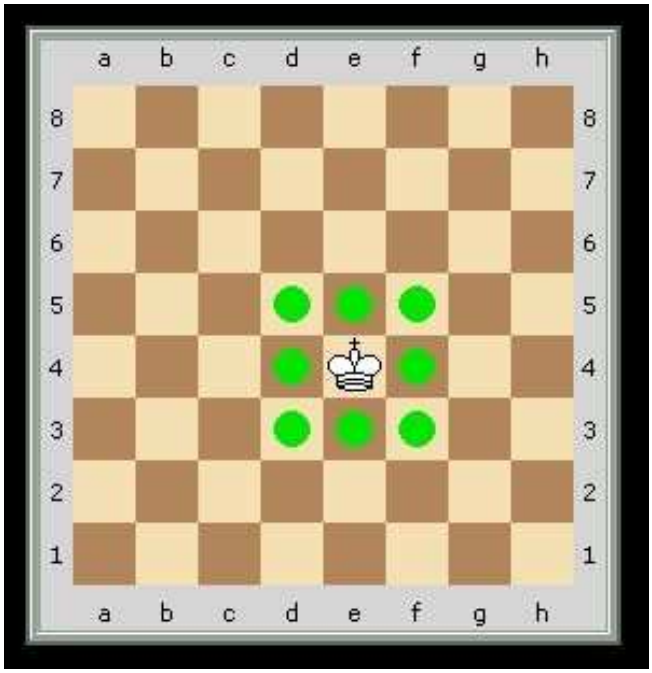
Az alapállásban a világos tisztek világos alapsorán, az 1. soron, a sötét tisztek pedig sötét alapsorán, a 8. soron helyezkednek el. A világos gyalogokat a 2. sorra, míg a sötéteket a 7. sorra állítjuk. A sakktáblát a a III.1.1.1. ábra alapján kell felállítani.

A lépések során valamelyik bábunkat az egyik mezőről egy másik mezőre helyezzük. Általános szabály az, hogy bábjainkkal olyan mezőre nem léphetünk, ahol saját bábunk áll. Ha olyan mezőre lépünk, ahol ellenséges báb tartózkodik, akkor azt leüthetjük. Saját báb nem üthető.

III.1.1.1 ábra a sakkjáték alapállása

A figurák lépésmódja:

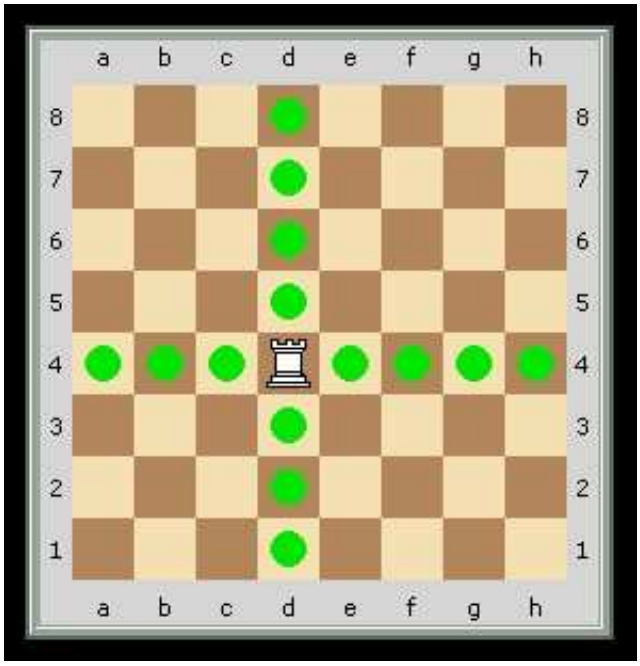
A király tetszés szerinti irányban a szomszédos mezőre léphet, feltéve, hogy azt ellenséges báb nem támadja. A király ütésbe, azaz sakkba nem léphet.



III.1.1.2. ábra a király lépésmódja

Az alábbi ábrán a világos király a d5, e5, f5, d4, f4, d3, e3, f3 mezőkre léphet.

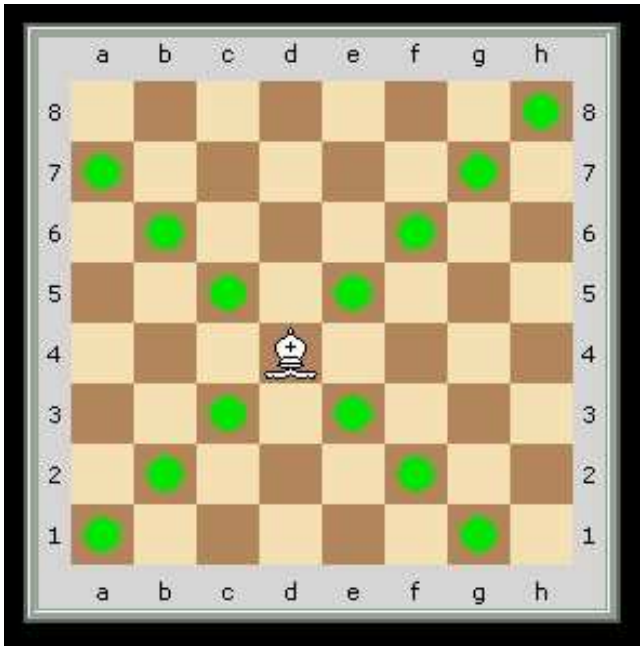
A bástya azon a soron, vagy azon a vonalon léphet, amelyiken áll, tetszőleges távolságra, de közbeeső saját, vagy ellenséges bábot nem ugorhat át



III.1.1.3. ábra a bástya lépésmódja

. Az ábrán a d4 bástya a következő mezőkre léphet: a4, b4, c4, e4, f4, g4, h4, d1, d2, d3, d5, d6, d7 és d8.

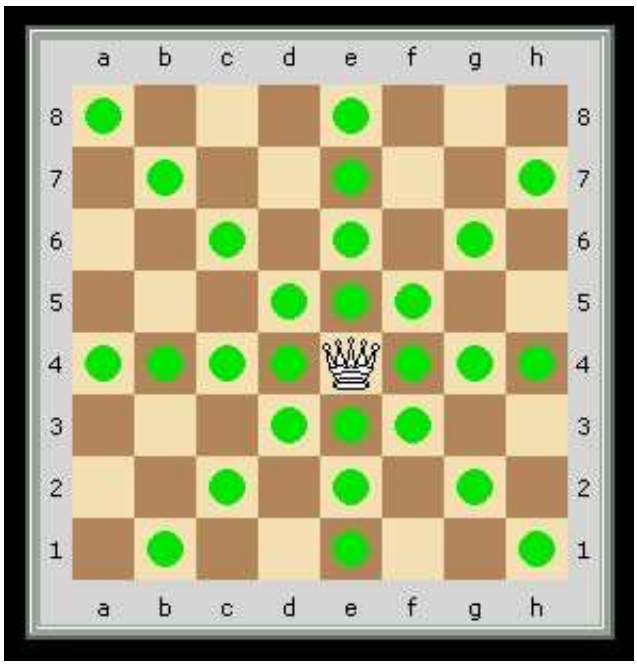
A futók átlós irányban mozoghatnak, a világos színen álló futó a világos színű átlókon, a sötét színen álló futó pedig a sötét színű átlókon. A futó a bástyához hasonlóan tetszés szerinti távolságra léphet, de közbeeső saját, vagy ellenséges figurát a futó sem ugorhat át.



III.1.1.4. ábra a futó lépésmódja

Az fentebbi ábrán a d4-en álló futó a következő mezőkre léphet: a1, b2, c3, e5, f6, g7, h8, a7, b6, c5, e3, f2 és g1.

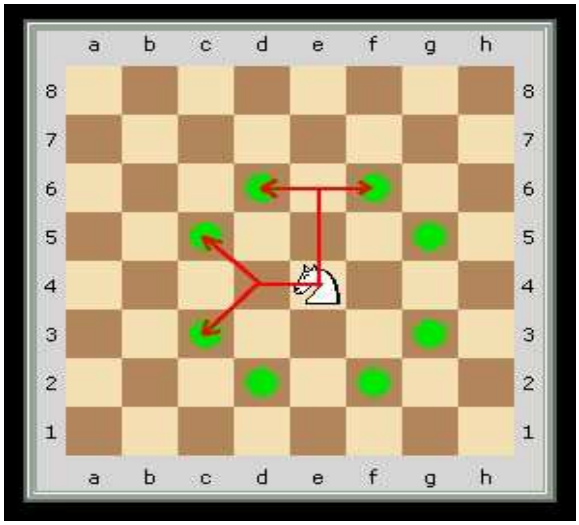
A vezér úgy lép, mint a bástya és a futó, azaz mozoghat a sorokon, a vonalakon, mint a bástya és átlós irányban, mint a futó. Azt, hogy a vezér éppen milyen színű átlón mozoghat az határozza meg, hogy éppen milyen színű mezőn áll.



III.1.1.5. ábra a vezér lépésmódja

A vezér a bástyához és a futóhoz hasonlóan tetszés szerinti távolságra léphet, mind a sorokon, vonalakon és az átlókon, de közbeeső saját, vagy ellenséges figurát nem ugorhat át. Az fenti ábrán az e4-en álló vezér a következő mezőkre léphet: a4, b4, c4, d4, f4, g4, h4, e1, e2, e3, e5, e6, e7, e8, b1, c2, d3, f5, g6, h7, a8, b7, c6, d5, f3, g2 és végül a h1.

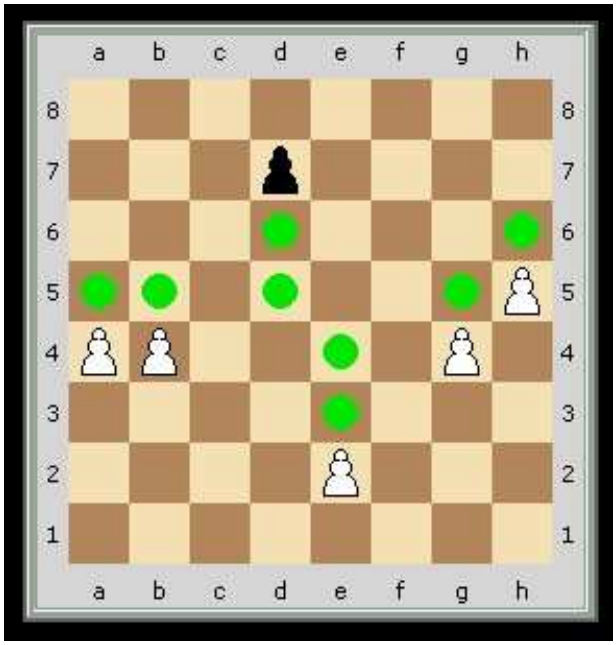
A huszár L alakban lép, vagyis kettővel előre, vagy hátra és eggyel oldalt, illetve két mezővel oldalt és eggyel előre vagy hátra.



III.1.1.6. ábra a huszár lépésmódja

A huszár sem léphet olyan mezőre, amelyiken saját báb áll.
A huszár a többi figurával ellentétben a közbeeső akadályokat, legyen az saját, vagy ellenséges báb, átugorhatja.
A huszár menetmódját szokták úgy is magyarázni, mintha egy királlyal kettőt lépnénk, egyet úgy, mint a bástya és ugyanabban az irányban (tehát nem visszafelé) még egyet, úgy, mint a futó.
A III.1.1.6. ábrán az e4 huszár a következő mezőkre léphet: c3, c5, d6, f6, g5, g3, f2 és d2.

A gyalogok alapvetően különböznek a többi bábtól. A gyalogok csak előre, az ellenfél hadállása irányában egyenesen léphetnek. A gyaloggal általában csak egy mezőt haladhatunk előre, kivéve azt az esetet, amikor a gyalog még az alapállásban áll (a világos gyalogok a 2., a sötétek pedig a 7. soron), ekkor a gyaloggal 2 mezőt is léphetünk.
Az alábbi ábrán vegyük sorra a gyalogok lépéslehetőségeit! Az eddig elmondottak alapján az a4-en álló világos gyalog csak a5-re, a b4-en álló pedig csak b5-re léphet. A d7-en álló sötét gyalog még alaphelyzetben van, ezért léphet mind a d6, mind a d5 mezőre. Ugyanez a helyzet az e2 világos gyaloggal, az alaphelyzetből e3-ra és e4-re is léphet. A g4-en álló világos gyalog csak g5-re, a h5-ön álló pedig csak h6-ra léphet. A gyalogok a közvetlenül előttük, átlós irányban, a szomszédos vonalon álló bábot üthetik ki. Tehát a gyalog egyenesen előre nem üthet.
Ha valamelyik gyalog a játszma során állandóan előre haladva eléri az ellenfél alapsorát (tehát a világos gyalog a 8. sort, a sötét gyalog az 1. sort) a belépés pillanatában át kell változtatni a következő tisztek valamelyikévé: vezér, bástya, futó vagy huszár. Amíg ez az átváltozás nem történt meg, addig a lépés nem tekinthető befejezettnek.

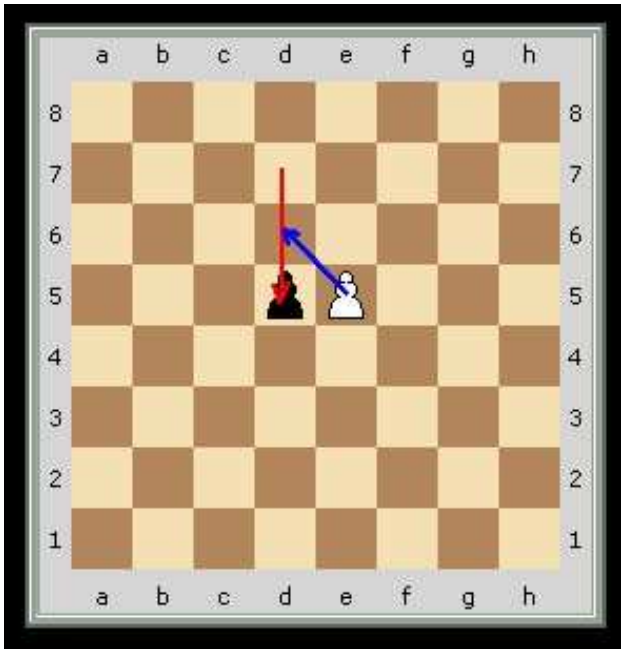


III.1.1.1.7 ábra a gyalogok lépésmódja

A játékos saját belátása szerint választhat a lehetőségek közül, attól függetlenül, hogy hány ilyen bábja van már a sakktáblán.

Az "en-passant" ütés:
Az alábbi állásban, ha a sötét d7 gyalogja d6-ra lépne, akkor azt világos e5 gyalogja

leüthetné. Ha a sötét gyalog d7-ről d5-re lép, akkor ugyan áthalad a világos gyalog ütőkörén, de d5-ön olyan mezőre kerül, ahol a világos gyalog már nem ütheti le. Az "en-passant" ütés szabálya azt mondja ki, hogy ebben az esetben a d5-re kerülő gyalogot "menet közben" ki lehet ütni, azaz úgy, mintha csak d6-ra lépett volna. Az ütési lehetőség hatálya kizárólag a következő lépés, ha ezt világos elmulasztja, többé már nem ütheti le "en-passant" ezt a gyalogot a parti során.



III.1.1.1.8 ábra az en-passant ütés

A sáncolás:
A bástya és a király közös lépése az úgynevezett sáncolás. A lépés során a játékos egyidejűleg két bábjával léphet, mégpedig a királyával és az egyik bástyájával egyszerre. Ezzel a lehetőséggel a parti során mindkét fél (játékos) egyszer-egyszer élhet. A sáncolásnak feltételei vannak. Aki a királyával már lépett, az a játszma során már nem sáncolhat. Ha valamelyik bástyával már léptünk, az a bástya már nem vehet részt sáncolásban. A sáncolás során a király nem haladhat át és nem léphet olyan mezőre, amelyet ellenséges báb támad. Az éppen sakkot kapott király sem sáncolhat, tehát a sakk sáncolással nem hárítható! A sáncolás során a király és a bástya között semmilyen figura nem állhat.
Az eredeti helyén álló király lép először két mezővel jobbra, vagy balra, majd az a bástyát, amelyik felé lépett a király, a királyt átugorva, a király mellé, a király másik oldalára helyezzük. Rövid sáncolás esetén a király a hozzá közelebb levő bástya felé, hosszú sáncolás esetén a király a tőle távolabb levő bástya felé lép.

Az alkalmazásnak a szabályok betartatása mellett bizonyos adminisztrációs teendői is vannak.
A legalapvetőbb a bejelentkezés, tehát a felhasználó azonosítása. Ehhez természetesen felhasználói név és jelszó adatokat kell tárolnunk. A név és jelszó pároson kívül még nyilvántartásra kerül egy e-mail cím is, ha esetleg később az alkalmazást ki kellene bővíteni egy elfelejtett jelszó kiküldése funkcióval. Egyéb adatokat a felhasználóról jelen esetben nem szükséges nyilvántartani.

Tárolni kell a felhasználói adatokon kívül természetesen a lejátszott játszmák adatait is,

úgymint:

- kik játszották a játszmát
- ki volt világossal és sötéttel
- mikor kezdődött a játszma
- mikor végződött a játszma
- mi lett a játszma eredménye
- a játszma kezdetekor a játékosok mennyi ponttal rendelkeztek (továbbfejlesztés céljára, jelenleg nem használt)
- továbbfejlesztés céljára a játszma lépésenkénti időtartama

Természetesen tárolni kell a játszma összes lépését, a lépés megtételének idejével együtt, hogy az alkalmazást továbbfejleszthessük a visszajátszás menüponttal, illetve a játszmák exportálása menüponttal. A valamilyen oknál fogva félbeszakadt játszmákat (pl. mindkét felhasználó feladás nélkül kilépett, vagy megszakadt internetkapcsolat esetén) a rendszer törölje.

Ügyelni kell arra is, hogy egy felhasználónévvel egyszerre csak egy partit lehessen játszani, illetve egy felhasználónévvel egyszerre csak egy kihívást lehessen tenni. Értelemszerűen kizárandó az is, hogy világos és sötét ugyanaz a felhasználónév legyen.

1.2. Használati esetek (Use case diagram)

Mivel a feladat bonyolultságát jelen esetben nem a funkciók nagy száma és a szerteágazó lehetőségek adják, hanem a viszonylag bonyolult szabályrendszer betartatása és az ellenfelek közötti kommunikáció megteremtése, ezért az ábrából is látható, hogy a feladat viszonylag kevés használati esettel leírható és ezek a használati esetek sem túlságosan összetettek logikailag.
A rendszert használóknak egyetlen típusa van (csak egy aktor látható a diagramon): a játékos.
Ez abból következik, hogy a rendszer jelenlegi formájában nem igényel adminisztrátori közreműködést.
A rendszer használati esetek diagramja a III.1.2.1. ábrán látható.

Látható, hogy minden egyes használati esetet, meg kell hogy előzzön a **Bejelentkezés** használati eset, mivel az alkalmazás minden egyes funkcióját csak regisztrált, bejelentkezett felhasználó veheti igénybe. Az ábrából is látható, hogy ennek megfelelően az összes használati eset közvetve vagy közvetlenül „include” kapcsolatban áll a **Bejelentkezés** használati esettel.

Lássuk tehát a használati esetek részletes leírását

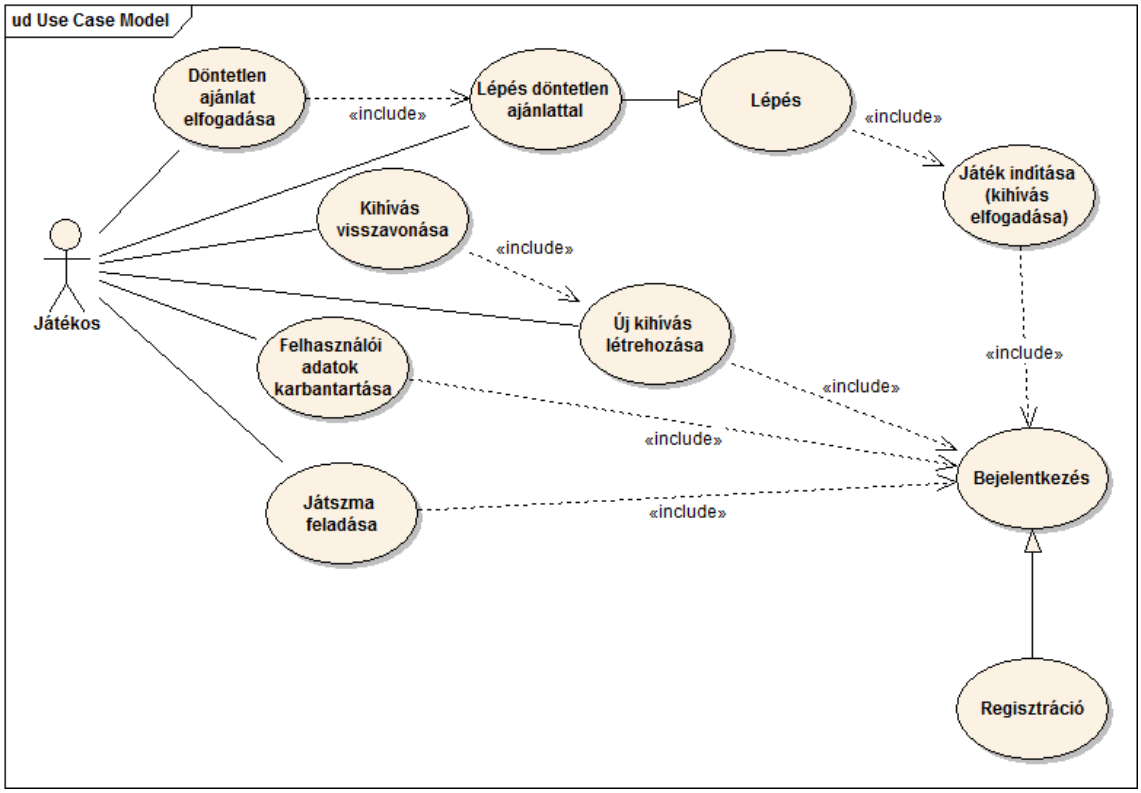
- **Bejelentkezés**

Leírás:
Lehetőséget ad a regisztrált felhasználóknak, hogy felhasználó nevüket és jelszavukat megadva belépjenek a rendszerbe, hogy használhassák a rendszer szolgáltatásait.
A használati eset keretében kezelni kell, a sikertelen belépést esetlegesen rosszul megadott név és jelszó esetén.

Aktor: Játékos

Előfeltétel: Hogy a bejelentkezés sikeres legyen léteznie kell egy érvényes felhasználói név jelszó bejegyzésnek az alkalmazás adatbázisában. Tehát egy sikeres regisztrációnak

meg kellett előtte történnie.



III.1.2.1. ábra az alkalmazás felhasználói esetek diagramja

Utófeltétel: A használati eset sikeres befejezése esetén a felhasználó hozzáfér az alkalmazás funkcióihoz.

Forgatókönyvek:
1. A Játékos a böngészőjében megnyitja az alkalmazás kezdő oldalát. A megjelenő lapon kitölti a felhasználó név beviteli mezőt és a jelszó beviteli mezőt. Ezután megnyomja a bejelentkezés gombot. Ekkor a rendszer ellenőrzi a megadott adatok helyességét. Ha az adatok helyesek akkor a Játékos hozzáfér a rendszer funkcióihoz és minden tevékenysége a megadott felhasználónéven lesz bejegyezve.
2. A kezdő oldal megnyitása és az adatok megadása ugyanúgy történik mint az előző esetben. Viszont ha a rendszer nem találja helyesnek a nevet vagy a jelszót, akkor természetesen nem engedi a hozzáférést, megjelenít egy oldalt amin értesíti a felhasználót a sikertelenségről és egy linket ad a bejelentkező oldalhoz való visszatérésre, ahol újra megkísérelheti a bejelentkezést.

