Grafilogika játék – Dokumentáció

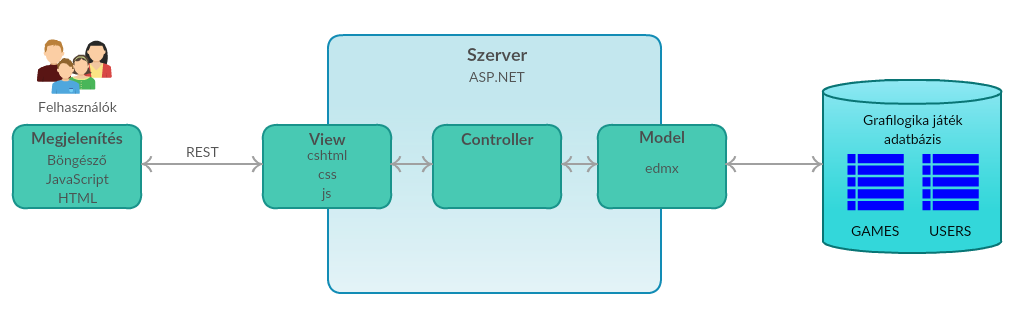
# Bevezetés

A grafilogika játék egy jól ismert keresztrejtvény, aminek a lényege, hogy az oszlopokban és sorokban lévő számok alapján színezzük be a mezőket, amit ha jól csinálunk, akkor egy képet kapunk. A mi feladatunk az volt, hogy ehhez egy olyan webes rendszert készítsünk, ami a következő funkciókkal rendelkezik:

* A rendszer háromféle usert különböztet meg: admin, regisztált felhasználó, látogató
* Minden felhasználó képes játékot játszani
* A regisztrált felhasználók tudnak játékot hozzáadni, és a saját játékaikat törölni. Emellett értékelhetik más felhasználók játékait, és megtekinthetik saját és a játékok statisztikáit.
* Az admin felhasználók ugyanezek mellett képesek bármely felhasználó játékát törölni.

Ennek megvalósítására mi az ASP.NET-et választottuk a kérések szerveroldali feldolgozására; html, css és javascript felel a kliens oldal megjelenítéséért és funkcionalitásáért; a felhasználók és a játékok adatait pedig SQL adatbázis tárolja. A továbbiakban ezek megvalósításáról olvashatunk részletesebben.

# Rendszer megvalósítása



. ábra. Architektúra átfogó képe

## Adatbázis

Az adatok tárolására két táblát hoztunk létre: Users tábla, a regisztrált és admin felhasználók adatainak tárolására; valamint a Games tábla, a játékok adataihoz. Az alábbiakban az egyes táblák mezőit, azok típusait és feladatát ismertetjük.

### Users

**Name**: varchar(100); Felhasználó nevét tárolja. Csak alfanumerikus karakterekből állhat. Ez a tábla egyedi azonosítója. Nem lehet két azonos nevű felhasználó.

**Password**: varchar(100); Felhasználó jelszavát tárolja. Csak alfanumerikus karakterekből álló lehet, melyet titkosítva tárolunk.

**Wins**: int; Játékos sikeres megoldásainak számát tárolja.

**Mistakes**: int; Játékos rossz megoldásainak beküldését tartja számon.

**Isadmin**: int; Jelzi, hogy az adott felhasználó rendelkezik-e az admin jogaival.

### Games

**Name**: varchar(100); Játék nevét tárolja. Csak alfanumerikus karakterekből állhat. Ez a tábla elsődleges kulcsa. Nem lehet két azonos nevű játék.

