Eötvös Loránd Tudományegyetem
Informatikai Kar
Média- és Oktatásinformatika Tanszék

QR játékok

Pluhár Zsuzsa
tanársegéd

Simon Péter (SIPNAAI.ELTE)
Programtervező Informatikus
nappali tagozat

Budapest, 2011.

A projekt az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósul meg,
a támogatási szerződés száma TÁMOP 4.2.1./B-09/1/KMR-2010-0003.

Téma bejelentő: a szakdolgozat bekötve kell, hogy tartalmazza a kitöltött és jóváhagyott (az Informatikai Kar dékánja által aláírt) Szakdolgozat-téma bejelentőt. **témabejelentőt**. Ennek a lapnak a helyére kell bekötni.

* * *

Tartalomjegyzék

1. fejezet

A honlap célja

1.1. QR kód

A hagyományos vonalkódot, más néven egy dimenziós kódot, 1974-ben kezdték el használni az Egyesült Államok, Troy városában. Magyarországon először 1984-ben, a Skála áruházban találkozhattak vele a vevők. Azóta a számítástechnika fejlődésének köszönhetően, már a világ minden pontján megtalálhatóak a legkülönbözőbb felhasználási formában. Az árucikkek tárolásánál leggyakrabban a kódok 13 számjegynyi adatot rögzítenek. Az első három számjegy tárolja a gyártási országot (például Magyarország esetében: 599), 4-12. számjegy, magával az áruval kapcsolatos információt tartalmaz. Míg az utolsó számjegy az előtte lévő adatokból kerül kiszámolásra. [7]

Hosszabb adatok tárolására az úgynevezett kétdimenziós kódok alkalmasak. Két legelterjedtebb formája a Data Mátrix, illetve a QR kód.

A QR kódot 1994-ben fejlesztett ki a Denso-Wave japán cég. Nevét a Quick Response (gyors válasz) rövidítésből kapta.

A kód nagy előnye, hogy a tartalom megsérülése esetén lehetőség van a CD-knél is használt Reed-Solomon hibajavítást alkalmazni. Tömörítéstől függően, a kódnak a 7–30% közötti sérülését képes javítani.

Széleskörű elterjedését a gyors és szinte bármilyen szögből történő leolvasás, valamint az olcsó, kamerával rendelkező készülékek megjelenése tette lehetővé. [5]

1.2. QR játék

A játék a TÁMOP 421B kutatási projekthez kapcsolódóan, a Média- és Oktatásinformatika Tanszék segítségével jött létre.

Az ELTE IK tanárképzésének Telementorálás kurzusain, 2002 óta állítanak össze a hallgatók olyan „Kihívás játékokat", melynek célja az IKT (Információs és Kommunikáció Technológia) eszközök valóban eszközként kezelése, valamint ezeknek a játékos formában történő megértése és elsajátítása. 2011-től ennek keretében valósul meg a QR játék.

A QR játék során a játékosok különböző kérdések megválaszolásával gyűjthetnek pontokat. A kérdést egy-egy QR kód leolvasásával, majd a kapott URL megnyitásával szerezheti meg a játékos. A játékban a QR kód szerepe kettős. Egyrészről kényelmesebb megoldás a kérdés elérésére, mivel a játékosnak nem kell a böngészőbe begépelni az URL-t. Másrészről biztosítja, hogy a kérdésekre csak akkor tudjon válaszolni, ha végigjárta azokat a helyszíneket ahová a QR kódot elhelyezték.

1.3. A honlap funkciói

A honlap szerepe két részre osztható. A játékos itt kapja meg a kérdéseket, a kérdés megválaszolása után itt nézheti meg az eddig megválaszolt kérdéseit és az ezzel elér pontjait. Illetve az adminisztrátorok itt hozhatják létre, szerkeszthetik a kérdéseket, valamint módosíthatják a játék egyéb funkcióit.

2. fejezet

Felhasználói dokumentáció

2.1. Oldal felépítése

Az oldal felépítését szemlélteti a 2.1-es ábra.

2.1. ábra. Az oldal felépítése

1: Aktuális oldal címe.

2: Menüpontok. Az első oszlopban szerepelnek a belépett felhasználóval kapcsolatos menüpontok. A második oszlopban találhatóak, a játék indulása előtt

elvégzendő teendőkkel kapcsolatos menüpontok. A harmadik oszlopban a játék közben szükséges funkciókat érhetjük el. A negyedik oszlopban pedig, a játék utáni teendőkhöz kapcsolódó linkek kaptak helyet.

3: Itt választhatjuk ki, hogy melyik játékhoz tartozó adatokat szeretnénk látni.

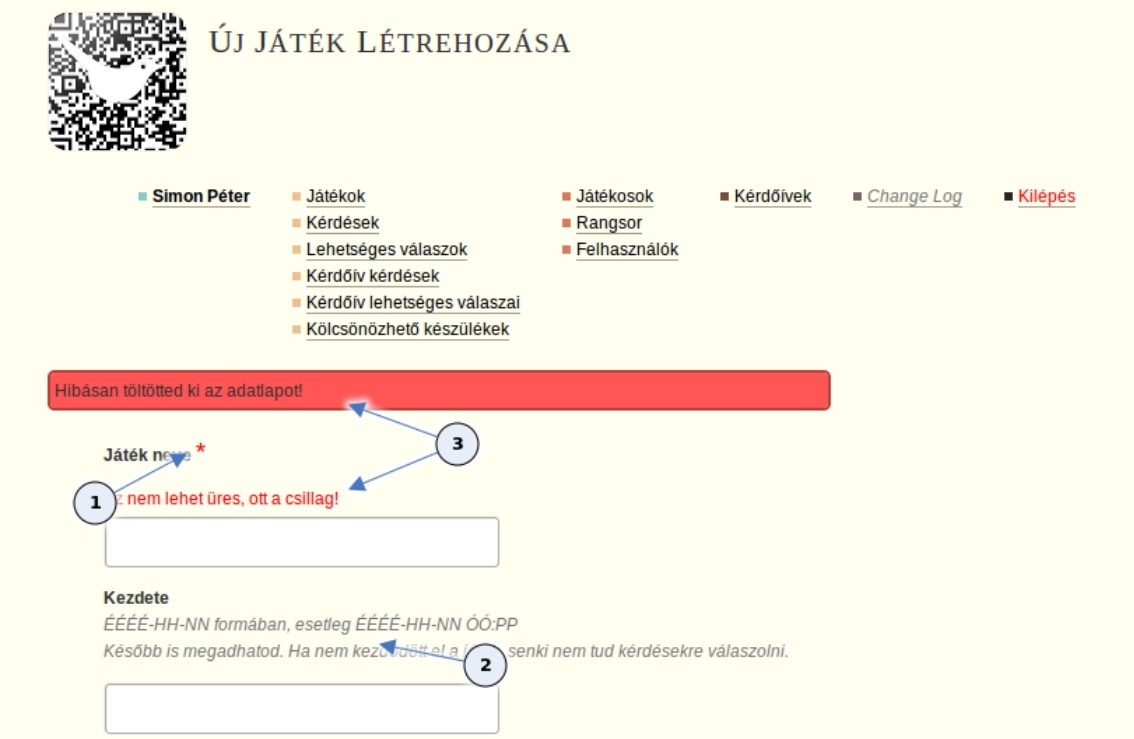
4: Bővebb információ az adott oldalon látható adatokkal kapcsolatban.

5: Aktuális oldalhoz tartozó adatok.

6: Az egyes sorokhoz tartozó menüpontok.

7: Az egér mutató pozíciójától függően, bővebb információ jelenhet meg az adott értékkel kapcsolatban.

Az űrlapok felépítését a 2.2-es ábrán láthatjuk.



2.2. ábra. Űrlapok

1: A kötelezően kitöltendő mezőket piros csillag jelöli.

2: A mező neve alatt további információk jelennek meg, eligazítást adva a beírt adatra vonatkozóan.

3: Hibásan kitöltött adatlap esetén, egy általános figyelmeztetést ír ki az oldal tetejére és egy másikat a mező fölé, a hiba pontos leírásával.

2.2. Felhasználók jogosultságai

A felhasználókat négy különböző csoportba lehet osztani.