- Regisztráció

Leírás:
A leendő felhasználók itt tudnak létrehozni maguknak felhasználónevet, amellyel igénybe tudják venni az alkalmazás szolgáltatását.

Aktor: Játékos

Előfeltétel: Nincs

Utófeltétel: A tevékenység sikeres lezárása után az alkalmazás adatbázisába szerepelni fognak az új felhasználó adatai.

Forgatókönyvek:
1. A játékos böngészőjében megnyitja az alkalmazás kezdő oldalát. A megjelenő adatlapon kitölti a regisztrációhoz szükséges adatokat (felhasználónév, jelszó kétszer a téves bevitel megakadályozására, és a felhasználó e-mail címe) és ezután megnyomja regisztráció gombot.
Ennek hatására letárolásra kerül az új regisztráció. A sikeres végrehajtás esetén a felhasználót a rendszer azonnal belépteti az újonnan létrejött hozzáférésével, tehát lezajlik egy **Bejelentkezés** használati eset is, a fentebb leírtak szerint.
2. Az előző forgatókönyvhöz hasonlóan megtörténik az adatok megadása, ám valamely okból ez nem felel meg a feltételeknek (pl. már létező felhasználónév, vagy a jelszót nem sikerült kétszer azonos módon megadni). Ekkor a program értesíti a felhasználót a sikertelenség tényéről egy hibajelentő oldalon és ad lehetőséget a kezdő oldalhoz való visszalépésre, a regisztráció ismételt megkísérlésére.

- Új kihívás létrehozása

Leírás:
A bejelentkezett felhasználó szeretne egy játszmát játszani, ezt a szándékát tudja jelezni ezzel a használati esettel. A Játékos egy kihívást hoz létre amelyet a többi bejelentkezett játékos észlel, mert az alkalmazás megjeleníti ezt a képernyőjükön. Ha valakinek megfelel a kihívás (a játékos személye illetve a színelosztás) és szintén játszani akar, akkor elfogadhatja a kihívást és ezzel új játszma indulhat, de ez már egy másik használati eset.

Aktor: Játékos

Előfeltétel: Előfeltétel csupán annyi, hogy a játékos be legyen jelentkezve.

Utófeltétel: A használati eset sikeres lezárása után a Játékos játszma indítási szándékát a többi bejelentkezett felhasználó felé jelezte és utána várakozik a kihívás elfogadására.

Forgatókönyvek:
1. A bejelentkezett felhasználó az alkalmazás fő oldalán megnyomja a két új kihívás indító gomb közül azt amely azt jelzi, hogy az adott Játékos világossal szeretne játszani. Ekkor az alkalmazás közread egy új kihívás kérelmet, amelyet a többi felhasználó észlelhet, és a kihívó játékosnak egy új ablakban megjelenít egy sakkjátszma alapállást és várakoztatja amíg ellenfél nem akad.
2. Ez a forgatókönyv csak annyiban tér el az előzőtől, hogy a Játékos azt a gombot nyomja meg amellyel azt jelzi, hogy sötéttel szeretne játszani. A program reagálása erre hasonló az előzőre, természetesen jelzi a többiek felé a sötéttel való játék szándékát. Egyébként szintén megjeleníti az alapállást, természetesen a megfelelő oldalra fordítva a sakktáblát. Az előző forgatókönyvhöz hasonlóan várakoztatja a Játékost, azzal a különbséggel, hogy a kihívás elfogadása esetén várakoztatja még az ellenfél lépésére is.
3. A bejelentkezett felhasználó az előzőek szerint valamely gombnyomással kihívást indít. A rendszer azonban megtagadja ezt, pl. egy felugró ablakban értesítve a hiba tényéről. Ezt az okozhatja, hogy az illető játékos már intézett egy kihívást, esetleg már játékban is van.

- Kihívás visszavonása

Leírás:
Ez a tevékenység lehetőséget ad a Játékosnak arra, hogy egy korábban megtett kihívást, amelyet még senki sem fogadott el, (tehát nem indult játék) visszavonjon. Például azért,

mert ki akar lépni, vagy, későbbi időpontban, esetleg más színnel szeretne játszani.

Aktor: Játékos

Előfeltétel: A bejelentkezett játékosnak legyen egy érvényben lévő kihívása.

Utófeltétel: A felhasználónak nem lesz érvényben kihívása, az erre vonatkozó bejegyzések törlődnek, ezt a többi felhasználó is észlelni fogja.

Forgatókönyvek:
1. A kihívást létrehozó játékosnak a kérelem indításakor megjelent egy új ablakban egy játszma alapállás attól (világossal vagy sötéttel) és várakoztatva van ellenfél jelentkezésére. Ekkor ha még az ellenfél érkezése előtt (tehát még a játszma indulása előtt) becsukja ezt az ablakot akkor a kihívás visszavonásra kerül, az ezzel kapcsolatos bejegyzések törlődnek és a Játékos új kihívást tehet vagy elhagyhatja az alkalmazást.

- Játék indítása (kihívás elfogadása)

Leírás:
A felhasználó, várakozik az alkalmazás fő oldalán, esetleg átnézi a már ott megjelenő élő kihívásokat, és ha számára megfelelő egy kihívás (az ellenfél személye és a színelosztás szempontjából) akkor a kihívás linkjére kattintva elfogadhatja a felkérést és egy új játék indul.

Aktor: Játékos

Előfeltétel: A bejelentkezett játékosnak ne legyen élő kihívása.

Utófeltétel:Egy új játék indul az alkalmazás keretei között.

Forgatókönyvek:
1. A Játékos az alkalmazás főoldalán egy listában látja, felkéréseket, ebben a listában látja, hogy mely kihívásban lehetne fehérrel, és melyikben feketével játszani, illetve, hogy ki az ellenfél. Ha az adott játékosnak éppen nincs folyamatban játéka, vagy kihívása akkor ezek közül választhat, a megfelelő linkre kattintva. Választva egy játszmát amelyhez fehérrel csatlakozhat, ezzel elfogadva a kihívást. Ekkor megjelenik egy ablak ahol megjelenik egy sakkjátszma alapállás a fehér figurákkal a képernyő alja felé és a sötét figurák feletti részen megjelenik az ellenfél felhasználói neve (nickname, becenév). Az ellenfél már régebben kinyitott ablakában ekkor megjelenik a tábla felső része fölött (az ellenfélnek a fehér színű figurák vannak a képernyő felső részén) a kihívást elfogadó felhasználó neve. A kihívást elfogadó Játékosnak ekkor lehetősége van lépni, ellenfele tovább vár a lépésére, de már egyik játékos sem vonhatja vissza a játszma indítását.
2. Az előző forgatókönyvhöz hasonlóan a Játékos kiválasztja a neki megfelelő kihívást ezúttal sötét színt választ. Megnyílik számára az új játszma ablaka, az alapállással az előzőekben részletezett módon. Most viszont a kihívást elfogadó játékost várakoztatja a program az ellenfél lépésére és a kihívást létrehozónak van lehetősége lépését megtenni.
3. Az előzőekhez hasonlóan a bejelentkezett felhasználó választ a felkérések közül. Kiválasztva egy felkérést (ebben az esetben most lényegtelen milyen színnel) az alkalmazás megkísérel indítani egy új játszmát, de ez sikertelen. Ennek több oka lehet. Például az adott felkérést már időközben elfogadták és a játszma már elindult, illetve az adott kihívást már visszavonták és ezért nem tud új játszma indulni. A sikertelen műveletről az alkalmazás egy felugró üzenetben értesíti a Játékost.

Lépés

Leírás: Az egyik legfontosabb használati eset, az alkalmazás „lelke”. Folyamatban lévő játszmában a soron következő játékos megteszi és elküldi lépését. Erre majd válaszul hasonlóképpen az ellenfél is lép és ez a tevékenység ismétlődik folyamatosan a játék végéig

Aktor: Játékos

Előfeltétel: Egy játszmában szereplő Játékos lépésen következzen.

Utófeltétel: Az adott lépés el lett küldve a lépett Játékos várakozik az ellenfél lépésére.

Forgatókönyvek:
1. A folyamatban lévő játékban az éppen soron következő játékos megfontolja lépését és mikor elhatározásra jut, akkor a rendelkezésre álló két beviteli mezőbe beírja a lépést. Az első beviteli helyre a sakktábla azon mezőjének koordinátáit (előre a vízszintes utána a függőleges koordinátát pl.: e2) amelyen az a figura áll amelyikkel lépni akar, a második beviteli helyre pedig annak a mezőnek a koordinátáit ahová lépni akar. Abban a speciális esetben ha a soron következő Játékos gyalogja eléri az átváltozási mezőt (az ellenfél alapsorát) akkor a második beviteli mezőbe három karakter kerül, a harmadik karakter pedig a felvenni kívánt tiszt nevének kezdőbetűje (pl.: e8v azt jelenti, hogy a gyalog az e8 mezőn vezérré változik). A lépés beírása után a felhasználó megnyomja a Lépés elküldése gombot, melynek hatására a rendszer ellenőrzi a lépést, hogy megfelel-e az összes szabálynak. Ha igen akkor letárolja a lépést, értesíti róla az ellenfelet. A lépő játékost a program várakoztatja és a kezelő gombokat nem engedélyezett állapotba hozza, ezzel is jelezve, hogy várnia kell.
2. Az előző forgatókönyvhöz teljesen hasonlóan a soron következő Játékos beírja lépését és a megfelelő gomb megnyomásával elküldi. A rendszer ellenőrzi, hogy az adott állásban (figyelembe véve a korábbi lépéseket lásd sáncolás illetve en-passant) a lépés szabályos-e. De tegyük fel a lépés nem felel meg valamennyi szabálynak. Ekkor a rendszer visszautasítja a lépést. Egy felugró ablakban tájékoztatja a hiba tényéről a lépést megkísérlő játékost és a lépést természetesen nem tárolja le és nem is értesíti róla az ellenfelet. A Játékos ismét megpróbálkozhat egy lépés megtételével.

- Lépés döntetlen ajánlattal

Leírás: Az előző (Lépés) használati esetből általánosítással (generalize) származtatható ez a használati eset. Szerepe teljesen hasonló az előzőhöz. A specialitása csak annyi, hogy a Játékos az állást értékelve arra a következtetésre jutott, hogy neki kedvező eredmény lenne a döntetlen és ezért ezt felkínálja az ellenfélnek, aki erről értesülve elfogadhatja azt.

Aktor: Játékos

Előfeltétel: Egy játszmában szereplő Játékos lépésen következzen.

Utófeltétel: Az adott lépés el lett küldve a döntetlen ajánlattal együtt. A lépést végző Játékos várakozik az ellenfél lépésére.

Forgatókönyvek:
1. A folyamatban lévő játékban az éppen soron következő játékos megfontolja lépését és mikor elhatározásra jut, akkor a rendelkezésre álló két beviteli mezőbe beírja a lépést. Az első beviteli helyre a sakktábla azon mezőjének koordinátáit (előre a vízszintes utána

a függőleges koordinátát pl.: e2) amelyen az a figura áll amelyikkel lépni akar, a második beviteli helyre pedig annak a mezőnek a koordinátáit ahová lépni akar. Abban a speciális esetben ha a soron következő Játékos gyalogja eléri az átváltozási mezőt (az ellenfél alapsorát) akkor a második beviteli mezőbe három karakter kerül, a harmadik karakter pedig a felvenni kívánt tiszt nevének kezdőbetűje (pl.: e8v azt jelenti, hogy a gyalog az e8 mezőn vezérré változik). A lépés beírása után a felhasználó megnyomja a Lépés elküldése gombot, melynek hatására a rendszer ellenőrzi a lépést, hogy megfelel-e az összes szabálynak. Ha igen akkor letárolja a lépést, értesíti róla az ellenfelet. A lépő játékost a program várakoztatja és a kezelő gombokat nem engedélyezett állapotba hozza, ezzel is jelezve, hogy várnia kell.
2. Az előző forgatókönyvhöz teljesen hasonlóan a soron következő Játékos beírja lépését és a megfelelő gomb megnyomásával elküldi. A rendszer ellenőrzi, hogy az adott állásban (figyelembe véve a korábbi lépéseket lásd sáncolás illetve en-passant) a lépés szabályos-e. De tegyük fel a lépés nem felel meg valamennyi szabálynak. Ekkor a rendszer visszautasítja a lépést. Egy felugró ablakban tájékoztatja a hiba tényéről a lépést megkísérlő játékost és a lépést természetesen nem tárolja le és nem is értesíti róla az ellenfelet. A Játékos ismét megpróbálkozhat egy lépés megtételével.

Döntetlen ajánlat elfogadása

Leírás: A lépésen következő játékos észleli, hogy az ellenfele a lépése megtételekor döntetlent ajánlott. Mérlegeli az állást és dönt az ajánlat elfogadásáról. Az ajánlat mindaddig érvényben van amíg nem válaszolt a döntetlen ajánlattal egy időben elküldött lépésre. Az ajánlat elfogadása után a játszma döntetlen eredménnyel befejezésre kerül.

Aktor: Játékos

Előfeltétel: A Játékos egy folyamatban lévő játszmában lépésen következzen és az ellenfél az előző lépésekor döntetlen ajánlatot küldjön.

Utófeltétel: A játszma döntetlen eredménnyel befejeződött.

Forgatókönyvek
1. A Játékos egy folyamatban lévő játékban észleli az ellenfél lépését és mivel a Döntetlen ajánlat elfogadása gomb engedélyezetté válik ezáltal észleli az ellenfél döntetlen ajánlatát. Ha az ajánlatot nem akarja elfogadni, akkor nincs más teendője, mint a Lépés használati esetet alkalmazva nem vesz tudomást az ajánlatról és akkor minden a Lépés használati eset szerint folyik tovább. Viszont ha elfogadja akkor csak meg kell nyomnia az elfogadást jelképező gombot és ezáltal az alkalmazás elkönyveli a játszmát döntetlenként és a játszma ablakban mindkét Játékosnak kiírja az eredményt.

Játszma feladása

Leírás: Ha a Játékos kilátástalannak ítéli az állását akkor lehetősége van feladni a partit, ha ő következik lépésen. Ekkor a játszma az ellenfél győzelmével végződik.

Aktor: Játékos

Előfeltétel: A Játékos egy folyamatban lévő parti egyik szereplője legyen és ő következzen lépésen.

Utófeltétel: A játszma az ellenfél győzelmével elkönyvelésre kerül

Forgatókönyvek:

1. Egy folyamatban lévő partiban a lépésen következő Játékos úgy ítéli meg az állását, hogy ideje feladni a partit. Ekkor megnyomja a Játszma feladása gombot. Ekkor a rendszer elkönyveli a játszma befejezését az ellenfél győzelmével és erről tájékoztatja egy felirattal mindkét felet.

Felhasználói adatok karbantartása

Leírás: Bejelentkezett felhasználó meg tudja tekinteni a regisztrációkor megadott adatait és a felhasználói név kivételéve módosíthatja azokat.

Aktor: Játékos

Előfeltétel: A felhasználó legyen bejelentkezve a rendszerbe.

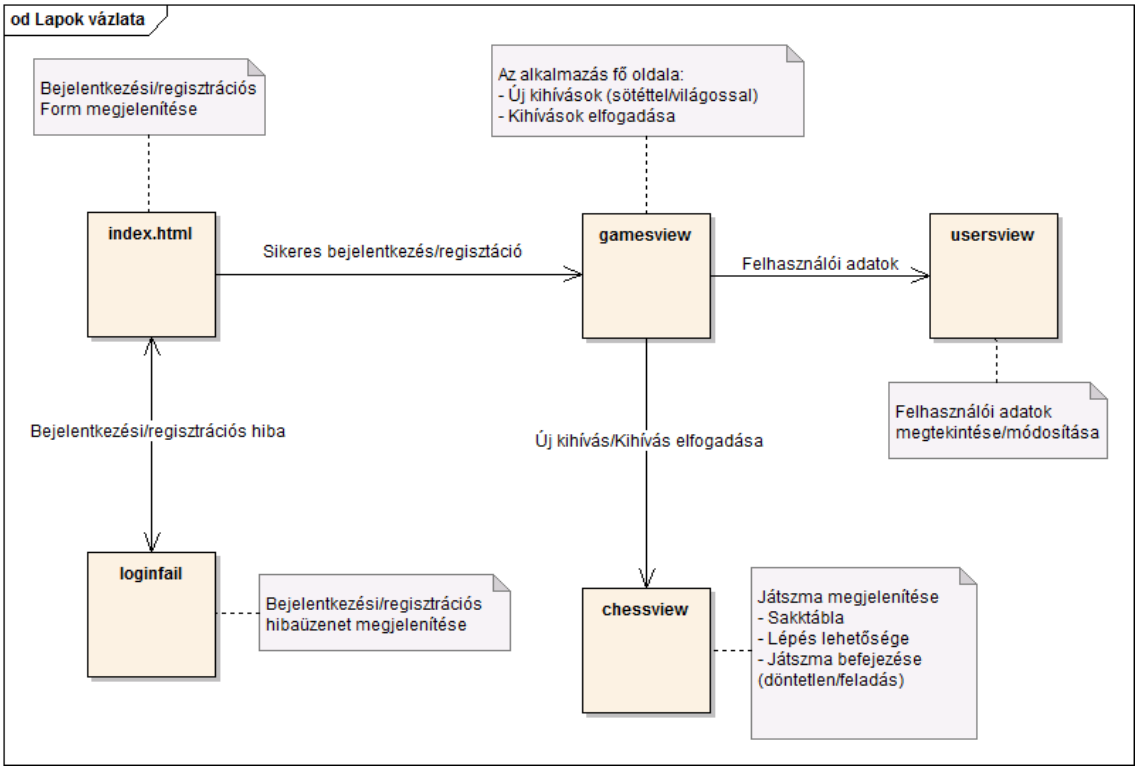
Utófeltétel: Az alkalmazás érvényesítette az esetleges változtatásokat. A felhasználói adatokban.

Forgatókönyvek:
1. A bejelentkezett felhasználó az alkalmazás fő oldalán megnyomja a Felhasználói adatok karbantartása gombot.
Ennek hatására megnyílik egy új ablak amelyben a Játékos a regisztrációkor megadott adatait láthatja.
Ezeket módosíthatja és a mentés gomb segítségével letárolhatja az esetleges változásokat, vagy pedig mentés nélkül az ablak bezárásával kiléphet.
2. Hasonlóan az előző forgatókönyvhöz. A felhasználó módosítja az adatokat, de a mentés gomb megnyomásakor a programnak nem sikerül menteni (pl. a jelszó a két mezőben különbözik). Erről a program egy felugró ablakban tájékoztatja a felhasználót, és ezután a felhasználó újra próbálkozhat, vagy bezárhatja az ablakot mentés nélkül.

1.3. Navigációs vázlat

Az III.1.3.1. ábrán azt láthatjuk, hogy az alkalmazás egyes oldalai milyen kapcsolatban állnak egymással. Az előző fejezetben részletezett felhasználói esetek alapján hogyan tudja a Játékos használni az alkalmazás funkcióit. Az **index.html** az alkalmazás kezdőlapja. Ezen a lapon van lehetőség a bejelentkezésre is és a regisztrációra is. Sikertelen bejelentkezés/regisztráció esetén betöltődik a hiba tényét jelző oldal (**loginfail**) ahonnan egy link segítségével visszatérhet a felhasználó a kezdő oldalra.Sikeres belépés esetén kerül a Játékos az ábrán is láthatóan az alkalmazás fő oldalára.

Gamesview oldal:
Itt három menüpont közül választhat illetve linkek segítségével elfogadhat egy már élő kihívást. Amíg a felhasználó használja a rendszert ez az oldal folyamatosan frissíti a rajta szereplő kihívások táblázatát (5 másodpercenként, AJAX technológia segítségével). Innen a Játékos vagy egy játszmát megjelenítő lapra kerül, mert elfogadott egy kihívást, vagy egy kihívását elfogadta egy másik játékos, vagy megnézheti a saját felhasználói adatait. Mindkét lehetőség esetén egy új ablak nyílik JavaScript segítségével, melyekben le van tiltva mindenféle navigációs lehetőség, csak az ablakokon lévő gombok segítségével lehet vezérelni a programot, illetve be lehet csukni az ablakot. Az alkalmazás fő oldala azonban továbbra is nyitva marad, de ha indítottunk már egy játszmát az előbb említett külön ablakban új kihívást már nem tehetünk, nem fogadhatunk el addig, amíg a már megkezdett játszmát be nem fejeztük.



III.1.3.1 ábra Az alkalmazás navigációs vázlata

A **chessview** lapon kizárólag csak az aktuális játszma állása látható és a játszmával kapcsolatos teendőknek megfelelő vezérlő elemek.
A **usersview** oldalon megtekinthetjük vagy módosíthatjuk adatainkat. Természetesen mindenki csak a saját adatait kérheti le.

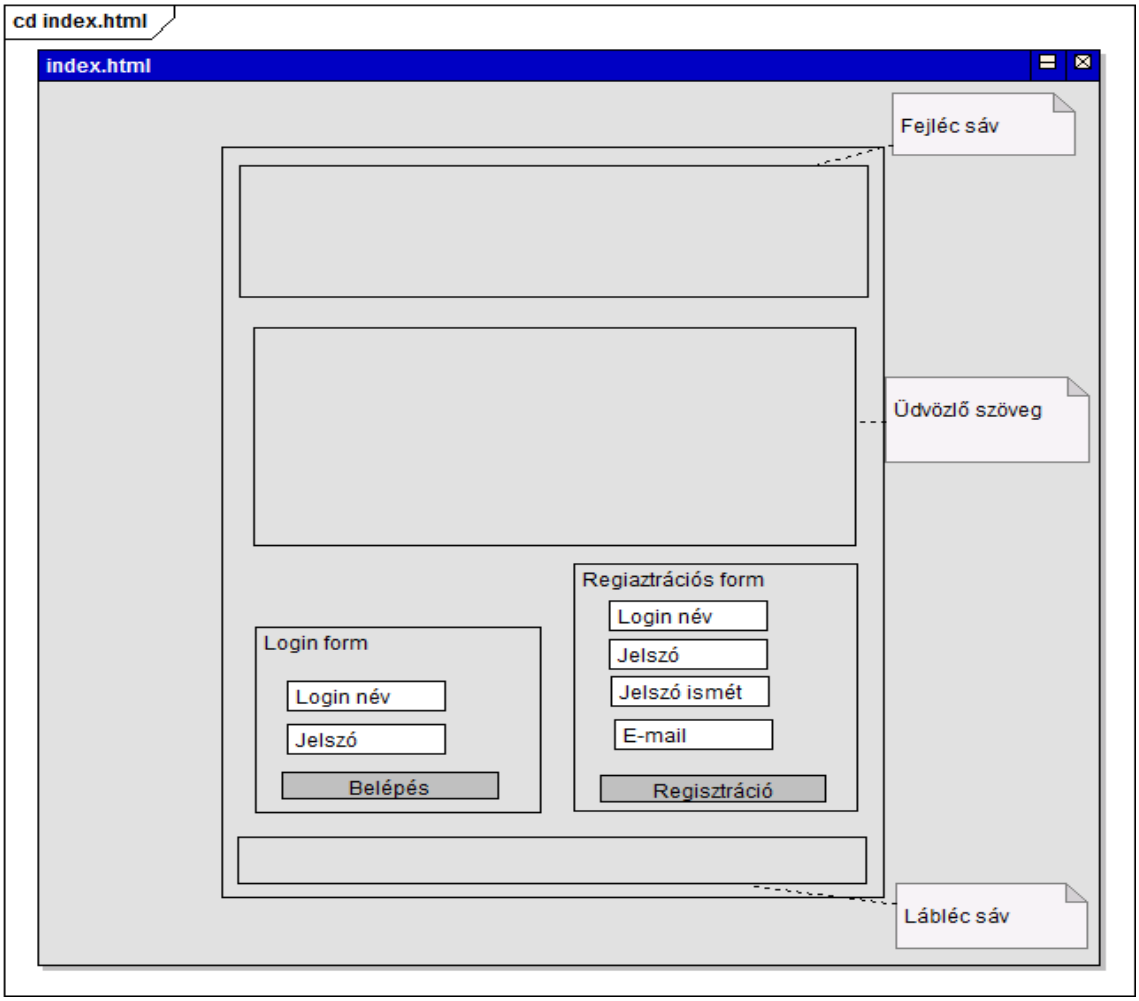
Ezután lássuk tehát a konkrét képernyőterveket a fentebb részletezett oldalakról.