**Uploader**: varchar(100); Játékot létrehozó vagy feltöltő játékos neve.

**Wins**: int; Számon tartja, hogy a játékot hányszor oldották meg jól.

**Mistakes**: int; Számon tartja, hogy mennyi rossz megoldást küldtek a játékra.

**Rating**: int; A feladványt megoldó játékosok értékelhetik 1-től, 5-ig a játékot. Ezek átlagát tárolja.

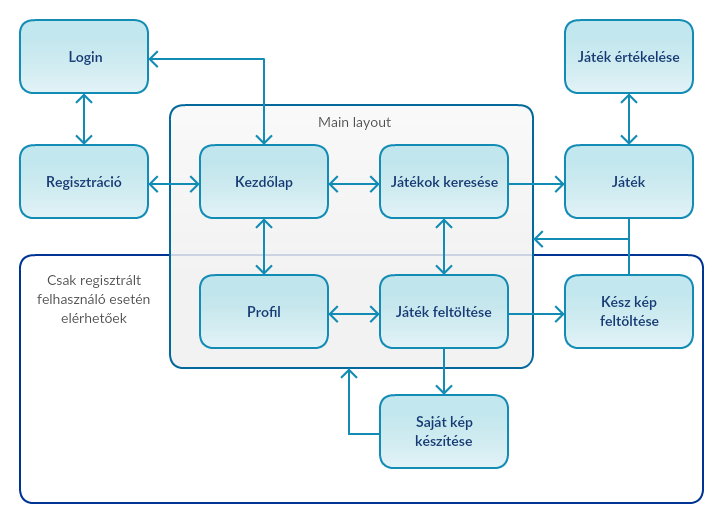
**Game**: varchar(max); Feladvány tárolása szövegesen történik ebben a mezőben. Ennek módját később részletesen is bemutatjuk.

## Szerver oldal

A következő pontokban a szerveren futó üzleti logikát, a kommunikációk irányát és módját, az adatbázis kezelését, valamint az adatok validációjának megvalósítására térünk ki.

### Kommunikáció

Első lépésként az alkalmazásból elérhető oldalakat, és az azok közti navigációt mutatnánk be az alábbi ábra segítségével.



2. ábra. Oldalak navigációs hálója

HÁT TOMI EZT MÁR TÉNYLEG NEKED KÉNE ÍRNI, MERT ITT BESZÉLJ AZ AJAXRÓL ÉS NEM, ÉS HOGY HOGYAN MENNEK ÉS MIÉRT ÉS MERRE, MAJD A JÁTÉK BEKÜLDÉSÉRE VEZESD ÁT, HOGY OTT RÉSZLETEZED AZTAT.

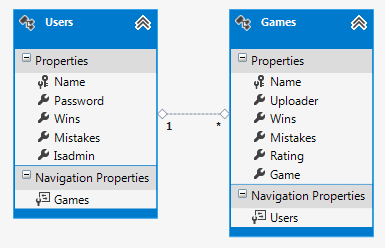
#### Játék megoldásának beküldése

TODO Beküldés jó/rossz mi jön fel, rating, stb. Egy ábra lehet menő lenne. TODO

### Adatbázis menedzsment

Az adatbázis menedzsment feladatát a *DBManager.cs* statikus osztály végzi. Ebben LINQ lekérdezéseket megvalósító függvényeket találhatunk az alábbi műveletekre:

* Új játék hozzáadása az adatbázishoz
* Új felhasználó hozzáadása az adatbázishoz
* Lekérdezés egy felhasználó összes feltöltött játékára
* Egy adott játék lekérdezése annak neve alapján
* Az összes játék lekérése
* Az össze felhasználó lekérése
* Egy adott felhasználó lekérdezése annak regisztrált neve alapján
* Update lekérdezések felhasználók és játékok győzelmeinek, rossz megoldásainak frissítésére, valamint a játékok értékelésének frissítésére



. ábra. Adatbázistáblák az edmx-ben.