- Játékos: aki a játékot játsza.
- Játékvezető: feladata a játékosok regisztrálása, kölcsönözhető készülékek és a kölcsönzés nyilvántartása, kérdőívek kitöltése.
- Adminisztrátor: feladata a kérdések létrehozása és szerkesztése.
- Super User: feladata a játékok, valamint további Játékvezetők, Adminisztrátorok, illetve Super User-ek felvétele és törlése.

A Játékosokon kívül, egyik felhasználói csoport sem vehet részt a játékban.

2.3. Játékok

Játékokat létrehozni, szerkeszteni, törölni a *Játékok* menüben lehet. Egyszerre több játék is játszható, de egy játékos csak egy játékban vehet részt. Másik játékba való regisztráláskor a rendszer, a korábbi játékát befejezettnek nyilvánítja.

2.3.1. Játék kezdése és befejezése

Játékot elkezdeni vagy az *elkezd* linkkel lehet, vagy meg lehet adni a szerkesztés menüpontban a kezdés dátumát, ÉÉÉÉ-HH-NN vagy ÉÉÉÉ-HH-NN ÓÓ:PP formában.

Ha a játék nincs folyamatban, a játékosok nem tudják lekérdezni a kérdést (így arra válaszolni sem tudnak).

Folyamatban lévő játék kérdéseit, illetve a kérdésekhez tartozó lehetséges válaszokat, nem lehet módosítani.

Játékot befejezni a *befejez* linkkel lehet. Ilyenkor megszűnik minden kapcsolat a játékosok és a játék között.

2.3.2. Játék újrakezdése

Játékot újrakezdeni az *újrakezd* linkkel lehet. Figyelni kell rá, hogy a játékosokat ebben az esetben újra a játékhoz kell rendelni, mivel a játék befejeztével megszűnt a kapcsolat játék és játékos között. Amennyiben a publikus regisztráció engedélyezve van az adott játéknál, ez a hozzárendelés automatikusan megtörténik, amikor a játékos leolvas egy játékhoz tartozó QR kódot. A játék újrakezdésével a versenyzők nem vesztik el az addig megszerzett pontjaikat.

2.3.3. Egyéb funkciók

Az *Automatikus lezárás* mező kitöltésével lehet szabályozni, hogy egy játékosnak, miután hozzá lett rendelve a játékhoz (vagy saját maga regisztrált be vagy a játékvezető rendelte hozzá) mennyi ideje a van kérdésekre válaszolni.

Publikus regisztráció engedélyezésével be lehet állítani, hogy a játékvezető segítsége nélkül is be tudjon regisztrálni egy játékos. A regisztrációhoz szükséges linket egy (a játékhoz tartozó) QR kód leolvasásával kaphat. Regisztrálás után a játékos automatikusan hozzá lesz rendelve az aktuális játékhoz.

2.4. Kérdések és lehetséges válaszai

2.4.1. Kérdések létrehozása és szerkesztése

Kérdések adminisztrálására a *Kérdések* menüpont ad lehetőséget. Továbbá itt nyílik mód, QR kód generálására egy-egy kérdéshez. Az URL, amelyen keresztül a kérdés elérhető, a *link* mezővel szerkeszthető. Kezdetben a rendszer felajánl egy egyedi véletlen karaktersorozatot, ha ezt meg szeretnénk változtatni ügyelni kell rá, hogy ne egyezzen meg, más játékban szereplő kérdések *link* mezőjével. Ha mégis megegyezik a rendszer figyelmeztetést ír ki.

Amennyiben azt szeretnénk, hogy egy QR kód leolvasásával több kérdés is megjelenjen, minden megjeleníteni kívánt kérdés *link* mezőjét azonosra kell állítanunk.

Minden kérdéshez és lehetséges válaszhoz feltölthetünk jpg, png vagy gif formátumú, 1,5Mb-nál nem nagyobb képeket. Feltöltéskor két dologra kell ügyelnünk: ha már létezik egy megegyező nevű és kiterjesztésű fájl, akkor a régebbi felül lesz írva, illetve, hogy a rendszer feltöltéskor nem méretezi át a képeket. Amennyiben azt szeretnénk, hogy mobil eszközön is könnyen átlátható legyen egy képpel rendelkező kérdés, érdemes 350 pixelnél nem nagyobb fájlokat feltöltenünk.

2.4.2. Kérdések és lehetséges válaszainak kapcsolata

A kérdések négy típusba sorolhatóak és a játékos válasz adása esetén is négyféleképpen kerülnek kiértékelésre.

Az adott válaszok kiértékelése minden esetben automatikusan történik, ezért szükséges mindig valamely csoportba besorolni a kérdést, illetve legalább egy lehetséges választ hozzárendelni.

Egy helyes válasz esetén a kérdésre, csak egy választ tudunk megjelölni. A kapott pontunk annyi lesz, amennyit a megjelölt lehetséges válasz, *pont* mezőjében beállítottunk (lehet negatív érték is).

Több helyes válasz esetén a kapott pontunk a megjelölt lehetséges válaszok, *pont* mezőjének összege lesz. Érdemes a rossz válaszoknak negatív értéket adni, így az összes válasz megjelölésével nem lehet maximális pontot szerezni.

Dátum válasz esetén a játékos kapott pontja az első megegyező lehetséges válasz, *pont* mezőjével egyenlő (ha van ilyen, különben nulla pontot kap). Dátum megadásának módja: ÉÉÉÉ vagy ÉÉÉÉ-HH-NN. A várt formátumra a rendszer, mind a játékost, mind a kérdés rögzítőjét figyelmezteti.

Szöveges válasz esetén a játékos kapott pontja az első megegyező lehetséges válasz, *pont* mezőjével egyenlő. A válaszban a kis-, nagybetű különbségek, valamint a válasz előtt vagy után lévő fehér szóközök (helyköz, új sor vagy tabulátor) nem számítanak.

2.5. Kérdőívek

Minden játékhoz létrehozható egy kérdőív, tetszőlegesen sok kérdéssel. A kérdőív kérdések és hozzájuk tartozó lehetséges válaszok létrehozásának módja megegyezik, a kérdések és lehetséges válaszok létrehozásával, azzal a különbséggel, hogy itt nincs értelme a *szöveges válasz* típusú kérdéshez, lehetséges választ felvennünk.

Kérdőívet kitölteni játékos nem tud. Erre a Játékvezetőnek, Adminisztrátornak vagy Super User-nek van lehetősége a *Rangsor* menüpontban.

Érdemes minden játékhoz kérdőívet létrehozni, mert fontos statisztikai eredményeket kaphatunk, melyekkel javíthatunk a későbbi játékok színvonalán.

2.6. Kölcsönözhető készülékek

Készülékeket, melyeket a játékosok rendelkezésére kívánunk bocsátani a *Kölcsönözhető készülékek* menüpontban tudjuk számon tartani. Mobil telefon regisztrálásánál szükséges a készülék IMEI számának megadása mellyel, a későbbiek során egyértelműen azonosítani lehet. Ezt a 14 vagy 15 számjegyből álló számot, a `*#06#` beírásával kérdezhetjük le. Kikölcsönzés esetén a játékosnak egyéb (a regisztrációhoz nem szükséges) személyes adatokat kell megadni, úgy mint: állandó lakcím és személyi szám.

2.7. Felhasználók

Játékosokat létrehozni és szerkeszteni a *Játékosok* menüben tudunk, míg Játékvezetőket, Adminisztrátorokat, illetve Super User-eket a *Felhasználók* menüpontban. Beregisztrált játékos törlésére nincs mód.

2.8. Rangsor

A *Rangsor* menüpont ad lehetőséget a játék állásának nyomon követésére. A rangsor az elért pontok alapján csökkenő sorrendbe rendezett, de előrébb kerülnek azok a játékosok akik negatív ponttal rendelkeznek, mint azok akik még egy kérdésre sem válaszoltak (jól vagy rosszul). Két megegyező pontszámú játékos helyezése is megegyezik. Egyéb összehasonlítást (pl.: gyorsaság) a rendszer nem tesz.