1.4.Képernyőtervek

Az alkalmazásnak igyekeztem egységes arculatot adni. Minden oldal egy közös stíluslapot használ, ezáltal az esetleges design változásokat könnyebb végrehajtani. Minden oldalon azonos a háttérkép, a fejléc és a lábléc kinézete. Minden oldalon a barna különböző visszafogott árnyalatai uralkodnak. A háttérkép mindenütt egy fa sakktábla.

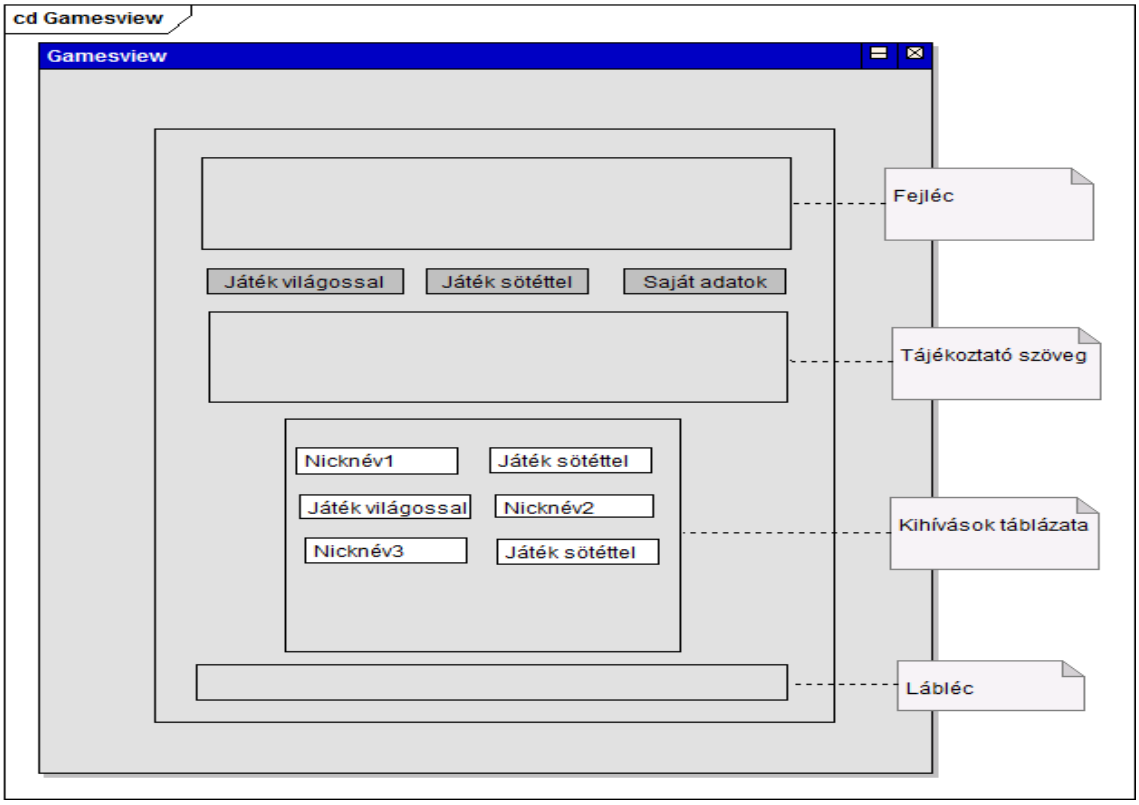
- index.html

Az alábbi ábrán a kezdőoldal látványterve látható



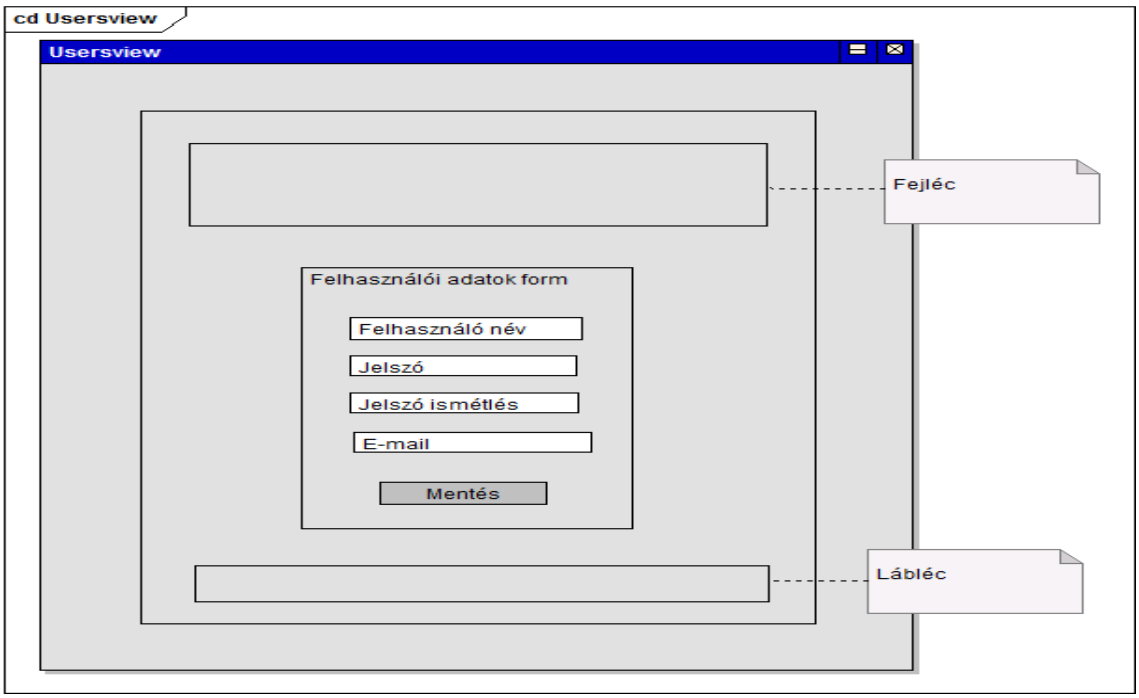
III.1.4.1. ábra Az index.html oldal terve

- Gamesview



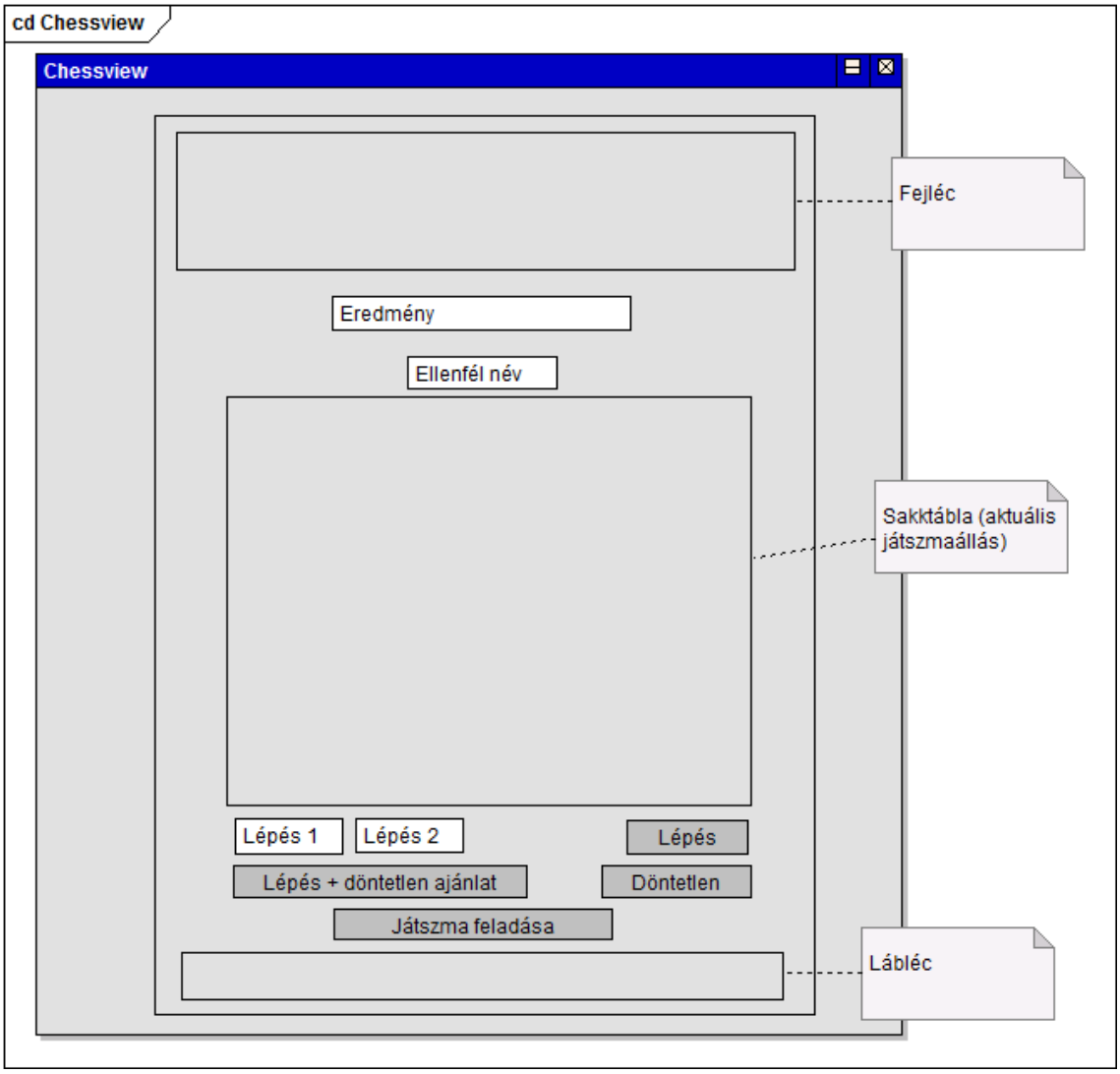
III.1.4.2. ábra a Gamesview oldal látványterve

Usersview



III.1.4.3. ábra a Usersview oldal látványterve

Chessview



III.1.4.4. ábra a Chessview oldal látványterve

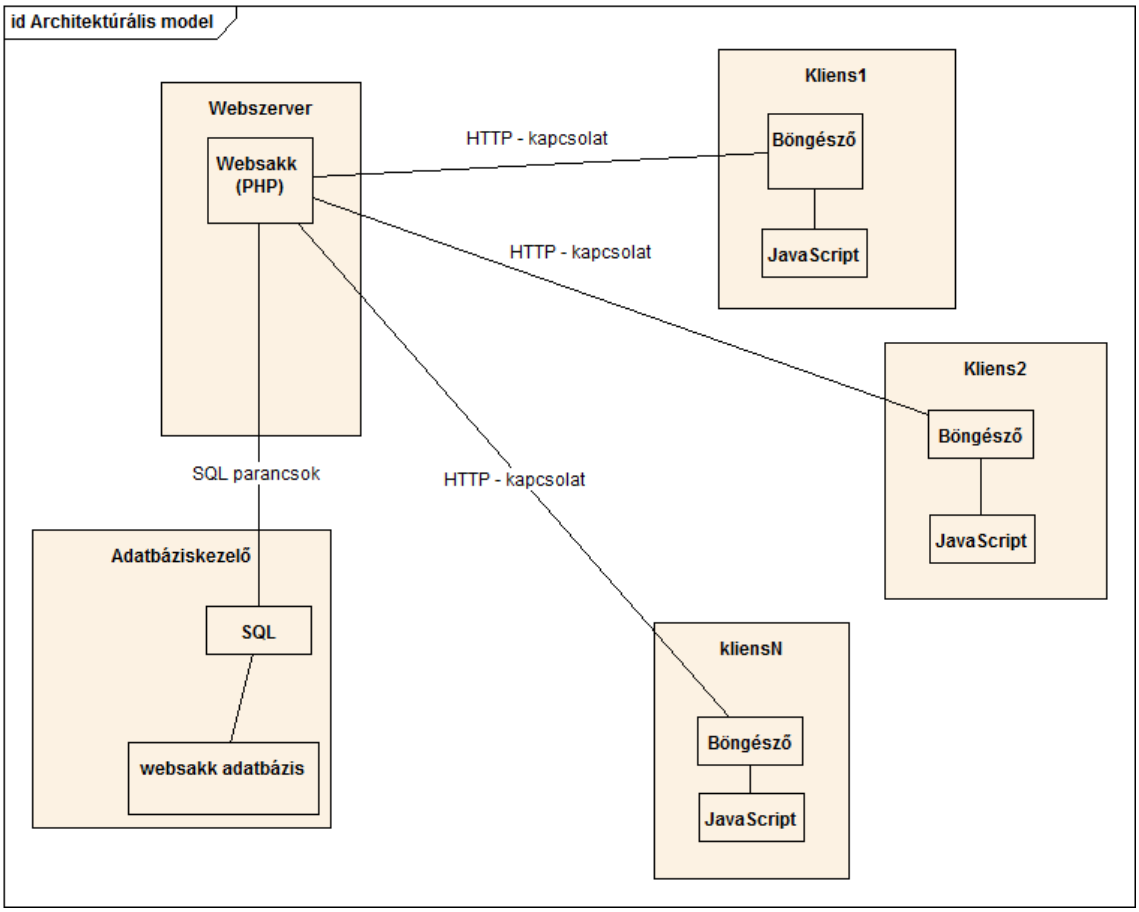
Ezen az oldalon lehet magát a sakkjátszmát lejátszani az alkalmazásban. Az ablakról a navigációs lehetőségek (pl. Vissza, cím beírás) hiányoznak, csak becsukni lehet. Ha a játszma vége előtt szeretnénk becsukni akkor az egyenlő a játszma feladásával.
Az oldal stílusa teljesen hasonló az alkalmazásunk előzőekben látott oldalaihoz. A háttér, fejléc, lábléc ugyanaz. A fejléc alatti mezőben lesz majd a játszma végén kijelezve az eredmény. Az eredmény megjelenése értesíti a játékost arról, hogy pl. az ellenfél feladta a játékot, vagy elfogadta esetleges döntetlen ajánlatunkat.
Alatta látható az aktuális ellenfél felhasználói neve, ha már elkezdődött a játszma. Ezután a sakktábla következik a mindenkori aktuális állást mutatva, a tábla alsó részén vannak a saját figurák.
A sakktábla alatt helyezkednek el a vezérlő elemek. A „Lépés1”, „Lépés2” beviteli mezőkben lehet a lépésünk koordinátáit megadni (Lépés1=a lépés kiindulópontja, Lépés2 a lépés végpontja, és esetleg ha gyalogátváltozásos lépésről van szó, akkor a Lépés2 3. karaktere a felvett figura kódja Vezér=v, Bástya=b, Futó=f, Huszár =h).

Ha mi következünk lépésen és sikeresen beírtuk a lépés koordinátáit, akkor a „Lépés”, vagy „Lépés + döntetlen ajánlat” feliratú gombbal lehet elküldeni a lépést, attól függően, hogy akarunk-e döntetlent ajánlani. Ha kilátástalannak ítéljük az állást, akkor a játszma feladása gombot kell megnyomni. Ha mi vagyunk lépésen és a „Döntetlen” feliratú gomb engedélyezve van, az azt jelenti, hogy ellenfelünk döntetlent ajánlott. Ha elfogadjuk akkor nyomjuk meg a „Döntetlen” feliratú gombot és a játék befejeződik.

2. Tervezés

2.1. Architektúrális modell

A fejlesztendő alkalmazást webes jellegére tekintettel **kliens-szerver architektúrájú** modellel jellemezhetjük.



III.2.1.1. ábra a rendszer architektúrális modellje

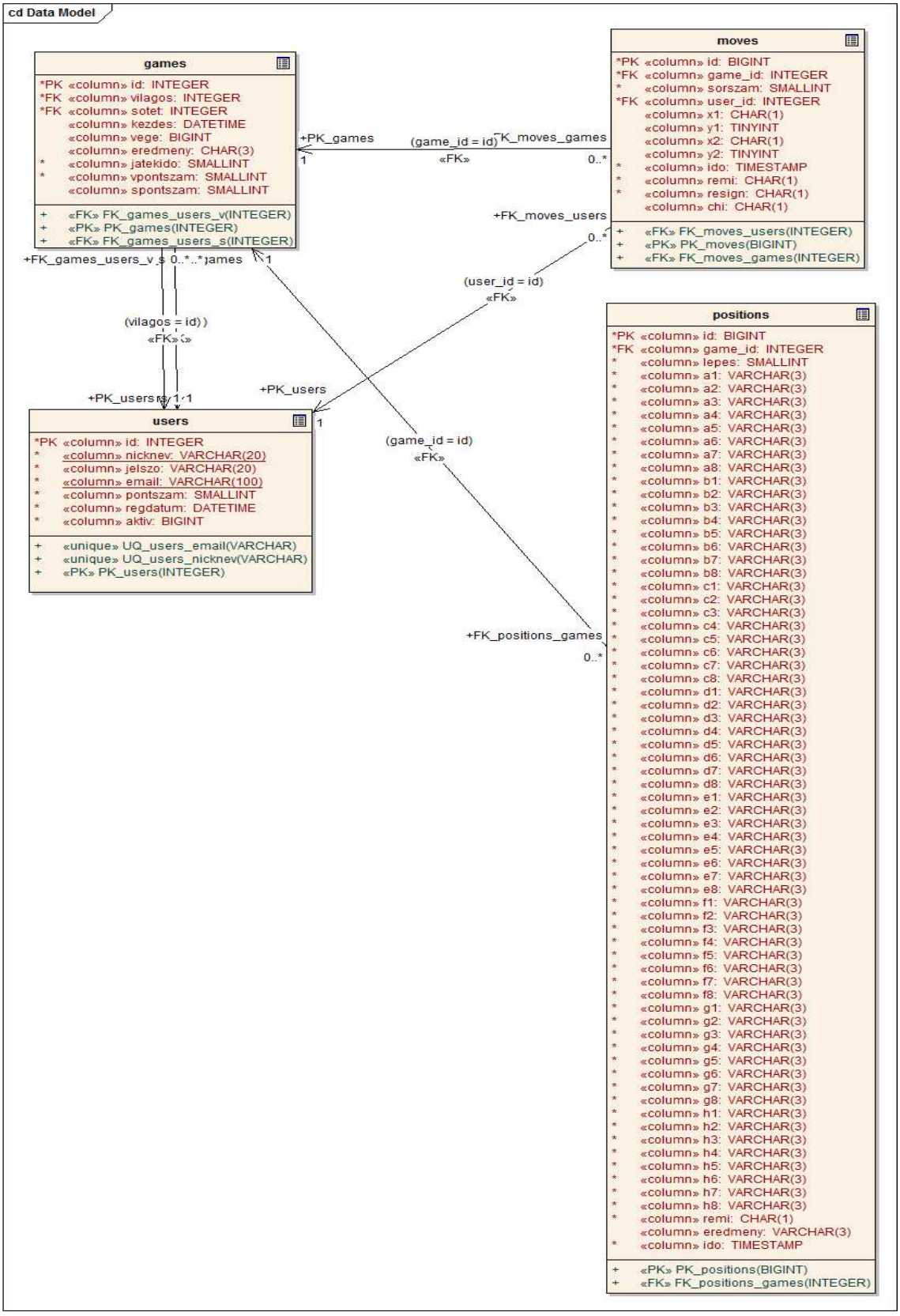
Mint az ábrán is látható az alkalmazás 3 jól elkülöníthető részre osztható:

- adatbázis, mely egy MySQL adatbáziskezelőben SQL parancsokkal manipulálható
- a webszerveren futó PHP program, mely a klienstől érkező kéréseket szolgálja ki természetesen az adatbázis használatával
- a kliens gépek böngészője által futtatott JavaScript kód, mely alapvetően a megjelenítésért felelős.

A fentiekből az is következik, hogy **vékony kliens modellt** alkalmazunk, mivel minden alkalmazásfeldolgozó és adatkezelő műveletet a webszerveren futó PHP kód végzi, a kliensen lévő böngésző a JavaScript segítségével csak a megjelenítésért felel.

A tervezést a fentebb említett 3 réteg közül az adatbázissal kezdem, tisztázva, hogy a felhasználók nyilvántartásához, a játékhoz, és a játszmák letárolásához (ez utóbbi csak az esetleges továbbfejlesztéshez a játszmák visszajátszhatóságával kapcsolatban szükséges), tehát a használati esetek megvalósításához mely adatok tárolása szükséges.

2.2. Adatmodell



III.2.2.1. ábra fizikai adatmodell

A játék elindításához és a játszmák nyilvántartásához alapvetően 3 egymástól függő entitás tárolása szükséges.

- felhasználó (játékosok)
- játszmák
- a játszmák lépései.

Az fentebb látható fizikai adatmodell ábrázolásán ennek megfelelően található három tábla a fentebb említett entitások tárolására és egy negyedik átmeneti (tranziens) tábla, amely redundáns információkat tartalmaz, de a könnyebb és gyorsabb játszma állás megjelenítés miatt szükség van rá (positions tábla).
Az adatbázist létrehozó SQL script teljes egészében megtalálható a CD mellékletben, a következőkben röviden ismertetem a táblák felépítését.

Users tábla:

A felhasználók adatait tartalmazó tábla.

- ***id***: A tábla elsődleges azonosítója (primary key), ebből következően nem lehet NULL érték. Az adatbázis-kezelő automatikusan generálja az értékét, biztosítva az egyediséget (INTEGER).
- nicknev: Felhasználói név, a játékosok ezen a néven látják egymást játék, közben VARCHAR(20), nem lehet null érték. Unique index van a mezőn.
- jelszó: A felhasználó jelszavát tartalmazó tábla VARCHAR(20), mindig ki kell tölteni,nem lehet null érték.
- email: A felhasználó e-mail címe (VARCHAR100)), kötelező kitölteni és kötelező egyedi értéket adni neki (unique index)
- pontszam: A felhasználó éppen aktuális pontszámát tartalmazza. Az alkalmazás jelen formájában nem használatos, az esetleges továbbfejlesztésre készülve került a táblába. A mező nem lehet NULL érték, jelenleg minden felhasználónál nulla kerül a mezőbe. Típusa SMALLINT
- regdatum: A felhasználó regisztrálásának dátuma. Nem lehet NULL érték, az adatbázis automatikusan feltölti a rekord rögzítésének dátumával. Típusa TIMESTAMP.