### Adatvalidáció

A felhasználók által megadható adatoknál és változóknál ügyelnünk kell azok helyességére, hogy biztosítsuk a helyes működést, és védjük az oldalt az esetleges támadások ellen. A következő felsorolásban az ellenőrizendő adatokat, és a hozzájuk tartozó validáció módját ismertetjük:

* Felhasználónév, jelszó, játéknév. Mind maximum 100 karakter hosszú, csak alfanumerikus karaktereket tartalmazó szöveg lehet. TODO HOGY BIZTOSÍTOD
* Grafikusan készített játék esetén a sorok és oszlopok száma csak számokat tartalmazhat, melynek maximális értéke 60. TODO HOGY BIZTOSÍTOD
* ASCII art fájl felöltésekor csak \*.txt típusú dokumentumot fogadunk, és az elvárt formátumnak megfelelő adattal feltöltve. Az elvárt formátum ellenőrzését a következő, üzleti logika fejezet alatt részletezzük. A fájl típusának ellenőrzése TODO PLZ

### Üzleti logika

A következőekben a szerveren futó feladatok megvalósítását részletezzük.

#### ASCII art fájl feldolgozása

TODO hogyan jön ide a file. Ezt követően lementjük a fájlt egy átmeneti „Upload” mappába, és innen olvassuk ki soronként. Ekkor történik az elvárt tartalom ellenőrzése. Ha bárhol sérti a követelményeket, egy kis felugró TODOFANCYBOX segítségével jelezzük az oldalon. Az első beolvasott sor hossza alapján máris meg tudjuk nézni, nem töltött-e fel több oszlopból álló rejtvényt, mint a megengedett. Ha azonban az első sor hossza megfelelt, elmentjük ezt a hosszt, és későbbi sorok beolvasásánál ezzel való egyenlőségüket vizsgáljuk, hogy ne lehessen olyan játékot feltölteni mely nem téglalap- és négyzetformát követ. Eltérés esetén jelezzük, melyik sor nem azonos hosszú. Ha a sor hossza megfelelő, elkezdünk rajta karakterenként végighaladni, és keressük az elvárt **X** és **space** karaktert. Az **X** tetszőleges méretű lehet, mikor feltölti a felhasználó, nem case sensitive a feldolgozás. Ha **X**-et olvasunk, egy *StringBuilder*-hez hozzáfűzünk egy **1**-est; ha **space**-t, akkor **0**-t. Bármely más olvasott karakter esetén jelezzük a felhasználó felé, hogy nem elfogadott elemeket is tartalmaz a feltöltött rejtvénye. Ha végigértünk a soron, jön hozzáfűzzük a sor végét jelölő **;**-t. A játék szöveggé alakításának pontosabb szemléltetését a kliens oldal ismertetésekor, a játék feltöltése fejezet alatt is részletesen ismertetjük. Az alábbi ábra szemlélteti egy beolvasás eredményét:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | X |  | **=>** | 010;101;010; |
| X |  | X |
|  | X |  |

. ábra. ASCII art - szöveges formátum átalakítás példa

Ha a fájl és tartalma teljesen megfelelő volt, és a felhasználó által megadott játéknév megfelelt a korábban ismertetett adatvalidációnak, hozzáadjuk a játékot az adatbázishoz - feltéve, hogy még nem volt ilyen nevű. Ha volt jelezzük -, a megadott névvel, az adott felhasználó neve alatt, és a kész szöveges játékleíróval. A játék egyéb statisztikái természetesen alapesetben 0 értéket kapnak. Befejezésként töröljük az átmeneti fájlt és mappáját. TODO KELL KÓD?

#### Játékok böngészése

JavaScript nélküli blackmagic

#### Profil

Lsd. korábbi böngészéses megoldás valszeg

#### Authorizáció és belépés

Háteztafaszomtudja

#### Regisztráció

Ha a felhasználó minden adatot megadott, majd azok átmentek a korábbiakban ismertetett adatvalidációs szűréseken, és még nem létezik ilyen nevű felhasználó, akkor elmentjük a felhasználót a *DBManager* segítségével. TODO KELL MÉG VMI?

## Kliens oldal

Kliensnél futnak a betöltött weboldalak, és az azok megjelenítéséért és működtetéséért felelős javascriptek. A következőkben az általa végrehajtható műveletek megvalósítását részletezzük, képekkel illusztrálva.