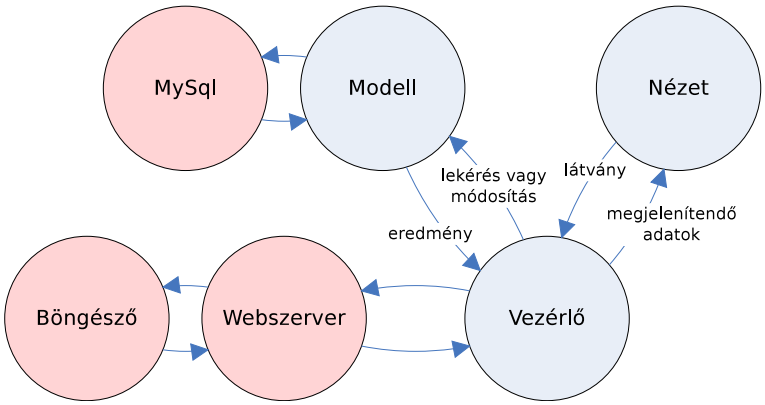
Továbbá, itt tudjuk az ajándék átvételét regisztrálni vagy, ha a játékhoz tartozik kérdőív, akkor azt kitölteni valamelyik játékos nevében. A kitöltött kérdőívek megjelenítésére a *Kérdőívek* menüpontban van lehetőségünk.

3. fejezet

Fejlesztői dokumentáció

3.1. Program szerkezete

A rendszer az MVC (Model *(modell)*, View *(nézet)*, Controller *(vezérlő)*) programozási modell alkalmazásával készült. A szerkezeti minta használatának nagy előnye, hogy a felhasználók elé nagy mennyiségű adatot tároló alkalmazásoknál, könnyen szét lehet választani az adatokhoz és a felhasználói felülethez tartozó részeket. Az előbb említett két rész összekötéséért pedig, egy közbülső összetevő, a vezérlő réteg felel. A rétegek szétválasztásával az MVC modell csökkenti a szerkezeti bonyolultságot és növeli a rugalmasságot. [4]



3.1. ábra. MVC modell működése

Ebben a szemléletben a program három fő részre oszlik:

- Modell réteg, mely az adatbázisban tárolt adatokhoz fér hozzá és felel annak konzisztenciájáért.

- Nézet réteg határozza meg az adatok megjelenítését és felületet biztosít a felhasználónak azok módosítására.
- Vezérlő réteg reagál a felhasználó tevékenységére, kommunikál a modell réteggel, majd előállítja a nézet réteg számára a kívánt adatokat.

3.1.1. Inicializálás

Minden oldal lekérés teljesítésekor a *libs/init.php* fut le, melynek három fő feladata van:

- Példányosítja a *Registry* osztályt, melynek egyetlen feladata, hogy tárolja azokat az adatokat és példányosított osztályokat, melyekre más osztályoknak is szüksége lehet.
- A kapott URL-ből meghatározza, a *Router* osztály segítségével, hogy melyik vezérlő osztályt kell példányosítani és melyik metódusát kell majd meghívni.
- Az *Authentication* osztály segítségével, azonosítsa a felhasználót és eldöntse, hogy van-e joga meghívni a kívánt metódust.

3.1.2. Az URL szerkezete

Minden oldal lekéréskor az *index.php* fájlt adja vissza a webszerver. Ez az egyetlen belépési pont. Az URL-ek a következőképpen néznek ki:

```
http://qrportal.elte.hu/index.php?url=adatlapom/belepes
```

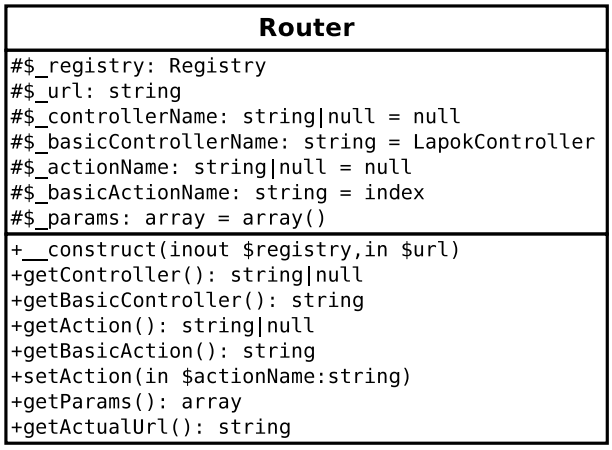
A *.htaccess* fájlok segítségével viszont, a felhasználó csak a következőt látja:

```
http://qrportal.elte.hu/adatlapom/belepes
```

Ahol az első paraméter (most az adatlapom) az elérni kívánt vezérlő osztályt határozza meg, míg a második paraméter (most a belépés) a vezérlő osztály egy metódusát. Ezeket a metódusokat akcióknak hívjuk.

Azt, hogy egy URL esetében pontosan melyik vezérlő, melyik akcióját kell meghívni, a *Router* osztály mondja meg.

Példányosításkor átadva az URL-t, lehetőségünk van a *getController* metódus segítségével lekérdezni, hogy melyik vezérlőt kell meghívnunk. Amennyiben nincs meghatározott vezérlő, a *getBasicController* segítségével lekérdezhetjük, melyik az alapértelmezett vezérlő.



3.2. ábra. Router osztály

A meghívandó metódus meghatározására a *getAction* és a *getBasicAction* tagfüggvények használhatóak.

Lehetséges, hogy további paramétereket tartalmaz az URL. Ezeket az extra adatokat a *getParams* függvény segítségével érhetjük el.

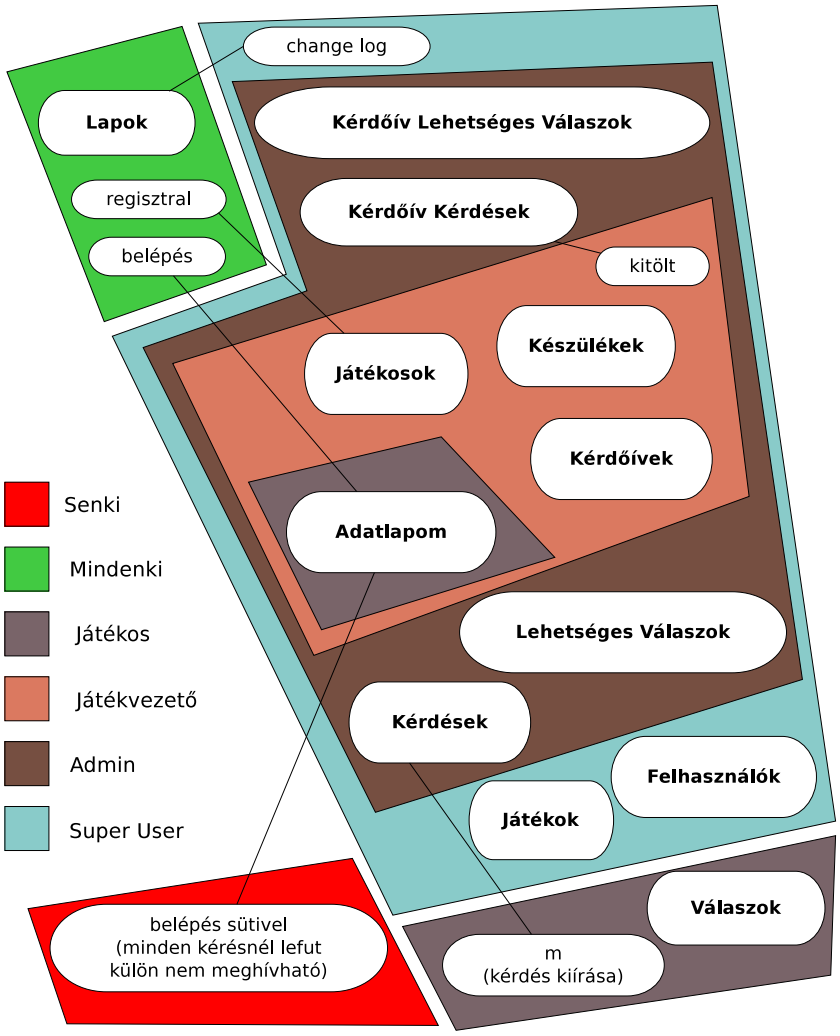
3.1.3. Autentikáció

Az autentikáció feladata a felhasználó azonosítása és jogainak meghatározása.