Games tábla:

A játékosok által játszott játszmák adatai tárolódnak a táblában. Ha a táblában még csak egy játékos user_id-je szerepel, akkor beszélünk kihívásról.

- ***id***: A Games tábla elsődleges azonosítója (primary Key) Az adatbázis-kezelő automatikusan generálja, típusa INTEGER.
- vilagos: A játszmában világossal játszó játékos user_id-je. A mezőt nem kötelező kitölteni, mert ha még csak kihívásról van szó, akkor lehet, hogy még csak sötéttel játszó játékos van. A mező típusa INTEGER.
- sotet: A játszmában sötéttel játszó játékos user_id-je. A mezőt nem kötelező kitölteni, mert ha még csak kihívásról van szó, akkor lehet, hogy még csak világossal játszó játékos szerepel a táblában. A mező típusa INTEGER.
- kezdés: A játszma kezdetének időpontja, illetve ha még kihívásról van csak szó akkor a kihívás utolsó frissítésének időpontja. A régi már nem aktuális kihívások törlésére felhasználható mező. Kitöltése kötelező, Típusa TIMESTAMP.
- vege: A játszma befejezésének időpontja. Kitöltése nem kötelezeő, típusa TIMESTAMP.
- eredmeny: A játszma végeredménye. Értékei lehetnek: '1-0' világos győzelme, '0-1' sötétgyőzelme, illetve '½' döntetlen. Kitöltése nem kötelező (eredmény

természetesen csak a játszma végén van), típusa CHAR(3).

- jatekido: Az egyes lépésekre maximálisan fordítható gondolkodási idő másodpercben. Jelenleg nem használatos, csak az esetleges továbbfejlesztés céljára került a táblába. Kötelező kitölteni, értéke jelenleg fix 180, Típusa SMALLINT.
- vpontszam: A világossal játszó játékos pontszáma a játszma kezdetének időpontjában. A játékos pontszámának folyamatos nyomon követésére szolgál, sőt ennek segítségével bármikor újra lehet számolni a játékos aktuális pontszámát. Jelenleg nem használatos, csak a továbbfejlesztés céljára került a táblába. Kitöltése nem kötelező típusa SMALLINT.
- spontszam: A sötéttel játszó játékos pontszáma a játszma kezdetének időpontjában. A játékos pontszámának folyamatos nyomon követésére szolgál, sőt ennek segítségével bármikor újra lehet számolni a játékos aktuális pontszámát. Jelenleg nem használatos, csak a továbbfejlesztés céljára került a táblába. Kitöltése nem kötelező típusa SMALLINT.

A táblához tartozik két FOREIGN KEY:

- FK_games_users_v, és FK_games_users_s, ezek a Games táblát és a Users táblát kapcsolják össze értelemszerűen.

Moves tábla

A játszmák lépéseit rögzítő tábla. Segítségével a játszma a későbbiek folyamán bármikor visszajátszható.

- ***id:*** A tábla elsődleges azonosítója (primary key). Az adatbázis-kezelő automatikusan generálja, típusa BIGINT.
- game_id: A lépésekhez tartozó játszma azonosítója, a kitöltése kötelező, típusa INTEGER.
- user_id: A lépést megtevő játékos azonosítója, a kitöltése kötelező a típusa INTEGER.
- sorszam: A lépés sorszáma. Kitöltése kötelező, értéke csak nullánál nagyobb lehet. Típusa SMALLINT.
- x1: A lépés kezdő koordinátájának vízszintes összetevője. Értéke 'a','b','c','d','e','f','g','h' lehet. Kitöltése nem kötelező, mert ha például valamelyik fél feladja a partit akkor nem ad meg lépést, csak eredményt. Típusa CHAR(1).
- y1: A lépés kezdő koordinátájának függőleges összetevője. Értéke 1,2,3,4,5,6,7,8 lehet. Kitöltése nem kötelező, mert ha például valamelyik fél feladja a partit akkor nem ad meg lépést, csak eredményt. Típusa TINYINT.
- x2: A lépés végső koordinátájának vízszintes összetevője. Értéke 'a','b','c','d','e','f','g','h' lehet. Kitöltése nem kötelező, mert ha például valamelyik fél feladja a partit akkor nem ad meg lépést, csak eredményt. Típusa CHAR(1).
- y2: A lépés végső koordinátájának függőleges összetevője. Értéke 1,2,3,4,5,6,7,8 lehet. Kitöltése nem kötelező, mert ha például valamelyik fél feladja a partit akkor nem ad meg lépést, csak eredményt. Típusa TINYINT.
- ido: A lépés megtételének időpontja. Kitöltése kötelező, az adatbázis automatikusan generálja az értékét. A lépések időmérése céljából került az adatbázisba, egyenlőre nem használatos. Típusa TIMESTAMP.
- remi: A döntetlen ajánlatnak, vagy a döntetlen ajánlat elfogadásának jelzésére szolgáló mező. Értékei 'i' döntetlen ajánlat, vagy döntetlen ajánlat elfogadás, 'n' nincs döntetlen ajánlat vagy nem fogadta el a döntetlen ajánlatot. Kitöltése kötelező, típusa CHAR(1).

- resign: A játszma feladásának jelzésére szolgáló mező. Kitöltése kötelező, értékei 'i' a játszma feladása, vagy 'n' a játszma nincs feladva lehetnek. Típusa CHAR(1).

- chi: Gyalogátváltozás lépéskor szükséges a kitöltése (change into). Azt mutatja meg, hogy a gyalog elérve az ellenfél alapsorát, milyen típusú tisztté változik át. Értékei lehetnek: üres – nincs átváltozás, 'v' – vezér, 'b' - bástya, 'f' – futó, 'h' – huszár. Típusa CHAR(1)

A táblához két FOREIGN KEY tartozik:

FK_moves_games ami a Moves táblát a Games táblával köti össze
FK_moves_users ami a Moves táblát a Users táblával köti össze.

Positions tábla

Az adatok átmeneti tárolására szolgáló tábla, minden játszma ahogy véget ért törlődik a tartalma. Játszma közben az aktuális állást mutatja. A lépések könnyebb és gyorsabb vizuális megjelenítését és az egyszerűbb, gyorsabb lépés szabályosság ellenőrzést teszi lehetővé. Ebben a táblában a sakktábla minden mezőjéhez rendelek egy oszlopot, amely azt tartalmazza, hogy milyen figura áll azon a mezőn. Minden lépéshez tartozik a táblában egy ilyen mező, a 0 sorszámú lépéshez tartozó rekord a kezdő állást tartalmazza.

- ***id:*** A tábla elsődleges azonosítója (primary key). Az adatbázis-kezelő automatikusan generálja, típusa BIGINT.

- game_id: A lépésekhez tartozó játszma azonosítója, a kitöltése kötelező, típusa INTEGER.

- lepes: A lépés sorszáma, melyhez az állás tartozik. Kitöltése kötelező, értékei 0 és az annál nagyobb egész számok lehetnek, típusa SMALLINT.

- a1...h8: Ez a 64 oszlop tartalmazza az aktuális sakk mezőhöz tartozó figura típusokat. Értékei lehetnek: 'u' – üres mező, 'vgy' – világos gyalog, 'vh' – világos huszár, 'vf' – világos futó, 'vb' – világos bástya, 'vv' – világos vezér, 'vk' – világos király, 'sgy' sötét gyalog, 'sh' – sötét huszár, 'sf' – sötét futó, 'sb' – sötét bástya, 'sv' – sötét vezér, 'sk'- sötét király. Kitöltése kötelező, típusa VARCHAR(3).

- remi: A döntetlen ajánlatnak, vagy a döntetlen ajánlat elfogadásának jelzésére szolgáló mező. Értékei 'i' döntetlen ajánlat, vagy döntetlen ajánlat elfogadás, 'n' nincs döntetlen ajánlat vagy nem fogadta el a döntetlen ajánlatot. Kitöltése kötelező, típusa CHAR(1).

- eredmeny: A játszma végeredménye. Értékei lehetnek: '1-0' világos győzelme, '0-1' sötétgyőzelme, illetve '½' döntetlen. Kitöltése nem kötelező (eredmény természetesen csak a játszma végén van), típusa CHAR(3).

- ido: A lépés megtételének időpontja. Kitöltése kötelező, az adatbázis automatikusan generálja az értékét. A lépések időmérése céljából került az adatbázisba, egyenlőre nem használatos. Típusa TIMESTAMP.

A táblához egy FOREIGN KEY tartozik:

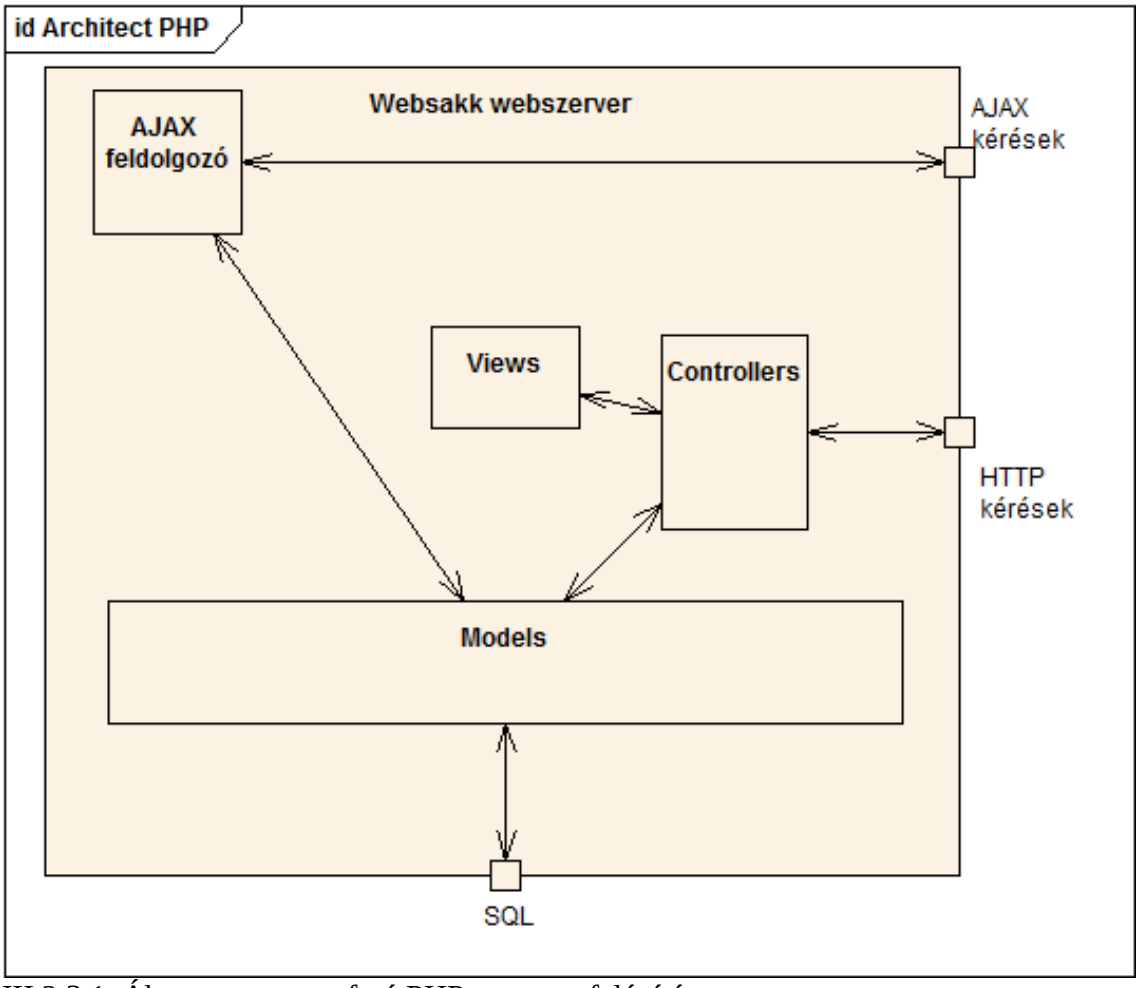
FK_positions_games ami a Positions táblát a Games táblával köti össze

2.3. A webszerveren futó PHP program architektúrája

Az alábbiakban a PHP program felépítését ismertetem részletesebben. Az ábrán látható, hogy a tervezés során az MVC (Model – View – Controller) tervezési mintát igyekeztem használni. Az MVC minta fő célja, hogy elválassza a felhasználói felület

megjelenítéséért felelős kódot, az üzleti logikától
Általában a Controller (Vezérlő) feladata, hogy feldolgozza a felhasználói adatbevitelt, illetve a tevékenységeket, függvényhívásokká képezi le.
A Modell testesíti meg az üzleti logikát.

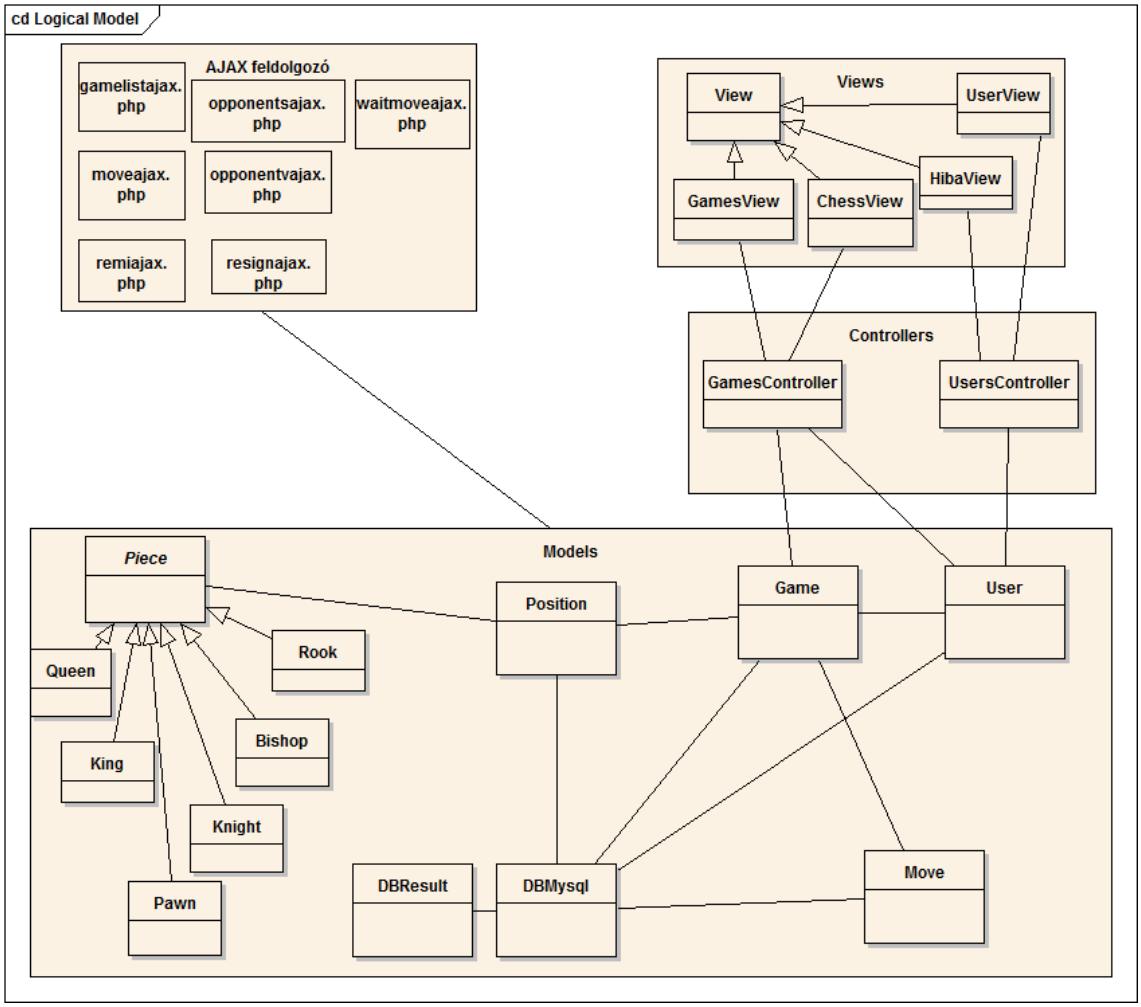
A Nézet (View) feladata, a felhasználói felület megjelenítése



III.2.3.1. Ábra a szerveren futó PHP program felépítése

Az ábrából látható, hogy a „normál” HTTP kéréseket a Vezérlők dolgozzák fel. A Modellek segítségével előállítják a válasz adattartalmát és a Nézetek függvényeinek felhasználásával előállítják a HTTP kérésre küldendő választ.
Az AJAX kérésekkel külön feldolgozó kódok foglalkoznak (pl. lépések). Ezek az AJAX feldolgozó kódok A Modellek segítségével előállítják az AJAX kérésre küldendő XML választ.

Tovább finomítva a modellt meghatározhatjuk a fenti ábrán látható komponensek szerkezetét, hogy megkapjuk a szerver program osztályszerkezetét. A következő ábrán tehát ezt a logikai modellt láthatjuk.
Az ábrából látható, hogy az AJAX kérések feldolgozását egy-egy PHP fájl végzi (ezek nem osztályok). Ezek a feldolgozók felhasználják a Models komponensben található osztályokat. A lépések ellenőrzésére a Piece abstract osztály leszármazottait, a játszmák kezelésére és mentésére, pedig a Game, Move és Position osztályokat használják.



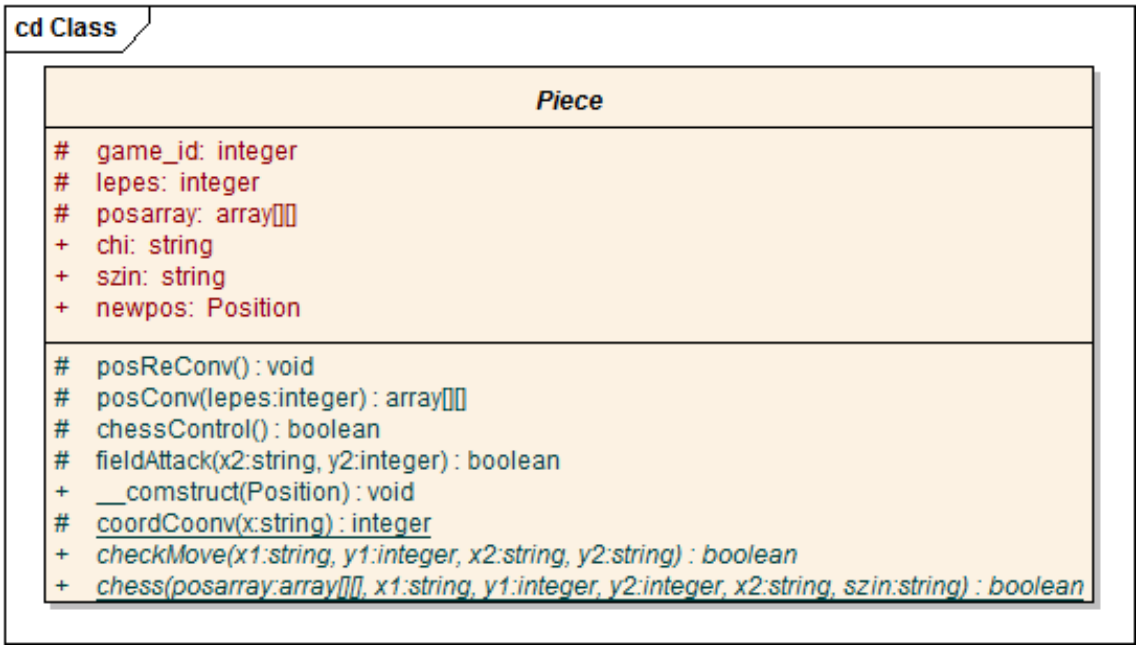
III.2.3.2. ábra A szerver program strukturális felépítése (osztályok és kapcsolataik)

A Views komponensben a View osztály és leszármazottai találhatóak, ezek az alkalmazás megjelenítéséért felelősek, ezek függvényei rajzolják ki a program vizuális elemeit, a fejlécet, láblécet, sakktáblát stb.
A Controllers komponensben két osztály található. A GamesController felelős, a kihívások kezelésével. A Game és User osztály segítségével megkeresi az adatbázisból az aktuális kihívásokat és a GamesView osztály függvényeinek meghívásával megjeleníti ezeket.
A UsersController osztály a felhasználókkal kapcsolatos teendőket vezérli. Intézi a regisztrációt, bejelentkezést és a felhasználó saját adatainak karbantartását.
Ezeket a feladatokat a User a UsersView és a HibaView osztályok segítségével végzi.
A User osztály a felhasználókkal kapcsolatos adatbázis műveletekben, a UsersView és a HibaView osztály a megjelenítésben segít.
A negyedik nagyobb csomag a Models komponens. Ebben találhatók a már korábban említett lépésellenőrző osztályok (Piece és leszármazottai), valamint a Game és User osztályok. Fontosak még a Move és Position osztályok, melyek a lépések letárolásánál végeznek fontos szerepet és az AJAX feldolgozók használják őket.
Valamint ebben a komponensben található az adatbázis kapcsolatot segítő két osztály a DBMysql és DBResult. Ezek végeznek minden adatbázis műveletet.

3. Osztályok

Az előzőekben felvázolásra kerültek az osztályok kapcsolatai, a következőkben részletesen ismertetésre kerülnek ezeknek az osztályoknak a szerkezete.

3.1. Piece osztály (sakkfigura)

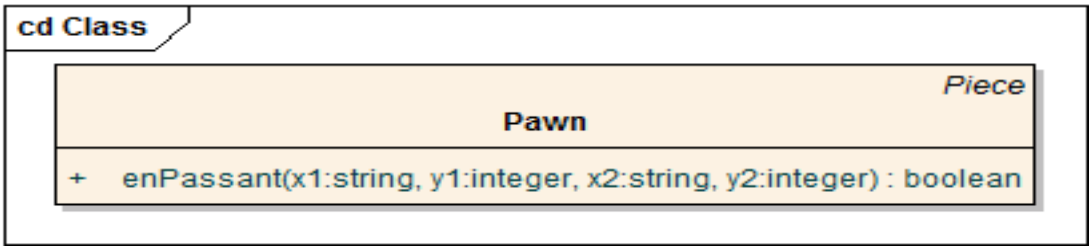


III.3.1.1. ábra Piece osztály

Ennek az abstract osztálynak a leszármazottai képesek eldönteni, hogy egy adott lépés egy adott állásban megfelel-e az osztályuk által meghatározott báb lépésének.
Az osztály konstruktora paraméterként kap egy position objektumot. Ebből létrehoz egy 8x8-as állásmátrixot (posarray) valamint feltölti az állással kapcsolatos alapváltozókat (game_id, lepes).
A checkMove függvény (amely csak a leszármazottakban van konkrétan implementálva) dönti el, hogy a neki paraméterként megadott lépés az adott figurával szabályos-e. Először azt ellenőrzi ez a függvény, hogy a mozgás szabályai szerint jó-e a lépés, majd pedig lefuttatja a chessControl függvényt, mely azt ellenőrzi, hogy az adott lépés után nem kerül-e sakkba a király. Ezek után a a posReConv függvény segítségével a lépés utáni helyzetet mutató új állásmátrixot átalakítja position objektummá (newpos), hogy a hívó azt egyszerűen le tudja tárolni lépésként.
A Bishop (futó), a Queen (vezér) a Rook (bástya) és a Knight (huszár) osztályok egymástól és a Piece osztálytól csak a bennük különbözőképpen implementált checMove és chess függvényekben különböznek.