### Regisztráció és belépés

Az oldalra navigáláskor egy login képernyő fogadja a felhasználókat, ahol megadhatják adataikat és továbbléphetnek a játék funkcióira. Ha nem akarnak belépni, vagy nincsenek regisztrálva, vendégként is elindíthatják a játékot, ekkor azonban statisztikáik nem kerülnek mentésre.

***INSERT SCREENSHOT HERE(REG/LOGIN)***

A regisztráláshoz és elfelejtett jelszó visszaszerzésére is elérhetők linkek, a korábban már ismertetett funkcionalitással. A validációkat szerver oldalon ellenőrizzük.

### Játék feltöltése

A regisztrált és admin felhasználók képesek általuk létrehozott rejtvények feltöltésére az oldalon. Ezt két módon tehetik meg: szöveges fájlban létrehozott úgynevezett ASCII art formájában; vagy az oldalon megadott SORxOSZLOP méretű négyzetháló kiszínezésével. Ha egy felhasználó a játék feltöltése menüpontot választja, akkor az alábbi oldal jelenik meg, ahol két gomb segítségével választhat a feltöltési módokból. Ezek működését láthatjuk a következő két fejezetben.

***INSERT SCREENSHOT HERE(UPLOAD GOMBOK)***

#### ASCII art formában

Ha ezen opció mellett dönt a felhasználó, a következő képen látható weboldal fogadja.

***INSERT SCREENSHOT HERE(UPLOAD FILE)***

Itt csupán meg kell adnia a fájlt, amiben a rejtvényt elkészítette, adni a játéknak egy nevet, majd feltölteni a szerverre a gomb lenyomásával. Csak \*.txt fájlokat fogad a rendszer, melyekben **X/x** jelöli a színesnek szánt, **space/szóköz** pedig az üresnek szánt mezőket. Ezekről a felhasználó az oldal jobboldali paneljén olvashat, vagy le is csukhatja azt, ha már ismeri a működést. Az alábbi kis példán bemutatjuk, hogyan kell a fájlba beírni az általunk elképzelt feladványt:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **=>** |  | X |  |
|  |  |  | X |  | X |
|  |  |  |  | X |  |

. ábra. Rejtvény - ASCII art konverzió

Beküldés után a szerver feladata feldolgozni a kapott fájlt. Ezt a szerver oldal leírásakor már ismertettük.

#### Grafikusan

Ha ezen opció mellett dönt a felhasználó, egy olyan weboldal fogadja, melyen megadhatja a sorok illetve oszlopok számát; majd a generálás gombra kattintva dinamikusan létrejön javascript segítségével egy ekkora négyzetháló, melyekbe belekkattintva átszínezheti a mezőket. Ezen oldal látható az ábrán.

***INSERT SCREENSHOT HERE(CREATE GAME TABLE)***

Alapvetően az oldalt leíró html csak egy üres táblázat (*<table>*) tag-et tartalmaz, ezt tölti fel a *CreateGameTable.js* javascript. Soronként hozzáfűz a táblázathoz egy sort *(<tr/>*), majd oszloponként egy cellát ehhez az új sorhoz *(<td/>*), végül a cellába tesz egy gombot, melyet a megfelelő megjelenés érdekében ellát egy *gamecell* stílusosztállyal, valamint egy eseménykezelővel, mely azért felel, hogy ki-be kapcsolja a cella háttérszínét adó css osztályt (*toggleclass(black)*) ha a felhasználó rákattint.

for (i = 0; i < row; ++i) {

var newrow = $('<tr class="game" />');

for (j = 0; j < col; ++j) {

var newcol = $('<td class="game" />');

var button = $('<input type="button"/>')

.addClass('gamecell')

.click(function () { $(this).toggleClass('black'); });

button.id = "button" + i + j;

newcol.append(button);

newrow.append(newcol);

}

$("#gametable").append(newrow);