Csoportok jogainak beállítása

A felhasználói csoportok jogait az *aco* és az *acl* adatbázis táblák segítségével határozhatjuk meg. Az előbbi tábla tartalmazza az elérni kívánt objektumokat (Access Control Objects) egy kétszintes fa gráffal reprezentálva, melyben a gyökér jeleni az összes objektumot, az első szinten a vezérlők, míg a második szinten az akciók (vezérlők egy-egy tagfüggvénye) találhatóak. Míg az utóbbi (Access Control List) tartalmazza a szabályokat, melyek a következőképpen néznek ki:

- *csoport_id* mező tárolja, hogy melyik csoportra vonatkozik az adott szabály (NULL érték esetén az összes csoportra vonatkozik).
- *aco_id* mező határozza meg, hogy melyik elérni kívánt objektumra érvényes az adott szabály. Értéke lehet NULL, ez esetben az összes objektumra érvényes, lehet egy konkrét vezérlő vagy egy vezérlő egy akciójának azonosítója.
- *muvelet* mező értéke vagy M (megtilt) vagy E (engedélyez).

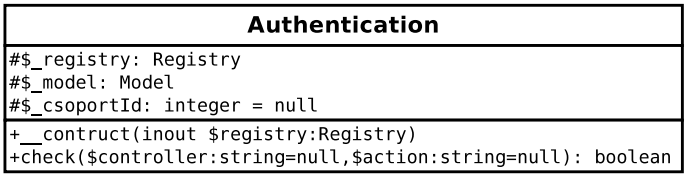


3.3. ábra. Csoportok jogai. Részletesebb leírás található a mellékletek között.

A jelenleg használt beállításokat a 3.3-as ábra szemlélteti. A vastaggal szedett nevek jelentik a vezérlőket, a normál betűtípussal írottak pedig az akciókat. A ki nem írt akciókra ugyanazok a jogok vonatkoznak, mint a hozzá tartozó vezérlőjéhez.

Autentikáció megvalósítása

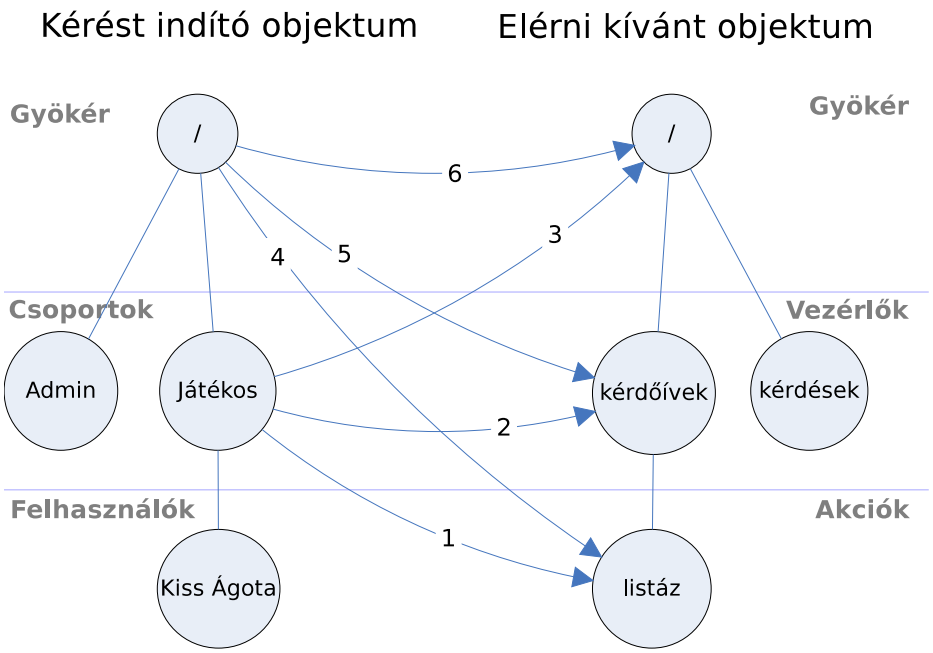
Az autentikáció megvalósítása az *Authentication* osztály segítségével történik.



3.4. ábra. Authentication osztály.

Az osztályban található *check* metódus segítségével lekérdezhetjük, hogy van-e joga a kérést indító felhasználónak elérni az adott objektumot.

Az *acl* táblában meghatározott szabályokat a 3.5-ös ábrán szemléltetett sorrendben vizsgálja meg.



3.5. ábra. Példa az autentikáció működésére.

Amennyiben egyetlen szabályt sem talált, megengedi a felhasználónak az objektum elérését. Ha az adatbázissal való kommunikáció során hiba lép fel, megtiltja a hozzáférést.

Jelszavak tárolása

A jelszavak biztonságos tárolása érdekében a rendszer a következőképpen rögzíti azokat: a *config/config.php* fájlban meghatározott 40 karakteres, véletlen karaktersorozat lesz minden jelszó prefixe. Konkatenálás után a jelszavakat SHA1 algoritmus segítségével kódolva tárolja el a program. Így még akkor is kellő védelmet biztosít a jelszavak megszerzése ellen, ha a támadó birtokában van az adatbázis tartalmának.

Jelszavak tárolásánál az SHA1 hasító algoritmus segítségével, nem olvasható ki közvetlenül a jelszavak tartalma az adatbázisból, míg a prefix használata védelmet nyújt a szivárvány táblák sikeres használata ellen.

Egy szivárvány táblában általában a gyakran használt szavak, több szó összetétele, illetve számjegy prefixel vagy szuffixel előállított szavak valamilyen hasító függvénnyel kódolt értékei találhatóak. Az olcsó tárhelyeknek köszönhetően több millió vagy akár több billió sort is tartalmazhat. Egy ilyen tábla segítségével, amennyiben nem használunk a jelszavak tárolásához prefixet, nagyságrendekkel gyorsabb egy jelszó visszafejtése. [1]

3.2. Modell réteg

3.2.1. Adatbázis felépítése

Az adatbázis szerkezetét a 3.6-os kép szemlélteti.

Jelölések

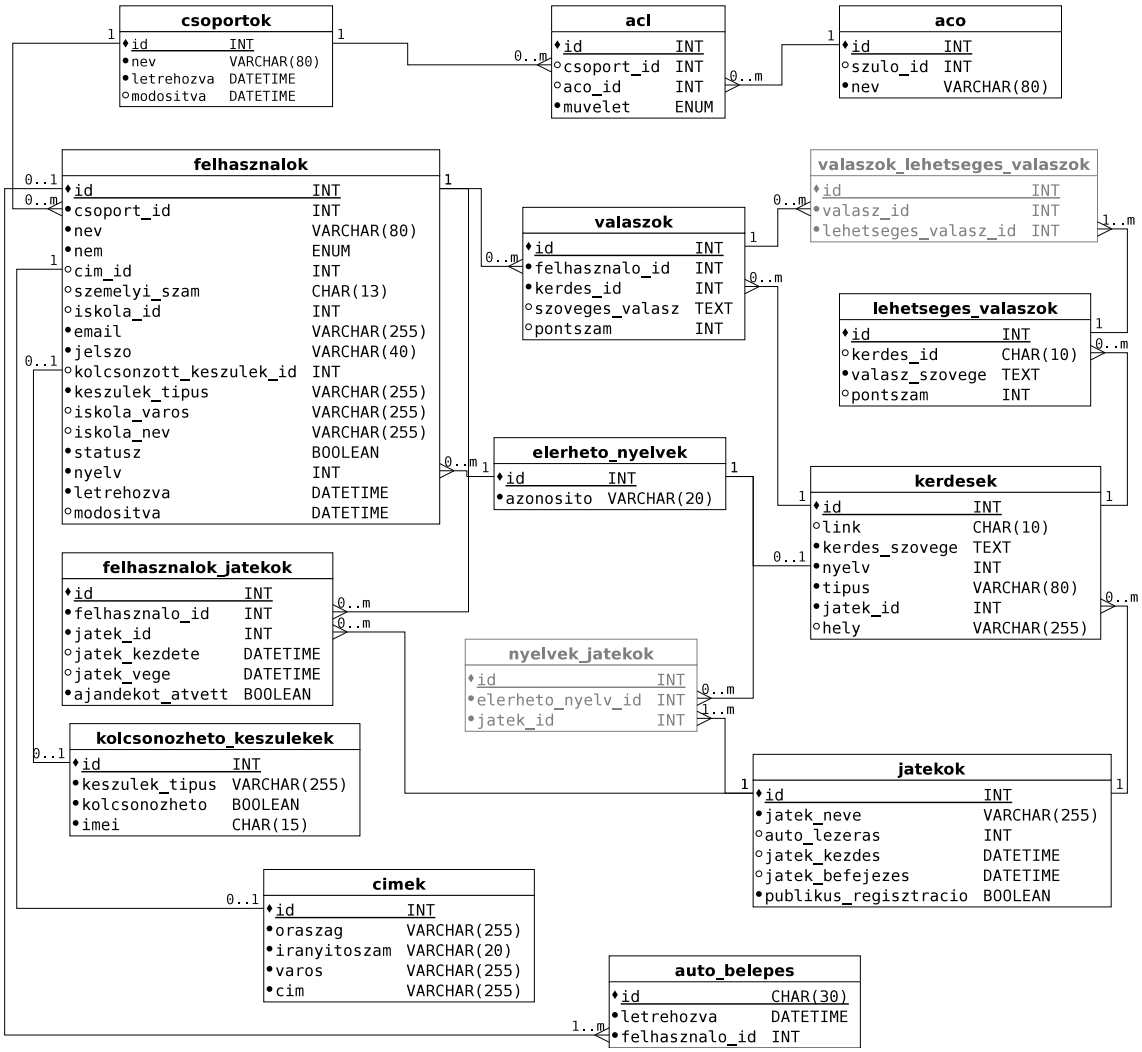
Az egy-több kapcsolatot, melyben a baloldali tábla egy mezőjéhez a jobboldali táblából nulla vagy több mező is tartozhat a következő nyíl jelöli: 