3.2. Pawn osztály (gyalog)

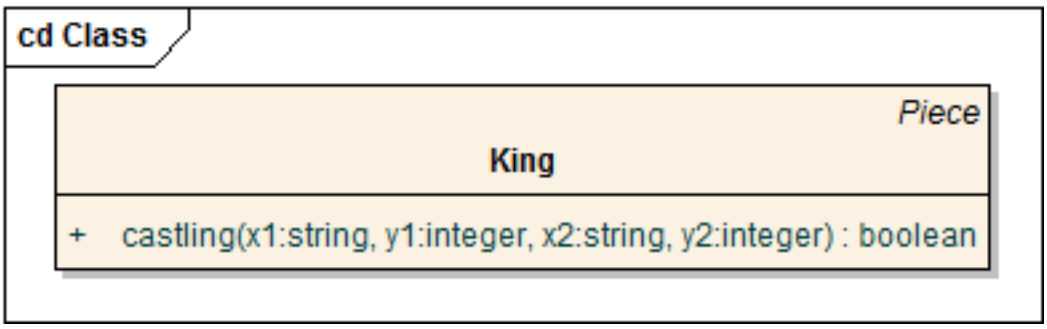
Ez az osztály, amellett, hogy megvalósítja a checkMove függvényt, tartalmaz egy enPassant függvényt is.



III.3.2.1. A Pawn osztály

Ez a szabályok között leírt „en passant” (menet közbeni ütés) gyalog ütésmódot ellenőrzi le. Úgy használható az osztály, hogy az adott koordinátájú lépésre először megvizsgáljuk a checMove függvénnyel, hogy helyes lépés-e, ha nem akkor még ezzel az enPassant függvénnyel is megpróbálkozunk. Ha ezzel sem megy akkor biztosan nem szabályos gyaloglépés. Természetesen ezután a lépés után is lefut a „sakkellenőrzés”

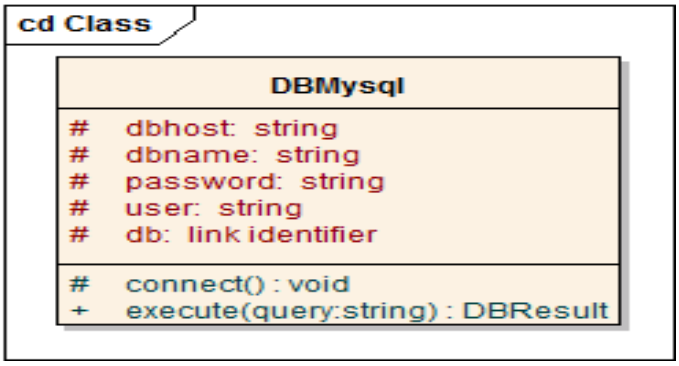
3.3. King osztály (király)



III.3.3.1. ábra King osztály

Erről az osztályról is azért kell szólnunk, mert van egy speciális lépése, mégpedig a sáncolás.
A sáncolás az egyik legbonyolultabb lépés, sok megkötés vonatkozik rá, mint ahogyan azt a követelmények között láthattuk.
A King osztály castling függvénye ellenőrzi, hogy a paraméterként neki beadott lépés lehet-e az adott állásban sáncolás.
Ehhez azt is kell tudnunk, hogy a király által átugrandó mezők nincsenek-e ütésben.
Erre szolgál a Piece osztály field Attack nevű függvénye.
Azt is ellenőrizni, kell, hogy a parti során a király és a bástya nem mozdult-e.
Mivel a position adattáblába, ideiglenesen a játszma folyamán az összes állás vektor tárolva van, nincs más teendő, mint a Position osztály canCastling függvényét használva megnézni, hogy léptek-e mr a szóban forgó figurák.
Ez az ellenőrzés nagyon egyszerű. Egy SQL SELECT-tel eldönthető, hogy az adott mezők értéke változott-e a játszma folyamán. Ha igen, akkor nem lehet sánc, mert már mozdult vagy a király, vagy a bástya.

3.4. DBMysql osztály



III.3.4.1. ábra a DBMysql osztály

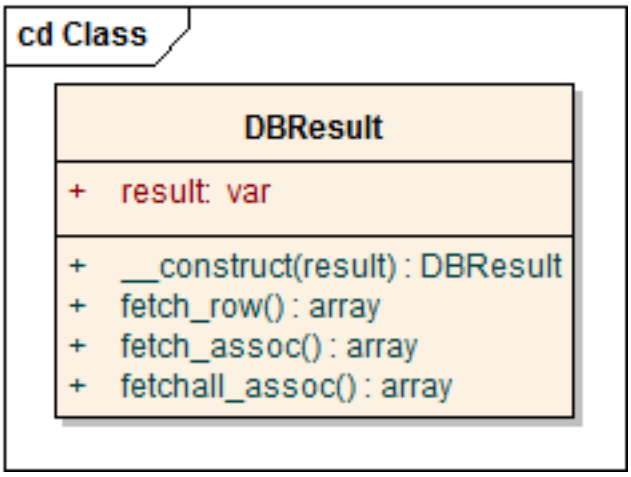
Ez az osztály felelős az adatbázissal való kapcsolatért. Tulajdonságai határozzák meg az adatbázis kapcsolatot.

- dbhost: Hostnév
- dbname: adatbázisnév
- password: jelszó
- user: felhasználónév
- db: az aktuális kapcsolat

A connect() függvény a db tagváltozóba megnyit egy MYSQL adatbázis kapcsolatot. Az execute függvény, pedig egy kapott query stringet megpróbál sql parancsként végrehajtani, visszatérési értéke pedig a következőben ismertetendő osztály objektuma lesz.

3.5. DBResult osztály

Ez az osztály egy sql lekérdezés eredményhalmazát tartalmazza. Result property-je maga az eredményhalmaz, amit konstruktora paraméterként megkap.



III.3.5.1. ábra a DBResult osztály

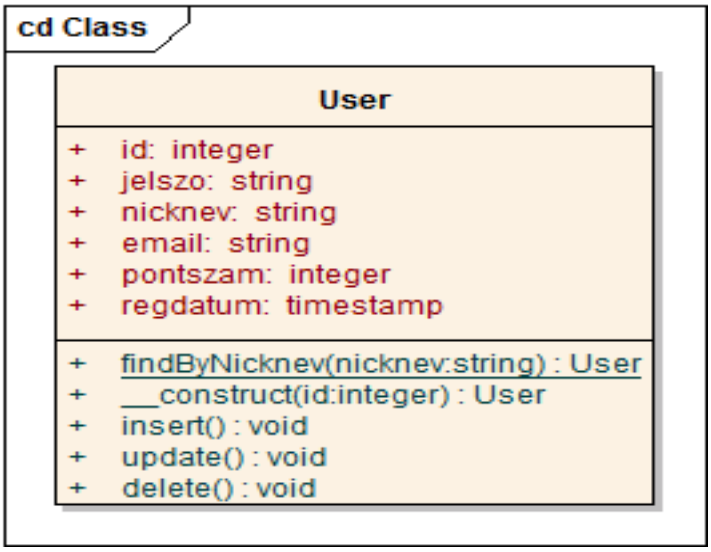
A már említett konstruktoron kívül 3 függvénye van, melyek ha létezik eredményhalmaz, azt kezelik:

- fetch_row(): egy tömbben visszaadja az eredményhalmaz egy sorát
- fetch_assoc(): egy asszociatív tömbben visszaadja az eredményhalmaz egy sorát
- fetchall_assoc() : egy kétdimenziós asszociatív tömmben visszaadja az egész eredményhalmazt.

3.6. User osztály

A User osztály a felhasználók menedzselésével kapcsolatos feladatokat segít ellátni. Tulajdonságai:

- id: azonosító
- nicknev: felhasználónév
- jelszo: jelszó
- email: a felhasználó e-mail címe
- pontszám a felhasználó mindenkori pontszáma (jelenleg használaton kívül
-regdatum: a felhasználó regisztrációjának dátuma

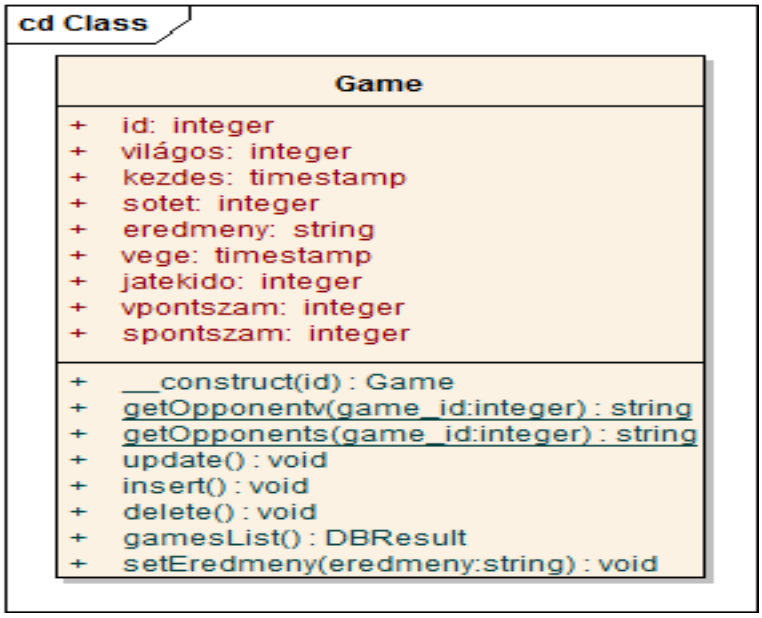


III.3.6.1. ábra a User osztály

Metódusai:

- findByNicknev: osztályfüggvény, visszatérési értéke User osztály. Egy felhasználónevet beadva paraméterkén megkeresi a hozzá tartozó rekordot az adatbázisban, ha van.
- __construct: konstruktor, megkeres egy megadott azonosítójú felhasználót az adatbázisban.
- insert: beszúr az adatbázisba egy új felhasználót a tulajdonságainak megfelelő adatkkal.
- update: módosít egy felhasználót a tulajdonságaiban megadott adatoknak megfelelően.
- delete: töröl egy felhasználót, az adatbázisból.

3.7. Game osztály



III.3.7.1 ábra a Game osztály

A fenti ábrán a Game osztály látható, amely játszmákkal kapcsolatos műveleteket végzi. Tulajdonságai:

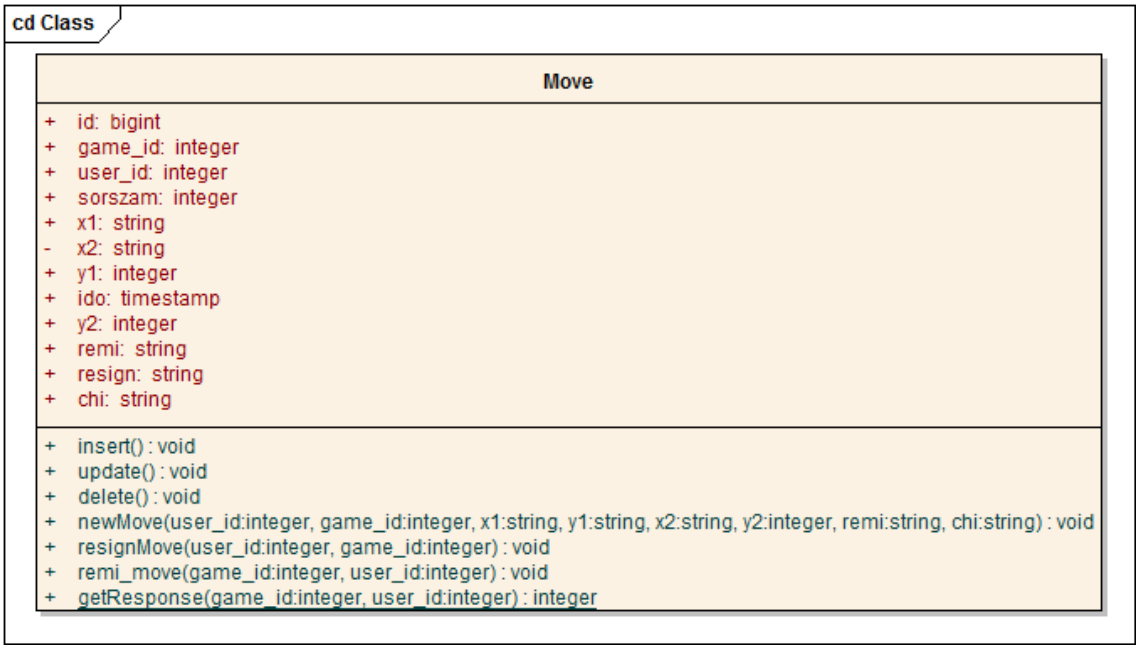
- id: játszma azonosító
- vilagos: a játszmát világossal játszó felhasználó adatai
- sötet a játszmát sötéttel játszó felhasználó adatai.
- kezdes: a játszma kezdetének, vagy a kihívás legutolsó frissítésének időpontja
- vege: a játszma végének időpontja
- eredmeny: a játszma eredménye
- ido: a játékidő, mely egy lépésre jut (jelenleg használaton kívül)
- vpontszam: világos aktuális pontszáma (használaton kívül)
- spontszam: sötét aktuális pontszáma (használaton kívül.

Metódusok:

- __construct: konstruktor, betölt egy megadott azonosítójú játékot.
- getOpponentv: osztálymetódus, egy adott azonosítójú játékban világos ellenfelének (tehát sötétnek) a felhasználói nevét adja vissza.
- getOpponents: osztálymetódus, egy adott azonosítójú játékban sötét ellenfelének (tehát világosnak) a felhasználói nevét adja vissza.
- gamesList: egy DBResult objektumban azon játszmák listáját adja vissza, amelyek még nem kezdődtek el (kihívások).
- setEredmeny: beállítja az adott játszmának a végeredményét és lezárja azt.
- insert: egy új kihívást ment le a beállított tulajdonságoknak megfelelően, és utána a position táblába lementi 0 lépésszámmal a kezdőállást.
- update: elmenti az osztály tulajdonságait a megadott játék rekordba az adatbázisba.
- delete: törli az adott játékot a játszmák táblából.

3.8. Move osztály

I



III.3.8.1. ábra a Move osztály

Ez az osztály végzi a játszma lépéseivel kapcsolatos műveleteket.

Tulajdonságai:

- id: azonosító
- game_id: a játszma azonosító
- user_id: az adott lépést megtevő játékos azonosítója.
- sorszam: a lépés sorszáma
- x1: a lépés vízszintes koordinátája
- y1: a lépés kezdőpontjának függőleges koordinátája
- ido: a lépés megtételének időpontja
- remi: küldött-e a lépéssel egy időben a játékos döntetlen ajánlatot
- resign: a játékos feladta-e a játszmát
- chi (change into): ha a lépés gyalogátváltozás volt akkor milyen figurára változott a gyalog.

Metódusok:

- insert: egy új lépést tárol le az adatbázisba
- update egy már meglévő lépést változtat az adatbázisban
- delete: egy lépést töröl az adatbázisból.
- newMove: létrehoz egy új lépést és letárolja az insert() segítségével.
- remiMove: egy döntetlen elfogadást tárol le a táblába.
- resignMove: egy játszma feladást tárol el a táblába.

3.9. Position osztály

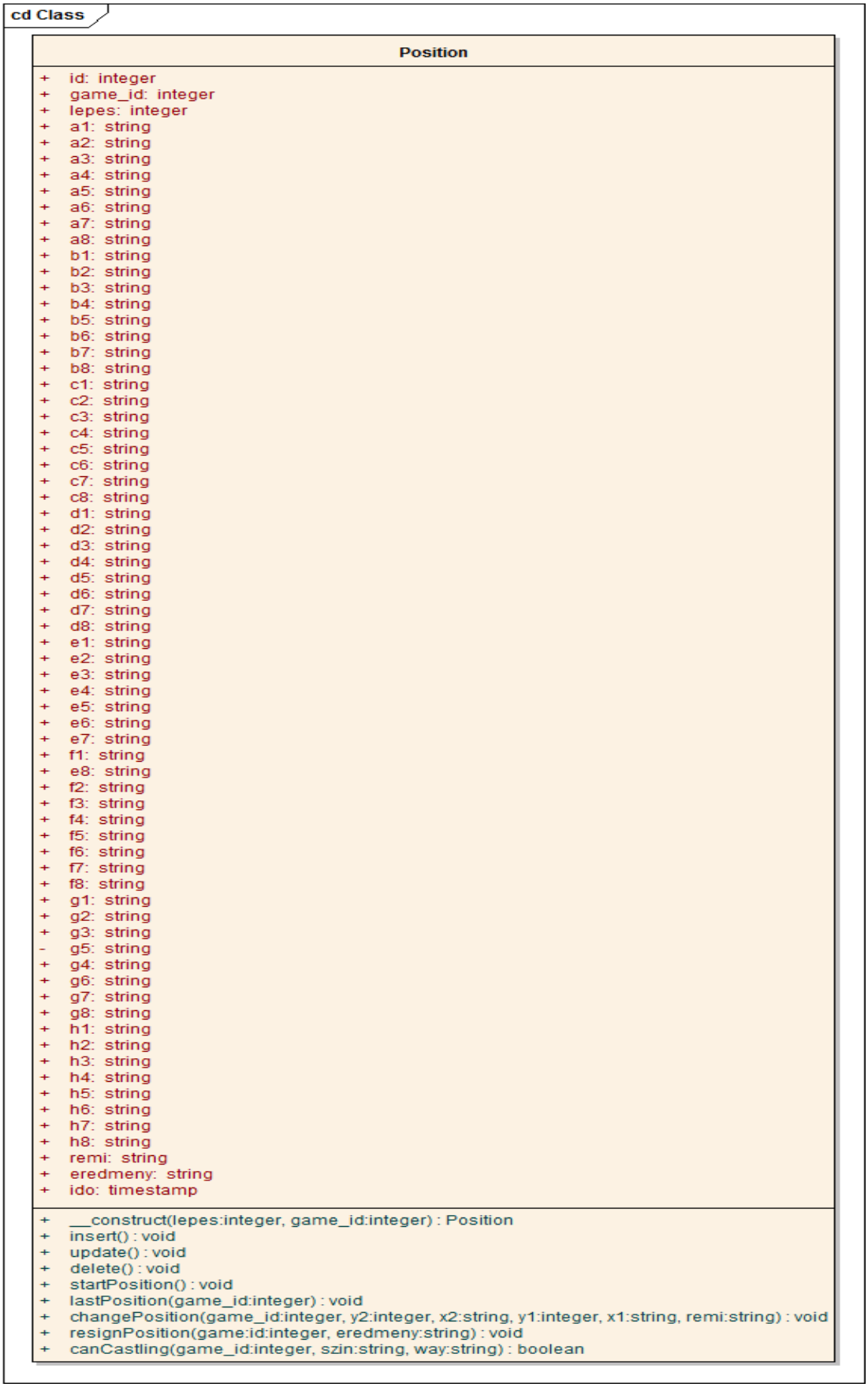
Egy adott állást egy vektorral leíró osztály, az átmeneti position táblában tartja karban az adatokat.
Tulajdonságai:

- id: azonosító
- game_id: játszma azonosító, melyhez az állás tartozik
- lepes: az adott állás mely lépés után alakult ki
- a1...h8: az állásvektor oszlopai, megmutatják, hogy a sakktábla adott mezőjén milyen figura található.
- remi: volt-e döntetlen ajánlat az adott állásban
- eredmeny: ha az adott állás végállás akkor mi az eredmény
- ido: az adott állást létrehozó lépés megtételének időpontja

Metódusai:

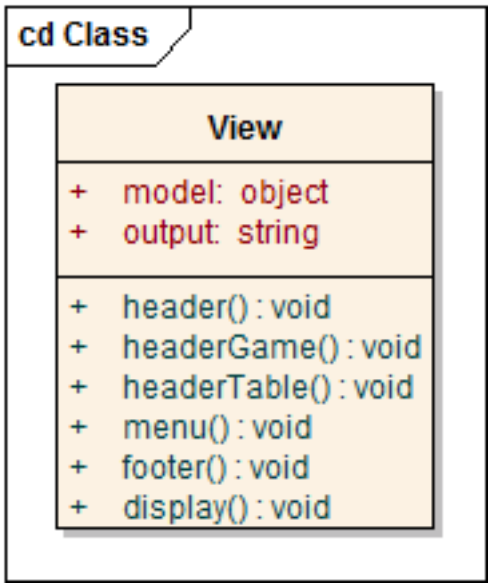
- __construct: konstruktor, ha átadjuk a megfelelő paramétereket akkor betölti az adott állást az objektum létrehozásakor.
- insert: beszúr az adatbázisba egy új rekordot az objektum tulajdonságainak értékeivel
- update: módosít egy rekordot az adatbázisban objektum tulajdonságainak értékével
- delete: töröl egy megadott rekordot az adatbázisból
- startPosition: az adott játszma azonosítóhoz beszúr az adatbázisba egy kezdőállást
- lastPosition: betölti egy adott játszmához tartozó utolsó lépés adatait az objektum tulajdonságaiba
- resignPosition: beszúrja a táblába újra az utolsó állást az eredmény jelzésével
- canCastling: paraméterként meg kell adni a függvénynek a játszma azonosítót, a színt és hogy „hosszú” oldalra (azaz a vezér felőli oldalra), vagy „rövid” oldalra akar-e sáncolni ('r','h' jelölés) és a kezdő lépésektől letárolt állásvektorokból megállapítja a függvény, hogy szabályos-e még a sánc, abban a tekintetben, hogy elmozdultak-e már az érintett figurák (király, bástya).

Az ábra a következő oldalon látható.