}

Ha a felhasználó elkészült a feladvánnyal a ’Pálya beküldése’ gombra kattintva feltöltheti azt a szerverre. A javascript ellenőrzi, hogy ne töltsön fel senki üres rejtvényt. Az adatbázisnál már említettük, hogy a játékok szövegesen vannak lementve. Az itt létrehozott táblázat gombjait úgy alakítjuk szöveggé, hogy vizsgáljuk rendelkeznek-e a korábban említett *’black’* css osztállyal. Ezek után az alábbi módon történik a szövegesítés: ha egy mező fekete, akkor **1** jelöli, ha fehér, akkor **0**. Ezzel még csak a színűket tároljuk. A sorok azonosítására, végének jelölésére a **;** szolgál. Könnyebb megértésért tekintsük meg az alábbi példát:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **=>** | 101;010;101; |
|  |  |  |
|  |  |  |

. ábra. Rejtvény szöveggé alakítása.

if ($("#gametable").children().length != 0) {

var str = "";

$("#gametable tr").each(function () {

var columns = $(this).find('td');

columns.each(function () {

var button = $(this).find(":button");

if (button.hasClass("black"))

str += 1;

else

str += 0;

});

str += ";";

});

if (str.indexOf("1")!=(-1)) {

…send to server here…

}

}

alert("Üres pályát nem lehet a szerverre küldeni!");

}

Ezzel végignéztük mindkét játékkészítési lehetőséget. Következő pontokban jön az, hogyan lehet elérni és játszani a létrehozott rejtvényekkel.

### Játékok böngészése

Ha a felhasználó játszani szeretne, átnavigál erre az oldalra. Itt egy táblázatban, sorban szerepel minden eddig létrehozott és feltöltött játék, statisztikáikkal együtt.

***INSERT SCREENSHOT HERE(BROWSE)***

A mellettük szereplő gombbal átnavigálhatunk az oldalra, ahol már valóban a rejtvény és a játék vár minket. Erről szól a következő pont.

### Játék

A következő oldal a böngészésből kiválasztott játékot jeleníti meg. A megjelenítés a grafikus játékgenerálásnál látott módon egy üres táblázat feltöltésével történik, de itt már a rejtvényt meghatározó számokat is ki kell dinamikusan rajzolni. Ennek megvalósításáért a *ShowGame.js* felel.

***INSERT SCREENSHOT HERE(GAME)***

A sorokhoz tartozó számokat úgy generálja, hogy számon tart a sor olvasásakor egy számlálót, mely nő eggyel minden **1**-esre a játékot leíró szövegben, majd nullázódik és a következő színes blokkot kezdi el számolni, ha **1**-esek után **0** jön. Ha lezárult egy színes blokk, vagy elfogyott a sor, a számláló tartalmát egy *string*-hez fűzzük, mely a táblázat adott mellett fog megjelenni, létrehozva ezzel a rejtvényt.

Az oszlopokhoz tartozó számokhoz hozzá kell adnunk egy 0. sort a táblázathoz, mely majd a generált számokat tárolja. A módszer hasonló a soroknál látotthoz, de mivel a szöveget csak soronként tudjuk olvasni, így hogy ne kelljen többször végigiterálni az adatokon, az oszlopok számlálóit és az azokat végül eltároló *string*-eket egy-egy tömbben tároljuk, hogy a soron végighaladva párhuzamosan változtathassuk azok értékeit, mintha „függőlegesen” is olvasnánk az oszlopokat.

$(document).ready(function () {

var gameString = $("#gameString").val();

var gameRows = gameString.split(";");

var cols = gameRows[0].length;

var rows = gameRows.length-1;

var firstrow = $('<tr class="game" />');

var colstrings = [];

var colBlackCtrs = [];

var firstcols = [];

var placeholdercol = $('<td class="game" />');

firstrow.append(placeholdercol);

for (j = 0; j < cols; ++j) {

colBlackCtrs[j] = 0;

colstrings[j] = "";

var newfirstcol = $('<td class="game" />');

var pre = $("<pre/>");

newfirstcol.append(pre);

firstcols[j] = pre;

firstrow.append(newfirstcol);