A mezők nevei előtt szereplő teli körök jelölik, hogy a mező értékének megadása kötelező.

Aláhúzott névvel szerepelnek az elsődleges kulcsok, melyeknek értéke minden esetben automatikusan növekszik.

Szürke táblázatokban azok a segéd táblák szerepelnek, melyek a több-több kapcsolat leírására szolgálnak.

Mivel a MySql adatbázis kezelőben logikai típus megadására nincs lehetőség, ezekben az esetekben a mező típusa TINYINT(1), melyben az 1-es érték jelentése igaz, a 0-ás értéké pedig hamis.



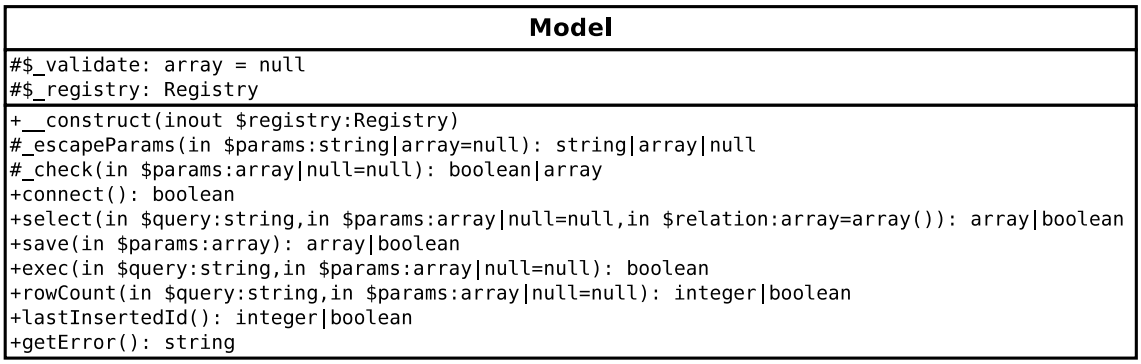
3.6. ábra. Adatbázis szerkezete. Részletesebb leírás található a mellékletek között.

3.2.2. Kommunikáció az adatbázissal

Az adatbázissal való kommunikáció a *libs/model.class.php* fájlban található, *Model* osztályon keresztül történik.

Az osztály példányosításakor lefutó *connect* tagfüggvény, létrehozza a kapcsolatot a MySql adatbázissal, amennyiben ez még nem történt meg. Valamint beállítja a helyes karakterkódolást.

A *select* tagfüggvénnyel tudunk adatokat lekérdezni. Első paraméterében az SQL kérést kell megadni, melyben a szám paramétert *%d*-vel, a szöveges paramétert *%s*-sel kell jelölni és a második paraméterben tömbként átadni a megfelelő sorrendben. Harmadik, nem kötelező paraméterrel lehetséges kapcsolatot definiálni, a lekérdezni kívánt táblával.



3.7. ábra. Model osztály

A *save* metódust az adatok mentésére használjuk. A paraméterben, asszociatív tömbként átadott adatok új sorként kerülnek beszúrásra, ha nincs megadva *id* oszlop, különben meglévő sor módosításának értelmezi a program. Az adatok helyességéért az automatikusan meghívott *check* tagfüggvény felel. A bevitt adatokra vonatkozó elvárásainkat a *$_validate* attribútum tartalmazza, melynek értékét a gyerek osztályok definiálják. Helyes értékkel tér vissza amennyiben minden adat megfelelő volt, asszociatív tömbbel, ha hibát talált. A hiba leírása a visszaadott tömbben, a tábla és a mező nevével indexelt pozícióban található.

Az *exec* tagfüggvénnyel tetszőleges MySql parancs futtatható le, visszatérési értéke pedig, a parancs sikeres lefuttatására utal.

A *rowCount* tagfüggvény egy SQL lekérdezés sorainak számát adja vissza vagy hamis értéket, amennyiben nem sikerült sikeresen lefuttatni a parancsot.

A *lastInsertedId* az utolsó beszúrt, automatikusan növekvő *id* oszlop értékét adja vissza.

A *getError* metódussal pedig, az utolsó MySql hibát kérdezhetjük le.

Védelem az SQL beszúrásos támadás ellen

Egy SQL beszúrásos támadás során a felhasználó, megfelelően összeállított adatokkal ráveszi az adatbázist, hogy olyan kéréseket futtasson le, amik meghaladják a jogosultságait.

A következő lekérésnél kétféle támadás ellen kell védelmet nyújtani:

```
"SELECT * FROM felhasznalok WHERE nev = '" . $userName . "'"
```

Az egyik támadási módszer, ha a felhasználó által megadott *$userName* változó, egy másik SQL parancsot tartalmaz:

```
a'; DROP TABLE csoportok; --
```

Ez ellen a PHP beépített *mysql_query* függvénye nyújt védelmet azzal, hogy egy kérésben csak egy SQL parancsot enged lefuttatni.

A másik támadási módszer, ha a *$userName* változó, egy SQL feltételt tartalmaz:

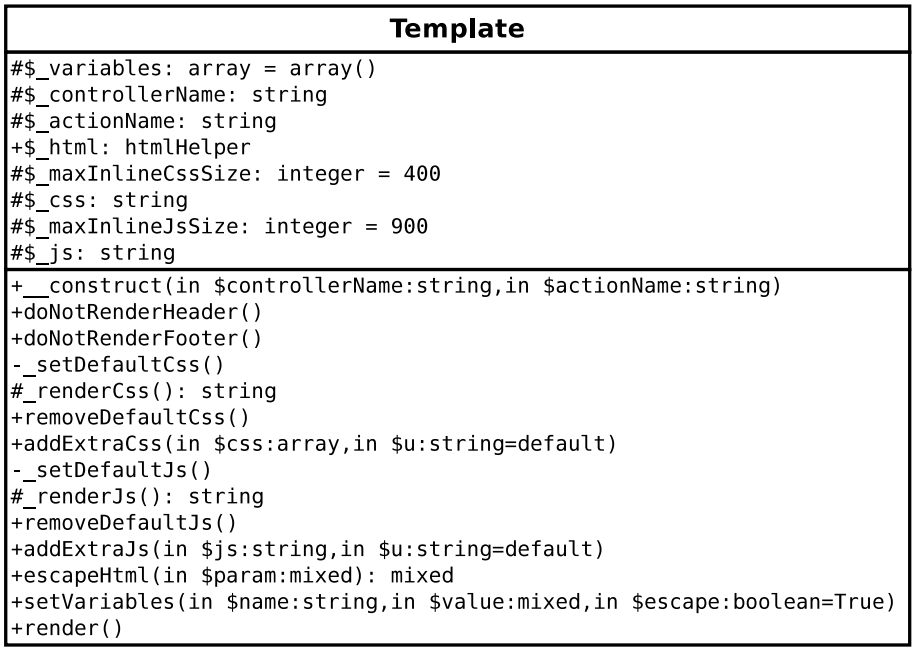
```
a' OR 1 = 1 --
```

A támadás kivédésére, mind a lekérdezés, mind a módosítás során, az összes változóra a rendszer meghívja a PHP beépített *mysql_real_escape_string* függvényét, mely kimenti a speciális SQL karaktereket, úgy mint: `\x00, \n, \r, \, ', "` és `\x1a`.