III.3.9. ábra a Position osztály

3.10. View osztály



III.3.10. ábra a View osztály

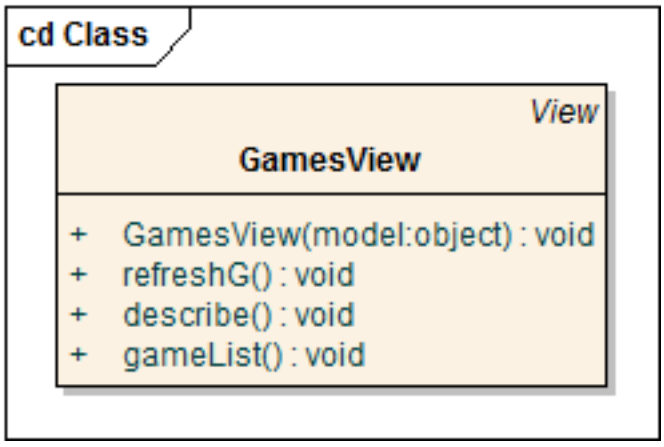
Az alkalmazás oldalainak megjelenítését meghatározó osztály. Az oldalak különböző részeit összeállító függvények egy output nevű publikus változóba készítik elő a megjeleníteni kívánt HTML kódot. Ezeket a függvényeket hívja meg az aktuális vezérlő PHP kód, majd ha a kívánt oldal összeállt, akkor a View osztály display() függvényével megjeleníti. A View osztály az őse az összes megjelenítő osztálynak.
Tulajdonságok:

- model: az esetlegesen átadni kívánt model objektum tárolója
- output: a kimeneti HTML kód gyüjtő változója

Metódusok:

- header: a lapok fejlécének kirajzolása
- headerGame: fejléc a Game ablak esetén
- headerTable: fejléc sakktábla megjelenítés esetén
- menu: menü megjelenítés
- footer: lábléc megkelenítés
- display: az összeállított HTML kódot kiíratja

3.11. GamesView osztály

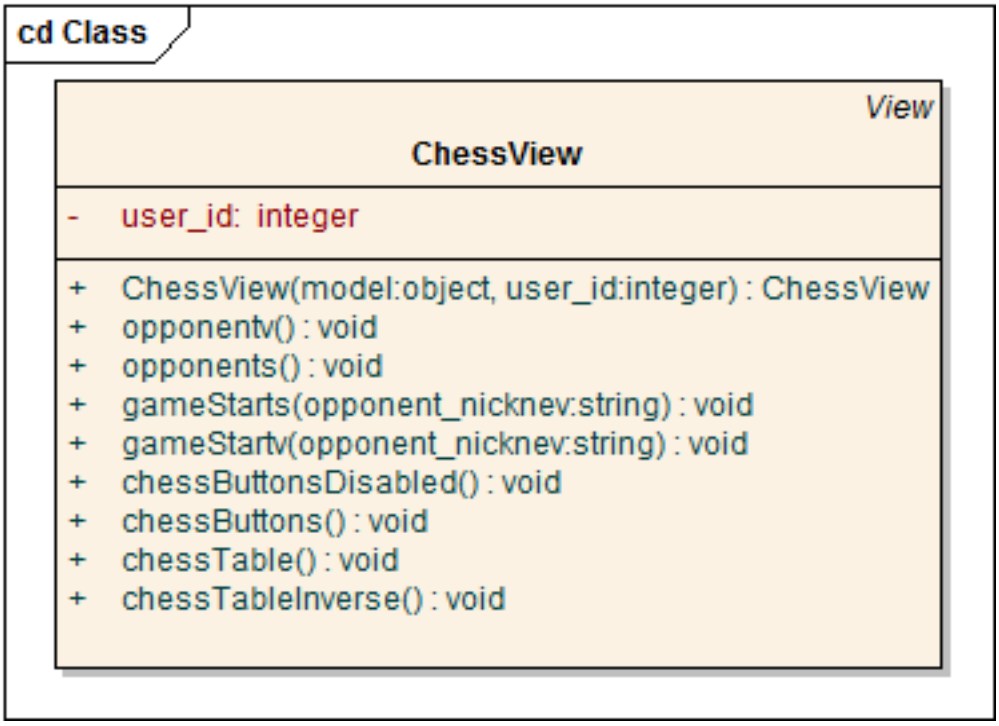


III.3.11. ábra a GamesView osztály

A GamesView osztály a kihívásokat listázó „játékterem” ablak megjelenítésében játszik szerepet.
Metódusok:

- GamesView: konstruktor
- refreshG: a lap onLoad eseményére adott választ határozza meg, erről a későbbiekben lesz szó az AJAX kéréseket tárgyaló fejezetben.
- describe: egy rövid használati leírást jelenít meg az oldallal kapcsolatban
- gameList: megjeleníti a kihívások listáját

3.12. ChessView osztály



III.3.12.1. ábra a ChessView osztály

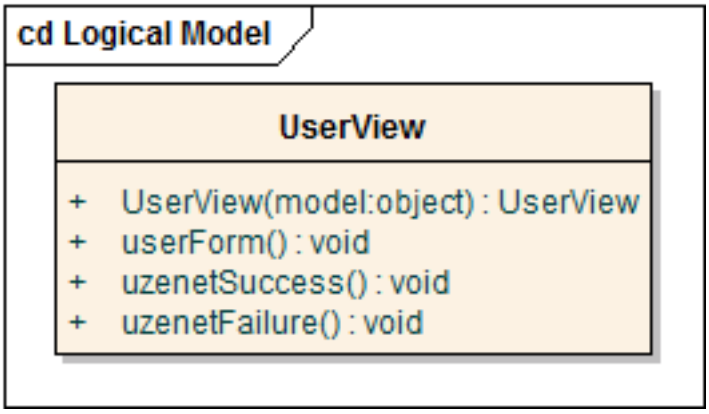
A ChessView osztály a sakkjátszmát megjelenítő oldal kirajzolásában segédkezik.
Tulajdonságai:

- user_id: a bejelentkezett felhasználó azonosítója

Metódusai:

- ChessView: konstruktor
- opponentv: ha világossal nyitják meg az ablakot az onLoad esemény lekezelésére szolgáló JavaScript függvényt adja meg, ellenfélre várás esetén.
- opponents: ha sötéttel nyitják meg az ablakot az onLoad esemény kezelésére szolgáló JavaScript függvényt hívja meg, ellenfélre várás esetén
-gameStarts: sötéttel elfogadva a kihívást az onLoad eseményre a várakozás indítása
- gameStartv: kiírja az ellenfél nicknevét, ha világossal vagyunk és vagy elfogadták a kihívásunkat, vagy mi fogadtuk el valakinek a kihívását és elkezdődött a parti
- chessButtonsDisabled: a játszma vezérlő gombjait rajzolja ki nem engedélyezett állapotban
- chessButtons: a játszma vezérlő gombjait rajzolja ki engedélyezett állapotban
-chessTable: a sakktábla kirajzolása
- chessTableInverse: a sakktábla kirajzolása fordítva a sötéttel játszott játszmákhoz

3.13. UserView osztály

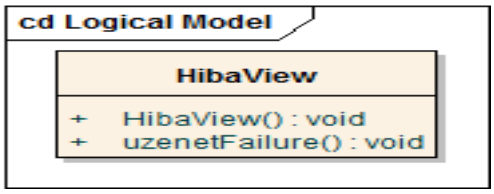


III.3.13. ábra a UserView osztály

A UserView osztály a felhasználó adatait megjelenítő oldal kirajzolásában segédkezik.
Metódusai:

- UserView: konstruktor
- userForm: kiírja a képernyőre a az aktuális user adatokat .
- uzenetSuccess: sikeres mentés esetén tájékoztató üzenetet jelenít meg.
-uzenetFailure: sikertelen mentés esetén tájékoztató üzenetet jelenít meg.

3.14. HibaView osztály



III.3.14. ábra a HibaView osztály

A HibaView osztály bejelentkezéskor, vagy regisztrációkor történő hiba jelzésére szolgál.
Metódusai:

- HibaView: konstruktor
- uzenetFailure: hibaüzenetet ír ki a képernyőre.

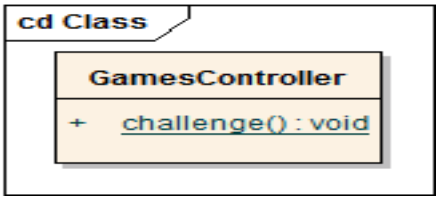
3.15. UsersController osztály

III.3.15.1. ábra a UsersController osztály

A belépéssel és a regisztrációval kapcsolatos vezérlési feladatokat látja el.
Metódusai:

- regsitration: a szükséges paramétereket átvéve megpróbál egy új felhasználót felvinni a rendszerbe. Ha ez sikerül akkor azonnal be is lépteti ezzel a felhasználóval.
- login: a megadott felhasználó nevet és jelszót ellenőrzi, hogy érvényes-e és ha igen akkor belépteti a felhasználót.

3.16. GamesController osztály



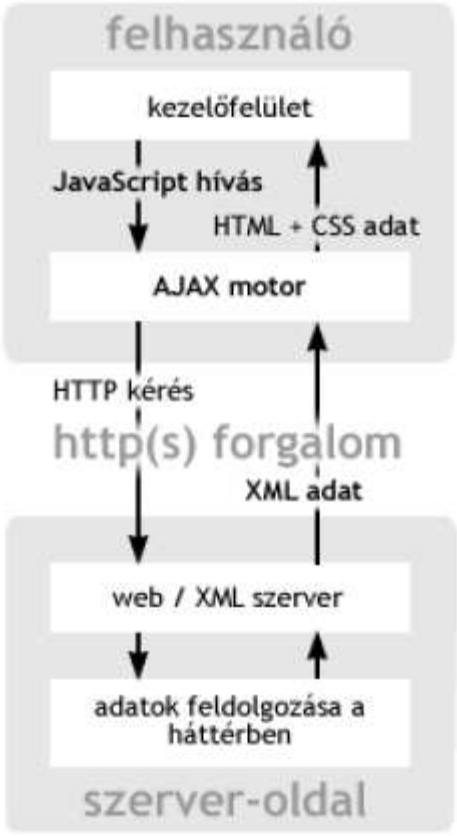
III.3.16 ábra a GamesController osztály

A kihívások ablak megjelenítését vezérli.
Metódusok:

- challenge. Létrehoz egy gamesView objektumot és annak függvényeit felhasználva kirajzolja a kihívás ablakot

4. AJAX kérések kiszolgálása

A webes sakkjátéknak az a legfontosabb jellemzője, hogy a két játékos fizikailag távol van egymástól. Alkalmazásunknak épp az a feladata, hogy a két távollevő játékos között közvetítsen és lehetővé tegye a kommunikációt. A játszma lényegét tekintve úgy zajlik, hogy az egyik játékos elküldi lépését a szerverre, amely ott letárolásra kerül. A lépés tényéről valahogy tájékoztatni kell a másik felet, aki várakozik erre a lépésre. Megtehetnénk, hogy amíg az egyik fél várakozik addig rövid időközönként folyamatosan újra meg újra letölti az egész oldalt addig amíg új lépést nem érzékel, de ez nem túl gyors, nem túl szép és nem túl elegáns megoldás. Mivel minket csak az új lépés érdekel, tehát viszonylag kevés adat kerül átvitelre, ezért logikusan adódik az igény, hogy ne az egész oldal töltődjön le időről időre, hanem csak az új állás, amely az új lépés hatására előáll. Így a sávszélességgel is takarékoskodhatunk és sokkal szebb alkalmazást tudunk építeni.
A fentebb leírtak megvalósítására kézenfekvő az AJAX (Asynchronous JavaScript and XML*)* technológia alkalmazása. A technológia lényege, hogy az AJAX alkalmazás az oldal letöltődése után további kéréseket küld a szerver felé HTTP protokollon keresztül. A kérések küldése és fogadása a JavaScript felhasználásával és aszinkron módon történik. Az AJAX technológia kulcsa az XMLHttpRequest objektum amely lehetővé teszi, hogy az oldal háttérkérelemként kapjon adatokat a kiszolgálótól, illetve küldjön adatokat a kiszolgálónak, ami azt jelenti, hogy a folyamat során nem frissíti a böngészőt. A következő ábra az AJAX kérelem–válasz modelljét mutatja:



III.4.1. ábra az AJAX webalkalmazás modellje

A felhasználó kérése JavaScript közvetítésével az AJAX motorhoz érkezik. Az AJAX kezeli le a kérést, majd dönt arról, milyen adatoknak az elküldésére van szükség. Így a választ igénylő tevékenység kérése most is eljut a szerverig, majd a feldolgozott adat vissza a felhasználóig, de nem kell az oldalt teljes egészében újra letöltenie minden részletével együtt, hanem csak azon belül a megváltozott tartalmat. Ha a motornak a továbbiakban szüksége van valamire a szervertől – ha adatot küld, fogad, vagy új felületet tölt be – azt aszinkron módon, általában XML használatával végzi.

A következőkben az alkalmazás AJAX kérésfeldolgozó részét tekintjük át. Egy funkciót (a lépésre várakozást) példakód segítségével mutatok be, a többinek pedig a logikai felépítését részletezem.

4.1. Várakozás az ellenfél lépésére (waitmoveajax.php)

Az egyik játékos elfogad egy kihívást, amelyben sötéttel kell játszania (nevezzük a továbbiakban S-nek). Amint rákattint a kihívás elfogadása linkre, megnyílik egy új ablak (ChessView) az alapállással és ekkor S-nek várakoznia kell az ellenfél lépésére. A ChessView oldal Kódjában szerepel a következő sor:

```
<body onLoad="waitMove('.$this->model->id.','.$this->userid.')">
```

Tehát az oldal betöltődésekor a **waitMove** JavaScript függvény fog lefutni. Tekintsük meg tehát ezt a függvényt, melyet az alkalmazás js könyvtárában a positions.js fájlban találunk meg:

```
function waitMove(gameid,userid)
{
        //XMLHttpRequest objektum létrehozása
        xmlHttp=GetXmlHttpObject()
        if (xmlHttp==null)
        {
            alert ("A böngészo nem támogatja a HTTP kérést!")
            return
        }
        game_id=gameid
        user_id=userid
        // URL összeállítása
        var url="../controllers/waitmoveajax.php?
game_id="+game_id+"&user_id="+user_id+"&sid="+Math.random()
        //Callback függvény megadása
        xmlHttp.onreadystatechange=yourMoveVisualise
        // kérés leírása
        xmlHttp.open("GET",url,true)
        // kérés elküldése
        xmlHttp.send(null)
}
```

Az első lépés az AJAX működés lelkét adó XMLHttpRequest objektum létrehozása. Ha

létrejött a kommunikációs objektum akkor összeállítjuk a meghívandó PHP kód URL-jét.
Egy véletlen számot hozzáadok az url-hez, elkerülve azt, hogy a szerver egy előző - lefutott - esemény eredményét adja vissza.
Az AJAX aszinkron működése azt jelenti, hogy a szerver felé küldött kérésre a JavaScript nem várja meg a választ hanem a felhasználót hagyja tovább dolgozni A kommunikációs objektumunk számára tehát egy úgynevezett. callback (visszahívható) függvény nevét adja meg. Tehát amikor a szerverről a waitmoveajax.php kódtól visszaérkezik a válasz akkor a yourMoveVisualise JavaScript függvényünk kapja meg a vezérlést, amely feldolgozza a választ, és szükség szerint frissíti a lap tartalmát.

Nézzük most meg akkor, hogy mi történik a szerveren a waitmoveajax.php meghívása után:

```
<?php
require_once "../models/movemodel.php";
require_once "../models/positionmodel.php";

// Ciklus, melyben 1 sec-ként leellenőrzöm hogy van-e válaszlépés
while ($responsemove==0 or is_null($responsemove)) {
        sleep(1);
        $responsemove=Move::getResponse($_GET["game_id"],
$_GET["user_id"]);
}

//Válaszlépés esetén egy position objektumban állásvektor előállítása
$position = new Position($_GET["game_id"],$responsemove);
if ($position->eredmeny=="" or is_null($position->eredmeny)) {
        $position->eredmeny="0-0";
}

//Válasz XML előállítása
header('Content-Type: text/xml; charset=ISO-8859-2'); echo'<?xml
version="1.0" encoding="iso-8859-2" standalone="yes"?>' .'<response>';
echo"<a1>".$position->a1."</a1>";
echo"<a2>".$position->a2."</a2>";
echo"<a3>".$position->a3."</a3>";
        …
        …
echo"<h4>".$position->h4."</h4>";
echo"<h5>".$position->h5."</h5>";
echo"<h6>".$position->h6."</h6>";
echo"<h7>".$position->h7."</h7>";
echo"<h8>".$position->h8."</h8>";
echo"<remi>".$position->remi."</remi>";
echo"<eredmeny>".$position->eredmeny."</eredmeny>";
echo "</response>";
?>
```

Látható, hogy a szerveren elkezd futni egy ciklus, mely mindaddig keresi a positions táblában a válaszlépést, amíg az meg nem érkezik. Ha ez megérkezett, akkor egy position objektumba betöltjük azt az állásvektort, ami a válaszlépés után rendelkezésre áll a positions táblában, és ebből az állásvektorból állítjuk elő az AJAX kérésre az XML formátumú választ, amely az előző JavaScript szerinti Callback függvényhez kerül feldolgozásra. Lássuk tehát, hogy hogyan történik az XML válasz feldolgozása a yourMoveVisualise JavaScript függvényben.

```
function yourMoveVisualise()
{
        if (xmlHttp.readyState == 4)
        {
            if (xmlHttp.status == 200)
            {
                try {
                    var response = xmlHttp.responseText;

                    if (response.indexOf("ERRNO") >= 0 ||
response.indexOf("error:") >= 0 || response.length == 0) throw(response.length
== 0 ? "Void server response." : response);

                    response =xmlHttp.responseXML.documentElement;
                    if(response.childNodes.length)
                    {
                        getFigur("a1","b",response);
                        getFigur("a2","w",response);
                        getFigur("a3","b",response);
                        getFigur("a4","w",response);
                    ...
                    ...
                    ...
                        getFigur("h5","w",response);
                        getFigur("h6","b",response);
                        getFigur("h7","w",response);
                        getFigur("h8","b",response);
                        var eredmeny=
response.getElementsByTagName("eredmeny"[0].childNodes[0].nodeValue
                        if (eredmeny=='1-0' || eredmeny=='0-1' ||
eredmeny=='1/2') {
            document.getElementById("eredmeny").innerHTML = eredmeny
                            control_disabled()
                        } else {
                            control_enabled()
                        }
                        if (response.getElementsByTagName("remi")
[0].childNodes[0].nodeValue=='i') {

        document.getElementById("elfogadremi").disabled=false
```

```
                                        } else {

            document.getElementById("elfogadremi").disabled=true
                                        }
                                    }
                                    } catch(e)
                                    {
                                        // Böngésző timeout miatti új meghívás
                                        waitMove(game_id,user_id)
                                    }
                        } else { alert (xmlHttpGetSuggestions.statusText);

                        }
                    }
}
```

Látható, hogy a függvény figyeli az XMLHttpRequest objektum readyState állapotát és ha ez azt jelzi, hogy a válasz teljes akkor kezd a feldolgozásba. Egyébként az alábbi táblázatban szereplő értékei lehetnek még a readyState állapotnak.

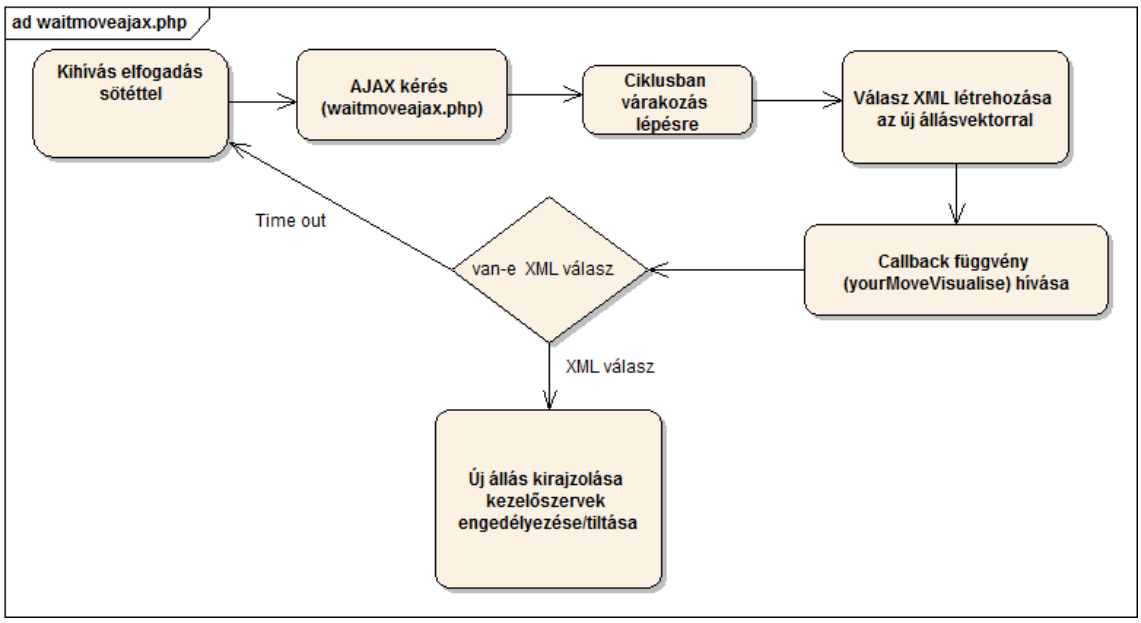
0	UNINITIALIZED	az open() függvény még nem került meghívásra
1	LOADING	a send() függvény még nem került meghívásra
2	LOADED	a send() függvény már meghívódott, a *headers* és a *status* értékei elérhetők
3	INTERACTIVE	letöltés; a *responseXML* már tartalmazza a válasz egy részét
4	COMPLETE	minden művelet befejeződött.

Tehát ha a válasz megérkezett akkor a getFigur JavaScript függvény az XML állásvektor alapján eldönti, hogy a táblázatban megrajzolt sakktábla mely cellájába mely figurának, mely háttérszínű képét rakja ki.
Példa sor a getFigur függvényből.:

```
document.getElementById(square).innerHTML = '<img src="../img/wpb.gif"
width="34" height="34" />';
```

A sakktáblán lévő állás módosítása után még megvizsgálja a függvény, hogy van-e eredménye már a partinak, vagy érkezett-e döntetlenajánlat, és ennek megfelelően tilt le vagy engedélyez kezelőszerveket. Egy fontos lépés látható még a kód végén kiemelve.

Ha sokáig nem érkezik válaszlépés és a böngésző time out miatt leállítja a kérést, akkor újra meghívjuk a waitMove JavaScript függvényt és a folyamat elölről kezdődik. A programban található AJAX hívások hasonlóan zajlanak. Egy mondatban összefoglalva a következőképpen:
egy JavaScript függvényben deklarálunk egy XMLHttp objektumot, amely meghív egy PHP kódot és a válasz XML-t átadja agy feldolgozó JavaScript függvénynek.
A fentebb részletesen vázolt várakozás az ellenfél lépésére művelet az alábbi folyamatábrával szemléltethető:

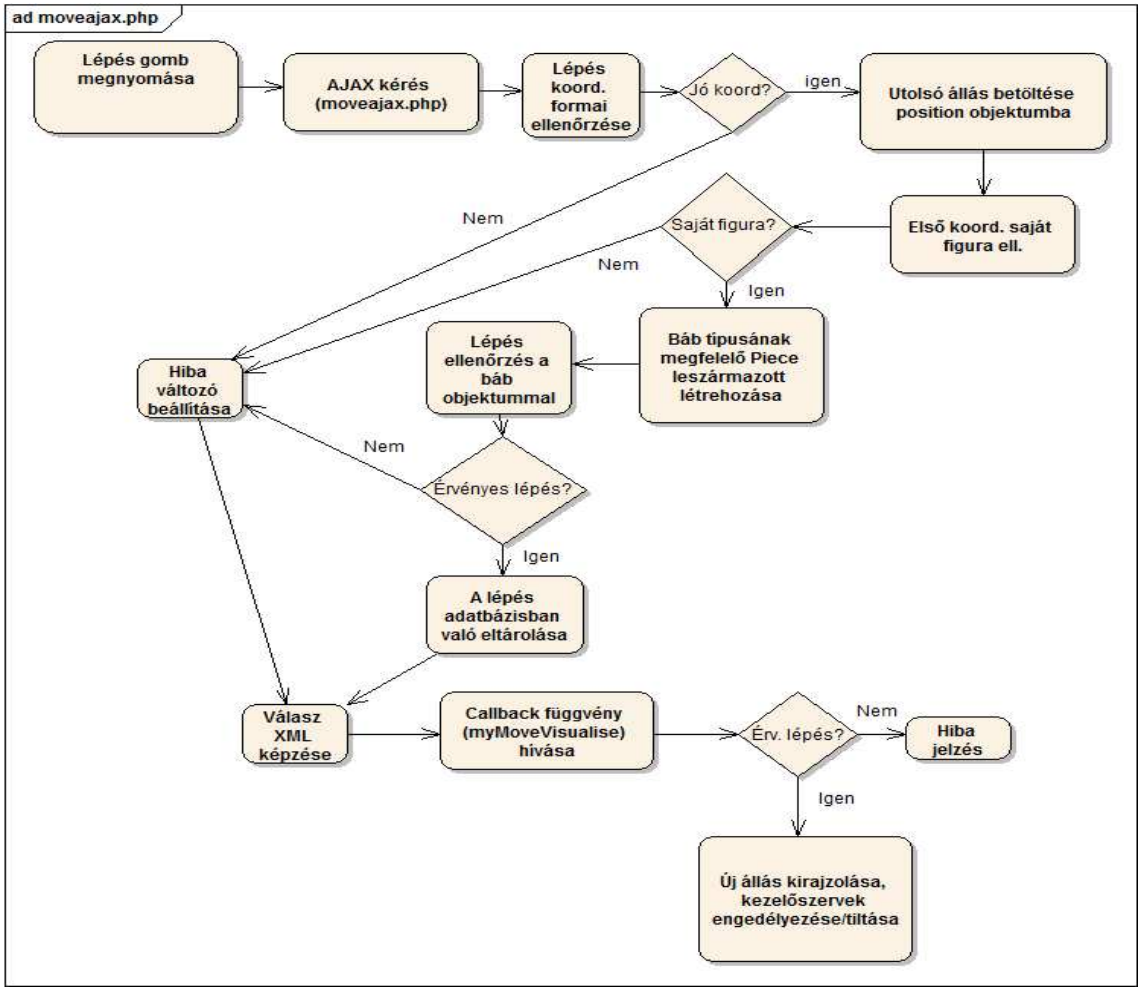


III.4.1.1. ábra várakozás lépésre folyamatábrája

4.2. Lépés (moveajax.php)

Az III.4.2.1. ábrán egy lépés folyamán lezajló tevékenységek folyamatábrája látható.