}

$("#gametable").append(firstrow);

for (i = 0; i < rows; ++i) {

var newrow = $('<tr class="game" />');

var firstcol = $('<td class="game" />');

var rowstring = "";

var rowBlackCtr = 0;

newrow.append(firstcol);

for (j = 0; j < cols; ++j) {

var newcol = $('<td class="game" />');

var button = $('<input type="button"/>')

.addClass('gamecell').click(function () { $(this).toggleClass('black'); });

newcol.append(button);

newrow.append(newcol);

if (gameRows[i].charAt(j) == "0") {

if (rowBlackCtr != 0) {

rowstring += rowBlackCtr + " ";

rowBlackCtr = 0;

}

if (colBlackCtrs[j] != 0) {

colstrings[j] += colBlackCtrs[j] + " ";

colBlackCtrs[j] = 0;

}

} else if (gameRows[i].charAt(j) == "1") {

++rowBlackCtr;

colBlackCtrs[j] += 1;

}

}

if (rowBlackCtr != 0)

rowstring += rowBlackCtr;

firstcol.text(rowstring);

$("#gametable").append(newrow);

}

for (j = 0; j < cols; ++j) {

if (colBlackCtrs[j] != 0)

colstrings[j] += colBlackCtrs[j];

var newlinestrings = colstrings[j].split(" ");

var puzzle = "";

for (x = 0; x < newlinestrings.length; ++x) {

puzzle += newlinestrings[x] + "\n";

}

firstcols[j].text(puzzle);

}

});

A megoldás átalakítása és beküldése a grafikus játékgenerálásnál látotthoz hasonlatos, csupán a rejtvényt adó számokat tartalmazó sorokra kell külön figyelni, hogy az első, számokat tartalmazó táblázat soron történő iterációs lépéskor ne fűzzünk a megoldást tároló *string* elejére **;**-t

# Jövőbeli tervek, bővítési lehetőségek

## Színes rejtvény

Színes rejtvények megvalósítására úgy lehetne bővíteni az alkalmazást, hogy a játékkészítési módot kiválasztó oldalon hozzáadjuk a színes rejtvény készítése opciót. Ez annyival lenne több fekte-fehér testvérénél, hogy megjelenítünk egy színválasztót is, melyből a felhasználó kikeverheti, milyen színt akar használni. Bővítjük a Site.css-t egy új osztállyal, amit a színes cellákra lehet felcsatolni, és dinamikusan színezhető. A szöveges tárolás úgy változna, hogy minden mező el van választva egy karakterrel, a mezők jellegét pedig **0** vagy **RGB** kód jelölné, sorválasztó pedig marad a **;**. A játék kirajzolásakor a rejtvény leírójából kapott színeknek megfelelő palettát helyezünk ki a felhasználónak, és a feladvány számait pedig ezekkel a színekkel jelenítjük meg, hogy megoldható legyen a színes rejtvény.

## Játékállás mentése

Menthetővé lehetne tenni a játék állását játék, és grafikus generálás közben is. Ehhez fel kellene venni egy új táblát a mentéseknek, mely tárolja a felhasználót, aki mentett; a mentett játék nevét; valamint az elmentett játékállást, a korábbiakban már részletezett szöveges formátumban. Ezek után pedig elérhetővé kell tennünk őket az oldalon (betöltés), kezelni javascriptben (ne csak hálót rajzoljon, színezzen is, ha betöltés van); megvalósítani a *controller*ekben valamint a *DBManager* lekérdezéseiben.

## Globális elérés

Jelenleg a játék csak akkor érhető el kívülről, ha valahová telepítették az általunk megadott módon, és tudjuk az adott gép IP címét, valamint 60 napig egy online szolgáltatás révén ideiglenesen elérhető. Az adatbázis is ehhez hasonló módon „vándorolhat”. Jobb lenne a rendelkezésre állás, ha ezeket globális, egységes címeken elérhetővé tennénk a jövőben.

# Telepítési útmutató

Tedd fel, ha saját adatbázist akarsz, akkor csinálj, írd át, és jó lesz. TODOKECSKEPORNÓÓÓÓÓ