A támadás sikeres kivédése érdekében fontos, hogy az SQL kérést és a benne lévő változókat, külön paraméterben adjuk át a *Model* osztály metódusainak, különben a *mysql_real_escape_string* függvényt a rendszer nem hívja meg. [6]

3.3. Nézet réteg

Nézetek előállítása a *libs/template.class.php* fájlban található *Template* osztály segítségével történik.



3.8. ábra. Template osztály

Példányosításkor meg kell adnunk annak a vezérlőnek és akciónak a nevét, mely számára szeretnénk nézetet generálni.

A konstruktor lefutása után *$_html* attribútumban a *htmlHelper* osztály egy példányát fogja tartalmazni, így az ott található függvényeket könnyel elérhetjük.

A *doNotRenderHeader* és a *doNotRenderFooter* metódusok meghívásával állíthatjuk be, hogy ne jelenjen meg a *views/header.php*, illetve a *views/footer.php* fájlban leírt fejléc, illetve lábléc.

A *_setDefaultCss* és a *_setDefaultJs* tagfüggvénnyel szabályozható, hogy alapból milyen stíluslapok és JavaScript fájlok jelenjenek meg az egyes felhasználói csoportoknak. Az alapértelmezett fájlokat törölni a *removeDefaultCss* és a *removeDefaultJs* metódusok segítségével tudjuk. Hozzáadni az *addExtraCss* és az *addExtraJs* függvények segítségével lehetséges. Az oldal számára elérhető CSS és JavaScript fájlokat a *webroot/css*, valamint a *webroot/js* mappák tartalmazzák. A jelenleg megtalálható stíluslapok közül a *print.css* írja le a nyomtatáshoz használt kinézetet. A *mobile.css* fájlban található a 640 pixelnél keskenyebb képernyővel rendelkező készülékek számára kialakított oldalstílus, míg a *master.css* fájlban található az átlag böngésző számára definiált kinézet.

A *_renderCss* és a *_renderJs* metódusok állítják elő azt a szöveget, aminek a HTML oldalba való beszúrásával beágyazható a kívánt tartalom. Az oldal betöltés gyorsítása érdekében, a *$_maxInlineCss* és a *$_maxInlineJs* attribútumok által meghatározott méretnél kisebb fájlok teljes tartalma be lesz illesztve.

Az *escapeHtml* függvény segítségével egy tetszőleges mélységű tömb értékeiből kimenthetjük a speciális HTML vagy JavaScript karaktereket (úgy mint: `&`, `"`, `'`, `<`, `>`).

A *setVariables* tagfüggvény, a nézet számára fontos változók átadására szolgál. Az első paraméterrel adhatjuk meg a változó nevét, a másodikkal az értékét, míg a harmadik paraméterrel szabályozhatjuk, hogy a változó értékéből ki legyenek-e mentve a speciális HTML és JavaScript karakterek.

Az XSS támadások kivédése érdekében fontos, hogy minden olyan változó értékéből ki legyenek mentve ezek a speciális karakterek, amikben a felhasználó által megadott adat található. Az XSS támadásról bővebben a 3.3.5 fejezetben olvashat.

A *render* függvény meghívásával, pedig megjelenik a képernyőn a megfelelő vezérlő, megfelelő akciójához tartozó nézet, melyet a *views* mappán belül a vezérlő nevével megegyező almappában fog keresni, az akció nevével megegyező néven.

A nézet réteg további három részre oszlik: struktúrára, prezentációra és viselkedésre.

3.3.1. Struktúra

A struktúra réteg HTML leíró nyelven íródott. Feladata a megjeleníteni kívánt tartalom szerkezetének leírása. A fájlok a *views* mappában találhatóak, további almappákba csoportosítva aszerint, hogy melyik vezérlőhöz tartoznak.

A *views/header.php*, *views/menu.php* és a *views/footer.php* fájlok tartalmazzák azokat a sablonokat, melyeket szinte minden oldal használ az oldal fejlécének és láblécének előállításához és a menü megjelenítéséhez.

3.3.2. Prezentáció

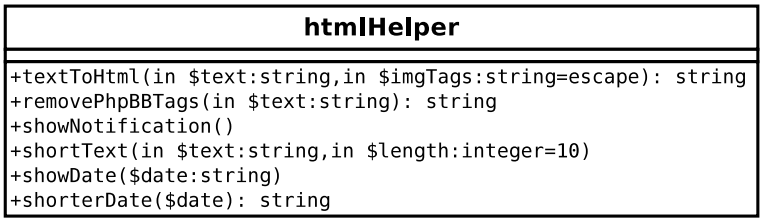
A prezentáció réteg, CSS nyelven íródott és a *webroot/css* mappában található. Feladata a megjeleníteni kívánt oldal stílusának definiálása. Az oldal betöltés gyorsítása végett, ezekből a fájlokból hiányoznak a fehér szóközök. Az eredeti fájlok megtalálhatóak a mellékletben.

3.3.3. Viselkedés

A viselkedés réteg JavaScript nyelven íródott és a *webroot/js* mappában találhatóak. Feladata meghatározni az egyes felhasználói viselkedésre adott reakciókat. Például a hely szűkében lerövidített tartalmak megjelenítése, amint az egérmutató a szöveg felé kerül. Ezekből a fájlokból, akár csak a prezentáció réteghez tartozóakból, hiányoznak a fehér szóközök és a megjegyzések. Az eredeti fájlok megtalálhatóak a mellékletben.

3.3.4. HTML segédosztály

A *htmlHelper* osztály tartalmazza azokat a függvényeket, melyek egyszerűbbé teszik a HTML fájlok írását.



3.9. ábra. htmlHelper osztály

A *textToHtml* tagfüggvény, egy PhpBB kódokat tartalmazó szöveget alakít át HTML szöveggé. Második paraméterrel meghatározható, hogy a kép beszúrására alkalmas *img* tagokkal mit kezdjen: *escape* esetén marad PhpBB kód, *show* esetén le lesznek cserélve HTML tagokká, míg a *hide* paraméter átadásánál nem jelennek meg a HTML kimenetben.

removePhpBBTags metódus segítségével el lehet távolítani egy szövegből az összes PhpBB tagot.

showNotification metódus kiír egy korábban beállított figyelmeztetést. A kiírással törlődik a szöveg, későbbiek során nem lehet újra kiíratni.

shortText tagfüggvény lerövidít egy szöveget, a második paraméterben megadott hosszra. A teljes szöveg meg fog jelenni, amennyiben az egér mutató a szöveg fölé kerül.

showDate kiír egy dátumot a magyar nyelvben használt formátum szerint.

shorterDate levágja egy dátum végén lévő esetleges felesleges értékeket. Például 1986-12-21 00:00:00 dátumból a következőt készíti: 1986-12-21

3.3.5. XSS támadás elleni védelem

Az XSS rövidítés a Cross-site Scripting (oldalak közti programozás) kifejezésre utal. Az általában web alkalmazásoknál előforduló támadás során egy rosszindulatú felhasználó megpróbál a honlapra olyan HTML, JavaScript vagy egyéb kliens oldali kódot beilleszteni, amit más felhasználó is lát. Egy megfelelően összeállított kóddal képes például más felhasználóhoz tartozó érzékeny (tipikusan sütikben tárolt) adatok ellopására. [3]

A támadás kivédése érdekében a rendszer az összes, nézet számára átadott változóból kimenti a speciális HTML és JavaScript karaktereket így gátolva, az esetleges felhasználó által beágyazott kódok lefutását. Ha azt szeretnénk, hogy az átadott változóból ne legyenek kimentve ezek a karakterek, a *Template* osztályban található *setVariables* tagfüggvény harmadik paraméterében hamis értéket kell megadunk. A változók ilyen típusú átadását, csak abban az esetben használjuk, amennyiben az adat biztos nem tartalmaz a felhasználótól származó információt.