A játékos aki soron következik megnyomja a lépés gombot. A gombnyomás hatására a vezérlés az AJAX hívást intéző JavaScript függvényhez kerül, amely elkéri a webszervertől a moveajax.php lapot paraméterként átadva többek között a játszma azonosítót és a lépés paramétereit.
Ezen a PHP oldalon először is ellenőrzésre kerülnek a lépés koordinátái formai szempontból Tehát az x1y1 paraméter pontosan kettő hosszúságú karakterlánc, első karaktere az {a,b,c,d,e,f,g,h} halmaz eleme, második karaktere 1 és 8 közé eső egész szám lehet. Az x2y2 paramétere minimum kettő, maximum három hosszúságú karakterlánc. Első karaktere szintén az {a,b,c,d,e,f,g,h} halmaz eleme, második karaktere 1 és 8 közé eső egész szám, az esetleges harmadik karakter pedig a {h,f,b,v} halmaz egyik eleme lehet. Ha ez nem teljesül, akkor a hiba változót 'i' értékre állítjuk és ezzel az értékkel adunk XML választ. Ha a koordináták formailag helyesek akkor egy position objektumba betöltjük a játszmához tartozó utolsó állást.
Ebben az állásvektorban ellenőrzésre kerül, hogy a lépés első koordinátája saját figurát takar-e. Ha nem akkor az XML válaszban hibajelzést küldünk.



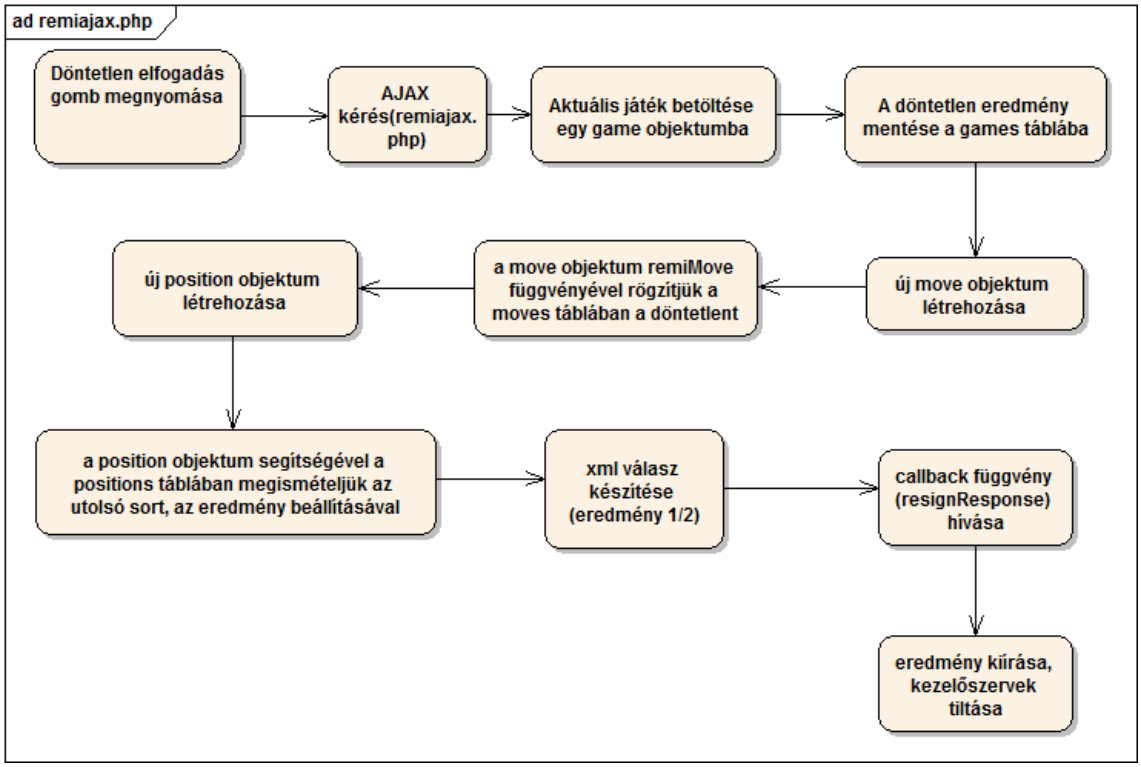
III.4.2.1 ábra a lépés folyamatábrája

Ezután megállapítjuk a figura típusát és ennek megfelelő objektumot hozunk létre. Ennek az objektumnak a segítségével ellenőrizzük a lépés helyességét. A figura típusától függetlenül először a checkMove függvényét hívjuk meg az objektumnak, de ha ez a függvény nem ad érvényes kimenetet akkor aztán a figura típusától függően egyéb ellenőrző függvényt is indíthatunk (pl.: gyalog esetében enPassant függvény, király esetében castling függvény). Ha az ellenőrző függvények nem találják érvényesnek a lépést akkor az XML válaszban hibajelentést küldünk. Ha lépés helyesnek bizonyul akkor tárolásra kerül az adatbázisbna (mind a moves, mind a positions táblában), majd az XML válaszban elküldjük mind a kialakult állást, mind pedih, hogy a lépés helyes. Ezt a választ dolgozza fel a myMoveVisualise JavaScript függvény, amely megjeleníti az új állást és beállítja a vezérlő elemek engedélyezését, ha a lépés nem helyes akkor pedig ez a függvény egy hibajelzést ad a játékosnak.

4.3. Döntetlen ajánlat elfogadása (remiajax.php)

Ha egy játékos egy játszma folyamán úgy ítéli meg, hogy az állás alapján döntetlent szeretne ajánlani, akkor meg kell várnia amíg lépésre kerül. Ekkor az érvényes lépésvel együtt elküldhet egy döntetlen ajánlatot a lépésével együtt, mely a lépéssel együtt letárolásra kerül az adatbázisba illetve, amiről az ellenfele úgy szerez tudomást, hogy

nála a döntetlen elfogadása gomb engedélyezett állapotba kerül. Ilyenkor lépés helyett megnyomhatja ezt a gombot. A gomb megnyomásának hatására ahhoz a JavaScripthez kerül a vezérlés mely lekéri a webszervertől a remiajax.php lapot és átadja paraméterként az aktuális játszma azonosítóját. Ez a lap először létrehoz egy game objektumot az adott játszmához, beállítja ennek eredmény változóját '1/2' értékre. Ezután egy move objektumot is létrehoz és az adott játszma utolsó lépéseként letárolja a döntetlent, majd létrehoz egy position objektumot is amely segítségével a positions táblában megismétli a végállást immár döntetlen végeredmény jelzéssel. Ennek következtében az ellenfél aki a döntetlen ajánlatot tette, amikor legközelebb belenéz a positions táblába észlelni fogja az ajánlat elfogadását, és azt hogy a játszma döntetlennel véget ért. A sikeres adatbázis műveletek után a remiajax.php lap megképzi az XML választ, amely az eredményt tartalmazza. Az XML válasz a resignResponse JavaScript feldolgozó függvényhez kerül, (ez a feldolgozó közös a feladás feldolgozó függvényével, mert pontosan ugyanaz a teendő abban az esetben is), mely megjeleníti az eredményt, és beállítja a kezelőszervek engedélyezését.
Az alábbi képen a fentebb leírtak vannak ábrázolva.

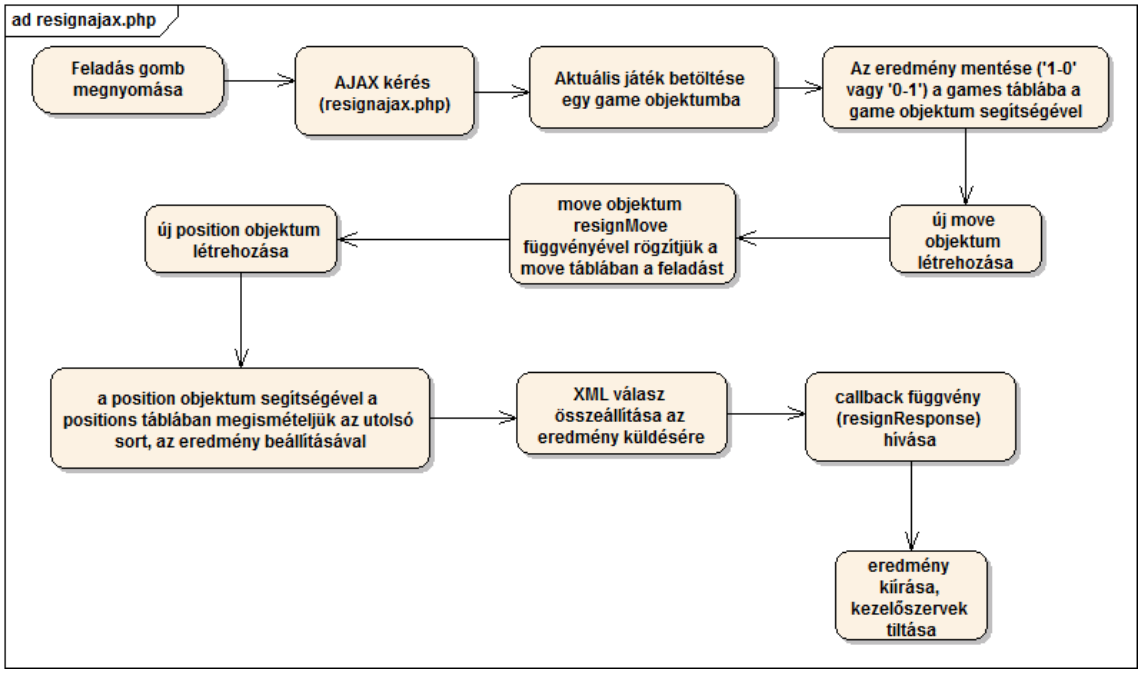


III.4.3.1. ábra a döntetlen elfogadás folyamatábrája

4.4. A játszma feladása (resignajax.php)

A III.4.4.1. ábrán a feladás folyamatábrája látható.

Ha játszma közben a játékos fel akarja adni a partit, akkor megvárja míg lépésre kerül és akkor lépés helyett megnyomja a feladás gombot. Ekkor egy JavaScript lekéri a resignajax.php lapot.



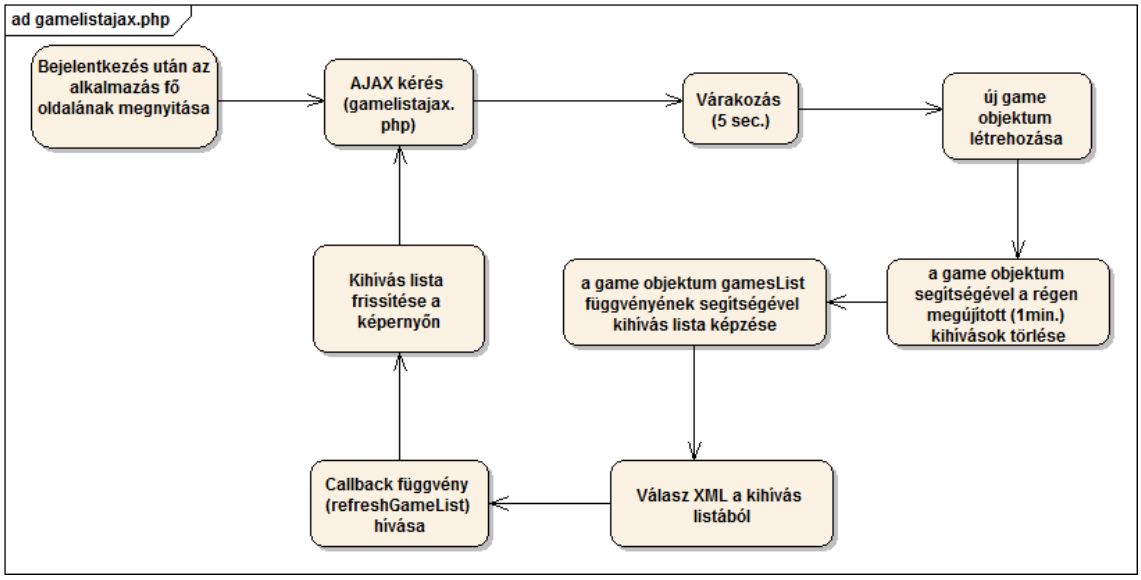
III.4.4.1 ábra a feladás folyamatábrája

A lap betölti a játszma adatait az adatbázisból egy game objektumba, beállítja az eredményt a játékos színétől függően '1-0'-ra , vagy '0-1'-re és lementi az adatbázisba. Ugyanezt teszi a move objektummal illetve táblával is. Majd egy position objektum segítségével a positions táblában lévő utolsó lépés sorát megismétli és beállítja az eredményt. Emiatt (hasonlóképpen a döntetlen elfogadásánál tárgyaltakhoz) az ellenfél is tudomást szerez a feladásról és a játszma befejeződéséről. A következő teendő, az XML válasz összeállítása, amely az eredményt tartalmazza. Az XML választ a resignResponse JavaScript függvény dolgozza fel a döntetlen elfogadásához hasonlóan (megjeleníti az eredményt, és beállítja a kezelőszervek engedélyezését).

4.5. Kihívások listájának frissítése (gamelistajax.php)

A III.4.5.1. ábrán a kihívások listájának frissítésével kapcsolatos teendők folyamatábrája látható.

Onnantól kezdve, hogy megnyitjuk az alkalmazás fő oldalát, amely a kihívások listáját tartalmazza, ez a lista folyamatos frissítés alatt áll, egészen addig, míg ezt az ablakot be nem csukjuk. Erre szükség, van, hogy a potenciális játékosok folyamatosan nyomon tudják követni azokat a kihívásokat, amelyeket elfogadva elindulhat egy játszma.
A GameView oldal betöltésekor az onLoad esemény a game.js JavaScript fájlból a refreshGame JavaScript függvényt hívja meg, amely az AJAX kérést indítja, lekérve a gamelistajax.php lapot. Ennek a lapnak a kódja egy 5 másodperces várakozással indul, hogy a szervert a folyamatos frissítési kérések ne terheljék túlságosan.
Ezután egy új game objektum segítségével törli a lejárt frissítésű (1 percnél régebbi) kihívásokat az adatbázisból. Így nem maradhatnak a games táblában régi esetleges megszakadt kapcsolatú játékosok elavult kihívásai.

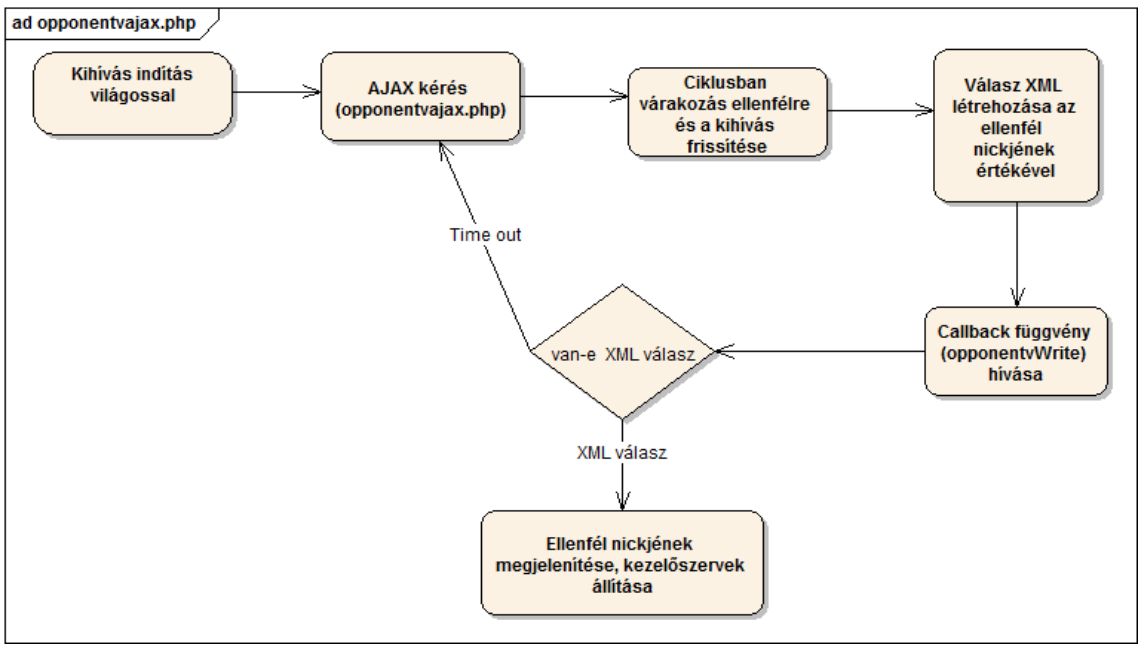


III.4.5.1. ábra a kihívások listájának frissítése

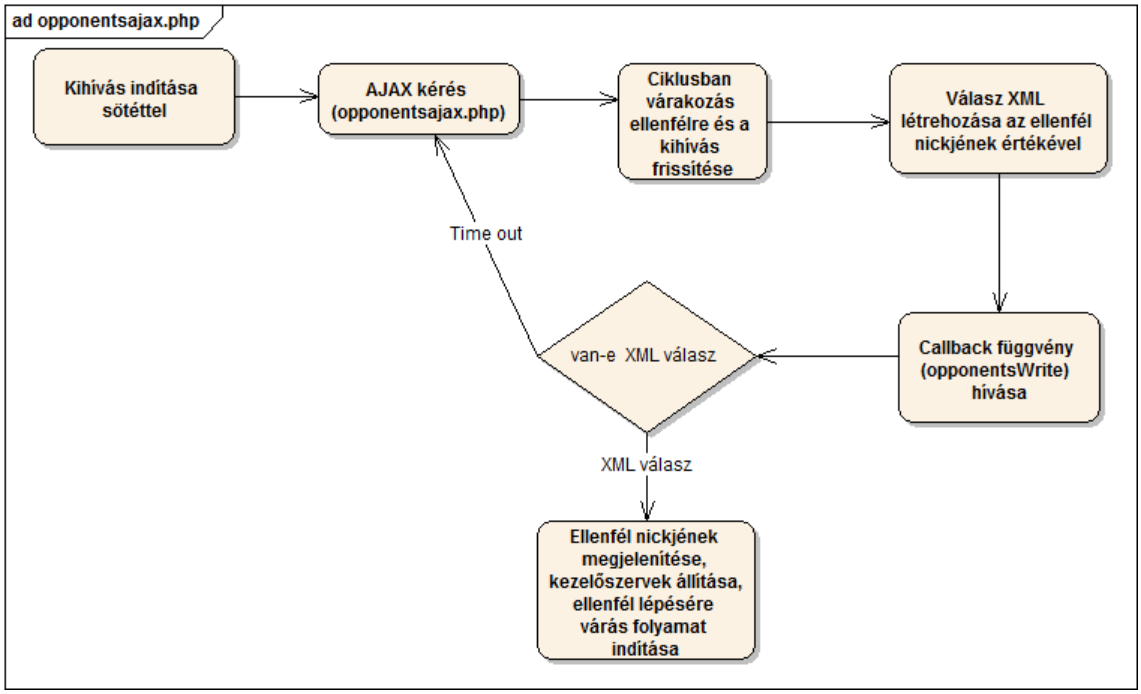
Ezek után a gamesList függvény segítségével egy XML választ képzünk a kihívások listájából. Az XML választ a refrsehGameList feldolgozó JavaScript függvény jeleníti meg kliens oldalon, majd pedig újraindítja a refreshGame JavaScript függvényt elölről kezdve az egész folyamatot, tehát kliens oldalon körülbelül 5 másodpercenként frissülni fog az aktuális kihívások listája.

4.6. Várakozás ellenfélre (opponentvajax.php, opponentsajax.php)

A következőkben a várakozás ellenfélre folyamatot mutatjuk be. A két különböző színnel való várakozást két különböző AJAX kérés dolgozza fel, de hasonlóságuk miatt ezeket egyszerre tárgyaljuk. Alább látható a két folyamatábra.



III.4.6.1. ábra várakozás ellenfélre világossal



III.4.6.2 ábra várakozás sötéttel

Egy játékos a GameView ablak egyik funkciógombjának használatával egy kihívást indít. Ekkor az opponentv JavaScript függvény indít egy AJAX kérést, az opponentvajax.php oldal lekérésével (sötéttel való kihívás esetén opponentsajax.php). Ezen a PHP oldalon egy ciklust indítunk, amely minden körben 1 másodpercet várakozik, majd a Game osztály getOpponentv statikus függvényének segítségével leellenőrzi, hogy csatlakozott-e már ellenfél a játékhoz. Ha nem akkor megújítja a kihívást azzal, hogy az időbélyeg mezőjébe friss időpontot ír. Ha érkezett ellenfél akkor kilép a ciklusból és létrehoz egy XML választ az ellenfél nickjével. Ezt a választ az oppnentvWrite (illetve sötét szín esetében az opponentsWrite) JavaScript függvény feldolgozza. Világossal megjeleníti az ellenfél nickjét és beállítja a kezelőszervek engedélyezési tulajdonságát a megfelelő értékre. Sötéttel indított kihívásnál ezenkívül az opponentsWrite függvény még ezután elindít egy ellenfélre való várakozást a waitmove Javascript függvény segítségével.