3.3.6. Optimalizálás mobil eszközökre

Mivel a játék során a felhasználók mobil készüléken keresztül kapják meg az adatokat, fontos az oldal felületének optimalizálása ezekre az eszközökre.

Optimalizálás során, két fontos feladatra kellett ügyelni: a kisebb kijelző méretre, illetve a gyors oldalbetöltés szükségességére.

A böngésző típusának felderítése és egy speciális mobil készülékekre kialakított oldalra való átirányítása helyett, inkább az új technológiák (pl.: CSS3) által nyújtott lehetőségek alkalmazását választottam. Így nem kell egy új böngésző vagy készülék megjelenése miatt, újra írni az oldal megfelelő részeit.

Oldal elemek elrendezése

A felhasználói élmény javítása érdekében fontos szempont volt a nagy kontraszt, az oldal háttér színe és a rajta lévő szöveg között. Így könnyen olvasható marad, ha erős napfényben böngészik a honlapot.

Sok felhasználó kikapcsolja a képek letöltését böngészés során, így gyorsítva az oldal betöltését. Ezért fontos minden egyes kép beágyazásánál szövegesen leírni, az *alt* attribútumban, a kép tartalmát.

Kép beszúrásánál mindig meg kell adni a szélességét és hosszúságát, így mikor a szöveg betöltődött a képek viszont még nem, a böngésző automatikusan kihagyja a helyet. Mikor a képek is betöltődnek, nem fogja a böngésző újra méretezni a szöveget.

Egy átlagos oldal megnyitása során a mobil készülékek böngészői először, egy összesített indexképet mutatnak, a felhasználó pedig, ennek tetszőleges részébe nagyíthat bele. Az Opera Mobile webböngésző például egy 850 pixel széles oldalt kicsinyít le annyira, hogy elférjen a telefon kijelzőjén. Ahhoz, hogy a látogató rögtön egy bele nagyított képet kapjon, a HTML oldal fejlécében a következő sor kell elhelyezni:

```
<meta name="viewport" content="width=device-width" />
```

A CSS3 új, *Media Queries* tulajdonságának köszönhetően, külön stílust lehet definiálni a kis és a nagy kijelzőkre. Használatának nagy előnye, hogy nem a szerver dönti el, hogy milyen elrendezést kell alkalmazni az oldalon, hanem a böngésző. Így a jövőben megjelenő új készülékeken is megfelelően fog megjelenni a tartalom. A 3.10-es ábra szemlélteti, hogyan változik az oldal címsora és menüpontjai, ha 640 pixelnél keskenyebb kijelzőn nézik.

3.10. ábra. Oldal elemek változása

Oldalbetöltés gyorsítása

Mobil böngészők esetében különösen fontos szempont az oldal mérete, mivel sok esetben az internet előfizetés díját, az aktuális hónapban letöltött adatok mérete határozza meg.

Minden egyes kép, CSS vagy JavaScript fájl új HTTP kérésként töltődik le. Ezért fontos, hogy a lehető legkevesebb fájlt használjuk. A kis képeket célszerű egy nagy képpé olvasztani és *optipng* vagy *jpegoptim* programmal, veszteség mentessen lecsökkenteni a méretüket. A CSS és JavaScript fájlokból kiszedni a fehér szóközöket és megjegyzéseket és a kis méretű fájlokat közvetlen a HTML lapba beszúrni.

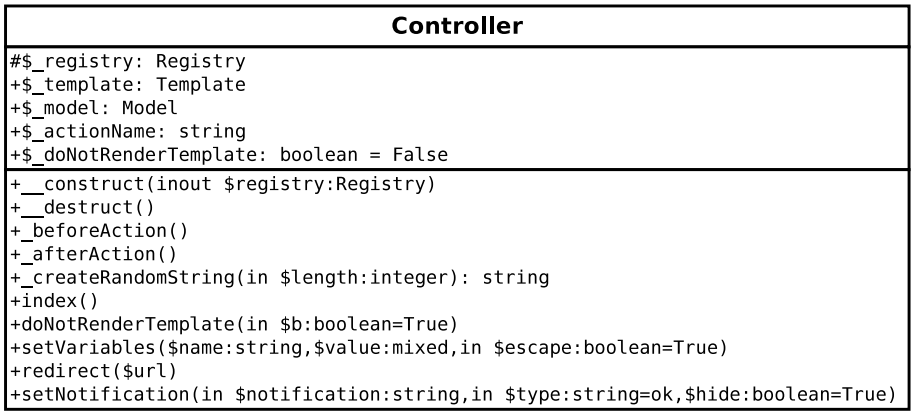
A böngésző megáll az oldal betöltésével, mikor egy JavaScript fájl letöltéséhez ér, így célszerű ezeknek a fájloknak minél későbbi beszúrása. [2]

3.4. Vezérlő réteg

3.4.1. Vezérlők működése

A honlap szolgáltatásai csoportosítva vannak. Minden egyes csoportot külön vezérlő valósít meg, melyek a *libs/controller.class.php* fájlban található, *Controller* ősosztály leszármazottai. Ilyen csoport például a kérdések, mely a kérdések szerkesztését, törlését, módosítását és a hozzá tartozó QR kódok generálását foglalja magában. Az egy csoportba tartozó funkciókat, a leszármazott osztály egy-egy metódusa valósítja meg. Ezeket a metódusokat hívjuk akcióknak. Egy új vezérlő

létrehozásakor, legalább egy *index* akciót definiálni kell.



3.11. ábra. Controller osztály

Az osztály konstruktora létrehozz a *Template* osztályból és a vezérlőhöz tartozó *Model* osztály leszármazottjából egy példányt.

A *_beforeAction* tagfüggvényben definiáltak, a leszármazott osztályok akcióinak meghívása előtt futnak le, míg a *_afterAction* metódusokban megadottak, azok után.

A *_createRandomString* metódus segítségével, tetszőleges hosszúságú, az angol ábécé kisbetűit és számokat tartalmazó, véletlen karaktersorozatot hozhatunk létre.

A *doNotRenderTemplate* tagfüggvénnyel állíthatjuk be, hogy generálódjon-e, az adott akcióhoz látvány.

A *redirect* metódus segítségével átirányíthatjuk a felhasználót egy másik oldalra.

3.4.2. Vezérlők és a látvány kapcsolata

A *setVariables* függvény segítségével tudunk átadni változókat a nézet számára. Az esetleges XSS támadások kivédése érdekében a függvény, minden változóból kimenti a speciális HTML és JavaScript karaktereket. Ennek megakadályozását a harmadik paraméterben átadott hamis értékkel érhetjük el.

A *setNotification* metódus segítségével figyelmeztető vagy informáló üzenetet adhatunk át a nézetnek. Egy üzenet addig tárolódik, ameddig a nézet ki nem írja vagy egy újabb üzenet beállításával felül nem írjuk azt. Harmadik paraméterrel állítható be, hogy az üzenet 4 másodperc elteltével magától eltűnjön-e vagy sem.

Miután egy vezérlő befejezte feladatát, a destruktornak köszönhetően, automatikusan elkezdődik a megfelelő látvány generálása, a korábban átadott üzenetekkel és adatokkal.

3.5. Rendszer követelmények

A program helyes működéséhez a következő feltételek teljesülése szükséges:

- Apache2 webszerver
- legalább PHP 5.2
- legalább MySql 5.0
- a webszervert futtató felhasználónak (általában www-data) írási jogosultság a következő mappákra:
  - tmp/logs
  - tmp/cache
  - webroot/files/kerdesek
  - webroot/files/lehetseges_valaszok
  - webroot/files/qrcodes
- *.htaccess* fájlok engedélyezése, illetve a *rewrite* modul megléte a webszerveren.
- továbbá nem szükséges, de az oldal betöltést gyorsítja, ha a webszerveren megtalálható az *expires* modul, illetve, ha a gzip tömörítés be van kapcsolva.