5. Tesztelési terv

A tesztelés feladata a program helyességének, megbízhatóságának, tudásának, vizsgálata. Két alapvető célja a rendszerben található hibák felderítése és annak ellenőrzése, hogy a rendszer a felhasználó céljai szerint használható.
Mivel jelen alkalmazás kifejlesztésekor nem volt cél, hogy egy nagyszámú játékost kiszolgáló komoly játékszervert építsek, hanem csak a technológia kipróbálása illetve egy továbbfejleszthető „játékmotor” kialakítása szerepelt terveim között, ezért úgynevezett „stressz-tesztet” (azaz hirtelen nagy terhelést szimuláló használatot) nem terveztem. Ugyancsak figyelmen kívül hagytam hasonló okokból „ellenségtesztet” sem terveztem, ezek elvégzése az esetleges továbbfejlesztés esetén válnak fontossá.

Természetesen az alkalmazás fejlesztése folyamán az egyes objektumok, részegységek

folyamatos tesztelésnek voltak alávetve, tehát a *komponens tesztelés* folyamatos volt.

Igyekeztem olyan teszteseteket tervezni, melyek segítségével az összes fő funkció helyes működése kipróbálható, valamint néhány játék teljes folyamatát végigkövetni. A feladat jellegét tekintve azonban (mint általában is) „kimerítő tesztelés” (azaz az összes „lehetőség” kipróbálása) nem lehetséges, illetve aránytalanul költséges lenne.

A teszteseteket a főbb funkciók szerint részekre osztottam.

1. Bejelentkezés, regisztráció:
    1.1. Regisztráció gomb megnyomása adatok megadása nélkül
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.2. Regisztráció gomb megnyomása, de nincs kitöltve a felhasználónév
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.3. Már létező felhasználónév megadása regisztrációkor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.4. Nem adom meg az egyik jelszót regisztrációkor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.5. Nem ugyanazt a jelszót adom meg regisztrációkor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.6. Nem adok meg e-mail címet regisztrációkor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.7. Nem adok meg felhasználónevet a bejelentkezéskor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.8. Nem létező felhasználónevet adok meg bejelentkezéskor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.9. Nem adok meg jelszót bejelentkezéskor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.10. Rossz jelszót adok meg bejelentkezéskor
Elvárt: A „Nem sikerült a bejelentkezés” feliratú hibakezelés ablak megjelenése
    1.11. Csupa helyes adatot adok meg regisztrációkor
Elvárt: Megjelenik az alkalmazás fő oldala, a játékterem.
    1.12. Csupa helyes adatot adok meg bejelentkezéskor
Elvárt: Megjelenik az alkalmazás fő oldala, a játékterem

2. Játékterem
    2.1. Bejelentkezve a játékterem oldalán várakozás
Elvárt: Néhány másodperc múlva megjelennek az éppen aktuális kihívások
    2.2. A játékterem oldalon megnyomni a „Játék indítása világossal” gombot
Elvárt: Megnyílik egy új játszmaablak, a sötét figurákkal a teteje felé és várakozik ellenfélre.
    2.3. A játékterem oldalon megnyomni a „Játék indítása sötéttel” gombot
Elvárt: Megnyílik egy új játszmaablak, a világos figurákkal a teteje felé és várakozik ellenfélre.
    2.4. A játékterem oldalon linkre kattintva csatlakozom egy játékoshoz világossal.
Elvárt: Megnyílik a játszmaablak, kiírja az ellenfél nickjét és engedélyezve van a lépés vezérlő gomb

2.4. A játékterem oldalon linkre kattintva csatlakozom egy játékhoz sötéttel.
Elvárt: Megnyílik a játszmaablak, kiírja az ellenfél nickjét és engedélyezve van a lépés vezérlő

2.5. A játékterem oldalon rákattintok a „Felhasználó adatai gombra”
Elvárt: Megnyílik a felhasználó adatainak megtekintése. Ezen az ablakon egy formon szerepel a felhasználó adatai

3. Játszma

3.1 A játszma ablakban, ha nem az adott játékos lép akkor a vezérlő gombok közül egyik sem engedélyezett.
Elvárt: Igaz

3-2- A játszma ablakban formailag rossz lépést adunk meg
Elvárt: Hibás lépés figyelmeztető ablak, új kísérlet lehetséges

4. Figurák lépéseinek próbálása

4.1. gyalog lépéseinek kipróbálása, többféle állásban, en-passant is ellenőrizendő.
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.2. A huszár lépéseinek kipróbálása, többféle állásban.
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.3. A futó lépéseinek kipróbálása, többféle állásban.
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.4. A bástya lépéseinek kipróbálása, többféle állásban.
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.5. A vezér lépéseinek kipróbálása, többféle állásban.
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.6. A király lépéseinek kipróbálása, többféle állásban.
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.7. A sáncolás kipróbálása, többféle állásban (a király vagy a bástya már lépett, a király sakkban van, a közbenső mezők támadva vannak).
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.8. A gyalogátváltozás kipróbálása
Elvárt: a jó lépéseket megteszi, a hamisak helyett pedig figyelmeztetés jön és újabb próbálkozás

4.9. A királlyal megpróbálok sakkba lépni.
Elvárt: Hibás lépés üzenet.

4.10. Figura ellépésére a király sakkban marad.
Elvárt: Hibás lépés üzenet.

4.12. Játszma feladás kipróbálása.
Elvárt: Játszma befejeződése a megfelelő eredménnyel.

4.13. Döntetlen ajánlás és döntetlen elfogadás kipróbálása.

Elvárt: Játszma befejeződése '½' eredménnyel.

5. Felhasználó adatainak változtatása

5.1. Felhasználók adatai ablakban megpróbálom módosítani a felhasználó nevet

Elvárt: Sikertelen a kísérlet.

5.2. Anélkül próbálok menteni, hogy jelszót adnék meg.

Elvárt: Nem sikerült a mentés figyelmeztetés.

5.3. Kétszer nem ugyanazt a jelszót adom meg.

Elvárt: Nem sikerült a mentés figyelmeztetés.

5.4. Üresen hagyom az e-mail cím mezőt és úgy próbálok menteni.

Elvárt: Nem sikerült a mentés figyelmeztetés.

5.5. Helyesen töltöm ki a változtatásokat.

Elvárt: Sikerült a mentés üzenet

5.6. Sikeresen megváltoztattam a jelszót és utána a régi jelszóval akarok belépni

Elvárt: Sikertelen belépés.

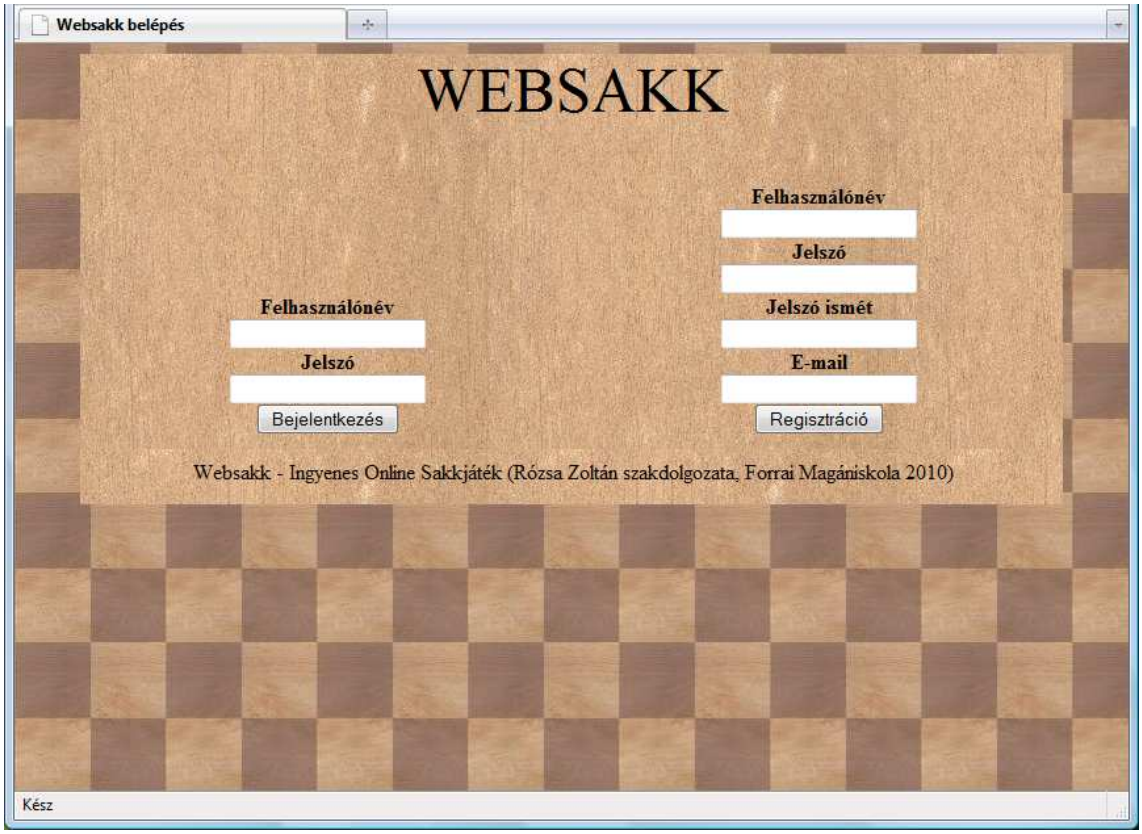
5.7. Sikeresen megváltoztatom a jelszót és megkísérlem a belépést az új jelszóval.

Elvárt: Sikeres belépés.

IV. Felhasználói dokumentáció

1. Felhasználói útmutató

A WEBSAKK egy online sakk játékfelület, ahol egymástól távol lévő játékosok mérhetik össze tudásukat a sakkjátékban.
Ahhoz, hogy valaki játékot kezdhessen ezen alkalmazás segítségével, először regisztrációra van szükség. A regisztráció roppant egyszerűen zajlik. A leendő játékosnak csak meg kell nyitnia egy böngészőben az alkalmazás kezdőlapját, ahol a képernyő jobb oldalán kitöltheti a regisztrációs űrlapot. A kezdőlap az alábbi ábrán látható.



IV.1.1. ábra az alkalmazás kezdőlapja

A regisztrációhoz választani kell egy felhasználónevet (ezt a nevet fogják látni a játékostársak), egy jelszót (amit kétszer azonosan kell begépelni regisztrációkor a biztonság kedvéért), valamint egy e-mail címet. Ezután ha megnyomjuk a

„Regisztráció” gombot, akkor ha helyesen töltöttük ki az adatokat akkor a regisztrációnk tárolása mellett azonnal be is léptet a játékba. Ha valamit elrontottunk (pl.: már van ilyen felhasználónév, vagy a kétszer beírt jelszó nem egyforma stb.), akkor az alábbi képernyőn értesít minket a rendszer a bekövetkezett hibáról. A hibajelző képernyőn láthatunk egy linket amivel visszatérhetünk a kezdőoldalra ismét megkísérelni a regisztrációt. Ha van már regisztrált felhasználói nevünk, akkor a kezdő képernyő bal oldalán látható található mezőkbe kell beírni a felhasználói nevet és jelszót a bejelentkezéshez és utána meg kell nyomni a „bejelentkezés” gombot. Ha helyes adatokat adtunk meg akkor beléphetünk a játékba, de ha valahol hibázunk (pl.: nem létező felhasználói nevet adunk meg, vagy eltévesztjük a jelszó beírását) akkor a regisztrációnál már megismert hibajelző ablak fogad minket. Itt használhatjuk a linket az ismételt bejelentkezéshez.

IV.1.2 ábra hibaüzenet bejelentkezéskor

Sikeres bejelentkezés után az alább ábrán látható „játékterembe” jutunk. Ez az alkalmazás fő oldala. Innen tudunk új játékot indítani. Kiválaszthatjuk, hogy ha új kihívást indítunk, világossal vagy sötéttel játszanánk.
Erre szolgál az „Új játék indítása világossal”, illetve az „Új játék indítása sötéttel” gomb. De nem csak új kihívást tehetünk itt meg, hanem mint az ábrán is látható, az aktuális kihívások táblázatos formában megjelenek a képernyőn és linkek segítségével csatlakozhatunk a még el nem fogadott kihívásokhoz. Értelemszerűen olyan színnel, amely szabadon volt hagyva.
Ha új játékot indítunk, akkor az alkalmazás nyit egy új ablakot amelyen a játékfelület látható. A játékfelületen egy alapállásba állított sakktábla látható (azzal a színnel a képernyő alsó része felé, amilyen színű bábokat mi vezetünk), valamint a vezérlő gombok és mezők. Új játék indításakor először ellenfélre kell várnunk, ekkor egyetlen gomb sincs engedélyezve a felületen. Ha valaki csatlakozik a játszmánkhoz, azt onnan vehetjük észre, hogy a tábla felső része fölé kiírja az ellenfelünk felhasználónevét és ha mi játszunk világossal akkor elérhetővé válik a lépés gomb

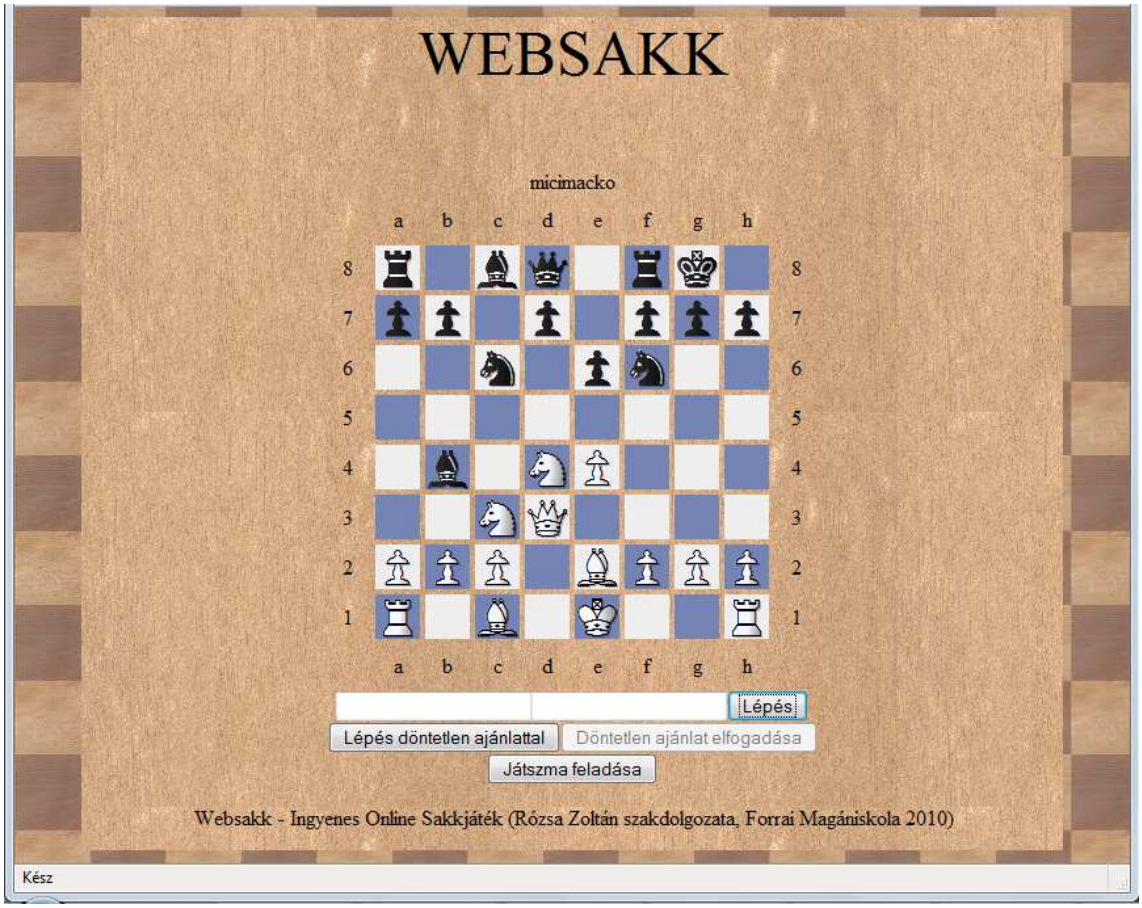
IV.1.3. ábra kihívások listája

Az alábbi ábrán látható az alapállás, még ellenfél nélkül.

IV.1.4. ábra új játék indítása világossal

Ha elfogadjuk valamelyik kihívást, akkor szintén megnyílik a játékfelület, de akkor már azonnal megjelenik az ellenfelünk felhasználóneve és ha világossal játszunk azonnal léphetünk is.

Lépni úgy lehet, hogy a sakktábla alatt látható két szövegmezőbe beírjuk a lépésünk koordinátáit. Az első szövegmezőbe a lépés kezdő koordinátáját (pl.: c4). A koordináták beírásához segítséget nyújt a tábla szélén látható sorszámozás illetve az oszlopok betűjele is.
Ha beírtuk a lépést akkor megnyomhatjuk a „Lépés” gombot, vagy a „Lépés döntetlen ajánlattal” feliratú gombot.



IV.1.5. ábra játék közben

A lépés ha helyesen írtuk be, mindkét esetben végrehajtódik és kijelzésre kerül nem csak a saját tábláján a játékosnak, hanem távoli ellenfele tábláján is. Ha a lépést döntetlen ajánlattal küldtük el akkor amellett, hogy a lépések letárolódnak még egy döntetlen ajánlat is eljut az ellenfélhez, aki mindezt onnan veszi észre, hogy a „Döntetlen ajánlat elfogadása” gomb engedélyezetté válik. Ilyenkor lépés helyett ezt megnyomva a játszma véget ér és kiírásra kerül a játszma végeredménye. Ez látható a IV.1.5. ábrán.

Hasonlóképpen tudjuk a kilátástalannak ítélt játszmánkat feladni. Meg kell várjunk, amíg lépésre nem kerülünk és akkor lépés helyett megnyomni a „Játszma feladása” gombot. Ekkor minden kezelőszerv letiltódik és a sakktábla fölé kiíródik a végeredmény.

Minden lépés esetén az első koordináta a lépést kezdőpontját jelenti, tehát azt a pontot amelyről a lépni kívánó figurával lépünk. Sáncolás esetén a király lépését kell begépelnünk a rendszerbe. Ha gyalogunk eléri az ellenfél alapsorát akkor valamilyen figurát fel kell venni. Gyalogot vagy királyt értelemszerűen nem lehet felvenni, marad

tehát négyfajta figura. Gyalog bevitelnél a lépés második koordinátájának végére egy betűt kell írni, hogy milyen figurát szeretnénk felvenni a gyalog helyett (h=huszár, f=futó, b=bástya, v=vezér). A rendszer ismeri az úgynevezett „en_passant” lépést is, amikor a gyalog alapállásból kettőt lép és ezzel átlépi egy ellenséges gyalog ütőkörét, az leüthető, de csak abban az egy lépésben (Az „en-passant” lépésről többet a feladat specifikációban olvashatunk). A rendszer a mattot nem jelzi ki, de sakkba nem enged lépni, úgyhogy a matt állás után úgysem lehet tovább játszani, mert a vesztes fél folyton sakkba lépne.

IV.1.6. ábra döntetlen végeredmény

Az IV.1.7. ábrán egy hibás lépésre adott figyelmeztető üzenet látható. Mindenféle hibás lépésre hasonlóképpen reagál a rendszer. Hibás lépés után megnyomva az „OK” gombot a figyelmeztető üzenet eltűnik és meg lehet próbálkozni újra egy jó lépés begépelésével.

Az alkalmazás fő ablakában a játszma indító gombok mellet található egy harmadik nyomógomb is „Felhasználói adatok” felirattal. Ezt megnyomva a felhasználó megtekintheti a saját regisztrációkor megadott adatait.
Ez a képernyő a IV.1.8. ábrán látható. Ebben az ablakban nem csak megtekintheti a játékos a saját felhasználói adatait, hanem módosíthatja is. Itt is kétszer kell megadni a jelszót és ezek egyezése esetén illetve ha egyik jelszó mezőt sem hagyjuk üresen, akkor a „Mentés” gombot megnyomva elmentődnek az adatbázisba a megadott adatok. Ha rosszul adunk meg valamely adatot, akkor a „Mentés” gomb alatt megjelenik a figyelmeztető felirat, hogy nem sikerült a mentés.

IV.1.7. ábra egy hibás lépésre adott üzenet.

IV.1.8. ábra felhasználói adatok megtekintése

IV.1.9. ábra sikeresen módosított felhasználói adatok

2. Telepítési útmutató

Jelen alkalmazásunkat PHP támogatással rendelkező webszerverre telepíthetjük és fontos, hogy a szerveren működjön egy MySQL adatbázis-kezelő is.

A mellékelt CD-n megtalálható a teljes webalkalmazás, a szükséges adatbázist létrehozó script és jelen dokumentáció is PDF formátumban.
A CD-n kívül a teljes, friss alkalmazás letölthető az ftp2.freeweb.hu FTP szerverről (login: micimaci, password: pansoft).
Valamint az alkalmazás élesben is kipróbálható a www.micimaci.freeweb.hu oldalon.

A mellékelt CD könyvtárstruktúrája az alábbi:

```
..\
        \websakk
                \controllers
                        gamelistajax.php
                        gamescontroller.php
                        moveajax.php
                        opponentsajax.php
                        opponentvajax.php
                        remiajax.php
                        resignajax.php
                        userscontroller.php
                        waitmoveajax.php
                \css
                        websakk.css
                \img
                        …
                        …
                \js
                        game.js
                        position.js
                \models
                        gamemodel.php
                        movemodel.php
                        positionmodel.php
                        rule.php
                        usermodel.php
                \views
                        chessview.php
                        gamesview.php
                        hibaview.php
                        userview.php
                        view.php
                dbbase.php
                index.html
        \document
                szakdolgozat.pdf
```

\telepites
datamodel.sql
telepites.txt

A telepites.txt fájlban megtalálható jelen leírás.

A telepítést a következőképpen kell elvégezni:

- Létrehozni a MySQL adatbázis kezelőn egy tetszőleges nevű adatbázist és ezen az adatbázison lefuttatni a CD-n a telepítés könyvtárban található datamodel.sql scriptet.
- A CD-n található websakk mappa tartalmát változatlan struktúrában felmásolni a webszerverre.
- A webszerveren az alkalmazás gyökérkönyvtárában található dbbase.php fájlban a következő változtatásokat tenni:

21. sor hostnév beírása az idézőjelek közé
22. sor adatbázisnév beírása az idézőjelek közé
23. sor felhasználónév beírása az idézőjelek közé
24. sor jelszó beírása az idézőjelek közé

Példaként álljon itt az alábbi:

```
class DBMysql  {
    protected $dbhost="micimaci.sql.freeweb.hu";
    protected $dbname="micimaci";
    protected $user="micimaci";
    protected $password="pansoft";
```

Egyéb teendőnk nincs is a telepítéssel kapcsolatban.

V. Felhasznált irodalom

- Ian Sommerville: Szoftver rendszerek fejlesztése (Panem 2002)
- Sike Sándor – Varga László: Szoftvertechnológia és UML (ELTE Eötvös kiadó 2001)
- Raffai Mária: UML2 - Modellező nyelvi kézikönyv 4. (Alexander 2005)
- Matt Zandstra: Tanuljuk meg a PHP5 használatát 24 óra alatt (Kiskapu 2004)
- George Schlossnagle: PHP fejlesztés felsőfokon (Kiskapu 2004)
- Joshua Eichorn: Az AJAX alapjai (Panem 2008)
- Kris Hadlock: Webalkalmazások fejlesztése AJAX segítségével (Kiskapu 2007)
- Nagy Gusztáv: Web programozás I. (internetes jegyzet *http://nagygusztav.hu)*
- A sakkjáték szabályai a Wikipédiáról: **http://hu.wikipedia.org/wiki/Sakk**