Az automatikus tesztek futtatásához a következő programokra van szükség:

- Firefox böngésző
- Ruby 1.8.6 vagy 1.8.7
- Watir
- Rspec

3.6. Felhasznált modulok

- MarkItUp 1.1.10 (2011.02.20)
  `http://markitup.jaysalvat.com`
  Szövegszerkesztő a kérdések és lehetséges válaszok könnyebb formázásához. Megtalálható a *webroot/js* mappában.

- jQuery 1.5.1 (2011.02.23)
  `http://jquery.com`
  JavaScript keretrendszer.
  Megtalálható a *webroot/js* mappában.

- PhpQrCode 1.1.4 (2010.10.07)
  `http://phpqrcode.sourceforge.net`
  QR kódok generálásához.
  Megtalálható a *libs/phpqrcode* mappában.

4. fejezet

Tesztelés

A program helyes működésének folyamatos ellenőrzését, az automatikus tesztek segítségével valósíthatjuk meg.

A tesztek Ruby programozási nyelven[1], a Watir[2] és az Rspec[3] könyvtárak felhasználásával készültek. A fájlok a *testing* mappában találhatóak. Futtatásukhoz az előbb említett két programkönyvtárra illetve, Firefox[4] webböngészőre van szükség. Egy Super User e-mail címének és jelszavának, valamint a holnap URL-jének, a *config.rb* fájlba írása után (érdemes létrehozni egy új felhasználót erre a célra), a `ruby start.rb` paranccsal futtathatóak.

Indítást követően megjelenik a böngésző és a program megkezdi a honlap alapvető funkcióinak tesztelését. Az összes teszt lefutása után, kiírja a talált hibákat és, hogy melyik funkció tesztelésénél találta azt.

A program egy játékot fog lejátszani, kezdve a játék létrehozásával, kérdések és hozzá tartozó lehetséges válaszok felvételével, egészen a kérdésekre való válaszolásig. A folyamat során megnézi, hogy helyesen működnek-e az oldal szolgáltatásai, megpróbál hibás adatokat bevinni, nem szerkeszthető kérdéseket szerkeszteni, illetve törölni, valamint különböző támadásokat végrehajtani az oldal ellen. Folyamatosan figyelve az oldal reakcióját (például nem elég, hogy nem fogadja el a hibás adatokat, megfelelő figyelmeztetéseket is ki kell, hogy írjon). A tesztek külön fájlokba vannak csoportosítva aszerint, hogy a teszt, az oldal melyik funkcióját ellenőrzi. Minden fájlban (megjegyzés formájában) megtalálható a tesztelés teljes forgatókönyve.

A tesztek hasznosak annak megállapítására, hogy a rendszer telepítése sikeres

[1] installálása Debián alapú rendszereken: `sudo apt-get install ruby-full rubygems`

[2] installálás: `sudo gem install watir`

[3] installálás: `sudo gem install rspec`

[4] letölthető a `http://getfirefox.com` címről

volt-e, illetve új funkciók implementálásával nem romlottak-e el az eddig meglévő részek.

Az automatikus tesztelésen kívül, alkalom nyílt két éles játék során, az esetleges hibák felderítésére.

5. fejezet

Összegzés

5.1. Továbbfejlesztési lehetőségek

A rendszer tervezése során, fontos szempont volt, hogy a következő funkciókat a későbbiek során, könnyen meg lehessen valósítani:

- Manuális kiértékelést igénylő kérdések felvételének lehetősége. A válaszért kapott pontot a játék vezető határozza meg.
- A kérdések és lehetséges válaszainak más nyelvere történő lefordítását követően, a felhasználó regisztráláskor beállíthassa a számára legmegfelelőbb nyelvet.

5.2. Eddigi játékok

A játékkal eddig két alkalommal találkozhattak az érdeklődők:

- 2011. január 26-án az ELTE IK Nyílt nap keretében, válaszolhattak a játékosok az ELTE 375. évfordulójával kapcsolatos kérdésekre.
- 2011. április 9-én és 10-én az ELTE IK és a MOME közös szervezésében, a 100 éves házak című rendezvényen[1] a játékos kedvűek, QR vadászattal fedezhették fel három, századik születésnapját ünneplő ház történetét.

  A szervezők kérésére, a honlap funkciói is bővültek. Egy QR kódhoz innentől kezdve, több kérdést is hozzá lehet rendelni, illetve megjelent a publikus regisztráció beállításának lehetősége az egyes játékoknál. Új kinézetet is kapott az oldal, igazodva a rendezvény hivatalos honlapjának stílusához.

5.1. ábra. 100 éves házak rendezvényre készített kinézet

Mindkét esetben a honlap az ELTE Caesar szerverén, a `http://qrportal.elte.hu` címen volt elérhető.

[1]bővebb információt a `http://www.budapest100.hu` honlapon olvashat

6. fejezet

CD tartalma

A mellékelt CD-n a következő fájlok találhatóak:

`config` mappában található, *config.php* tartalmazza az oldal működéséhez nélkülözhetetlen beállításokat. Például a MySql kapcsolathoz szükséges adatokat.

`controllers` mappa tartalmazza a vezérlőket.

`doc` mappában találhatóak az oldalon használt képek nagy felbontású változatai, a tömörítetlen CSS és JavaScript fájlok, az adatbázis létrehozásához szükséges SQL parancsok, valamint az adatbázis szerkezetét és a felhasználói csoportok jogait szemléltető képek.

`libs` mappában, a QR kódok generálásához felhasznált függvények, valamint a fontosabb osztályok találhatóak.

`models` tartalmazza a modell réteget leíró fájlokat.

`testing` mappában az automatikus teszteléshez szükséges fájlok találhatóak.

`tmp` mappába írja a rendszer az ideiglenes fájlokat, valamint a működés során fellépő hibákat.

`views` tartalmazza a nézet réteget leíró fájlokat.

`webroot` mappában találhatóak a rendszer működéséhez szükséges statikus fájlok. Például képek, CSS vagy JavaScript fájlok.

`.htaccess` fájl speciális beállításokat tartalmaz az Apache webszerver számára.

Irodalomjegyzék

[1] Guzel, Burak: Understanding Hash Functions and Keeping Passwords Safe, 2011. január 17. Link `http://net.tutsplus.com/tutorials/php/understanding-hash-functions-and-keeping-passwords-safe` (elérés dátuma: 2011. április 25.)

[2] Lawson, Bruce: Mobile-friendly: The mobile web optimization guide, 2010. július 28. Link: `http://dev.opera.com/articles/view/the-mobile-web-optimization-guide` (elérés dátuma: 2011. április 20.)

[3] Wikipédia szerkesztői: Cross-site scripting, 2011. április 18. Link: `http://en.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=424749244` (elérés dátuma: 2011. április 25.)

[4] Wikipédia szerkesztői: Modell-nézet-vezérlő, Wikipédia, a szabad enciklopédia, 2011. február 26. Link: `http://hu.wikipedia.org/w/index.php?title=Modell-n%C3%A9zet-vez%C3%A9rl%C5%91&oldid=9319989` (elérés dátuma: 2011. április 19.)

[5] Wikipédia szerkesztői: QR-kód, Wikipédia, a szabad enciklopédia, 2011. április 12. Link: `http://http://hu.wikipedia.org/w/index.php?title=QR-k%C3%B3d&oldid=9530318` (elérés dátuma: 2011. április 19.)

[6] Wikipédia szerkesztői: SQL injection, Wikipédia, a szabad enciklopédia, 2011. április 20. Link: `http://en.wikipedia.org/w/index.php?title=SQL_injection&oldid=424982648` (elérés dátuma: 2011. április 21.)

[7] Wikipédia szerkesztői: Vonalkód, Wikipédia, a szabad enciklopédia, 2010. december 5. Link: `http://hu.wikipedia.org/w/index.php?title=Vonalk%C3%B3d&oldid=8901051` (elérés dátuma: 2011. április 30.)