

**Záródolgozat feladatkiírás**

Tanuló(k) neve[[1]](#footnote-1): Blasek Balázs, Luksa Laura, Venter Alex

Képzés: nappali munkarend

Szak: 54 213 05 Szoftverfejlesztő  
(2016-tól érvényes kerettanterv)

**A záródolgozat címe:**

**House of Swords**

Konzulens: Horváth Norbert

Beadási határidő: 2023. 04. 14.

Győr, 2023. 10. 03

**Módos Gábor**igazgató



**Konzultációs lap**

|  |  |  |  |
| --- | --- | --- | --- |
|  | A konzultáció | | Konzulens aláírása |
| ideje | témája |
| 1. | 2022.10.28. | Témaválasztás és specifikáció |  |
| 2. | 2023.02.24. | Záródolgozat készültségi fokának értékelése |  |
| 3. | 2023.04.14. | Dokumentáció véglegesítése |  |

**Értékelés**

A záródolgozat százalékos értékelése: ……

Legalább 51%-ot elérő előzetes értékelés és három igazolt konzultáció esetén a záródolgozat megfelelt.

Győr, 2023. április 14.

értékelő aláírása

**Tulajdonosi nyilatkozat**

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

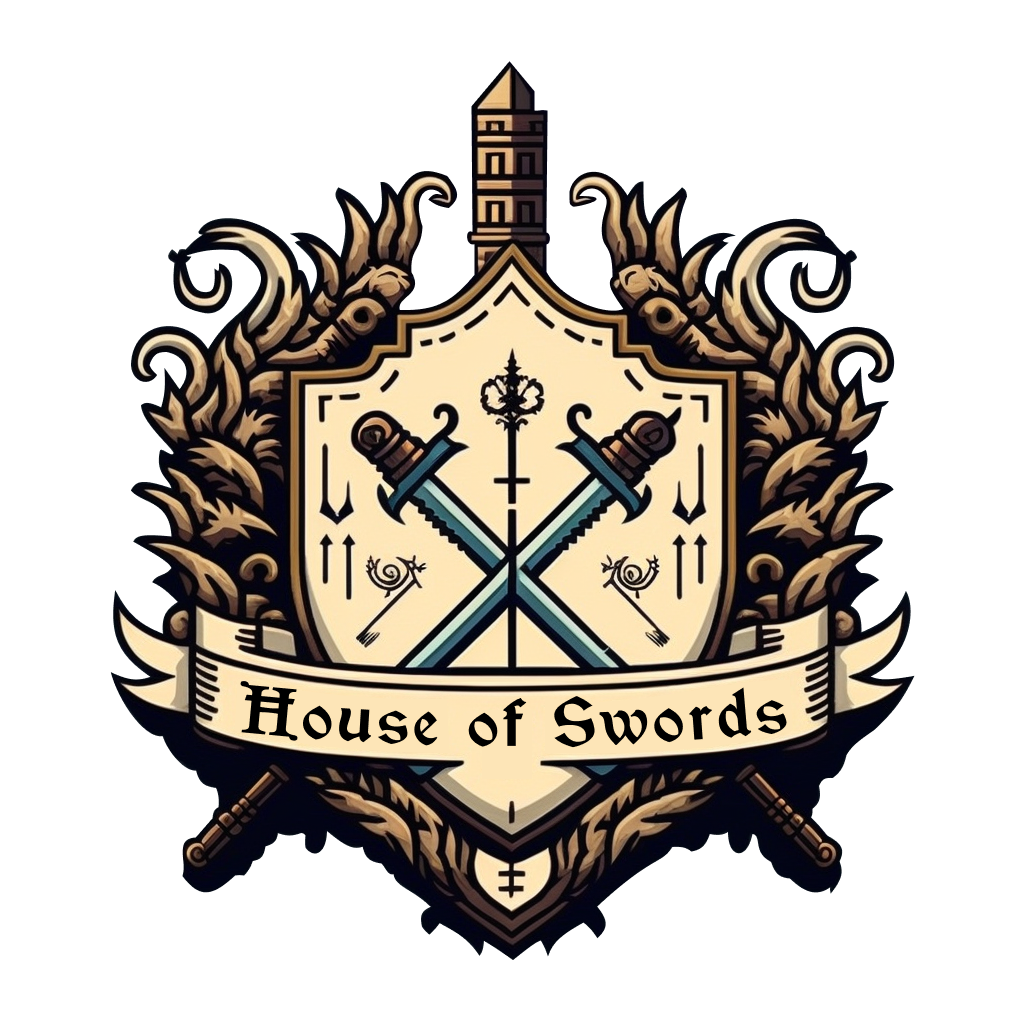
Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2023. április 14.

Blasek Balázs

Luksa Laura

Venter Alex



A House of Swords egy izgalmas középkori világban játszódó stratégiai játék, ahol a játékosoknak rafináltan kell irányítaniuk a hadseregüket és ügyesen kezelniük erőforrásaikat a győzelem érdekében..

Taralomjegyzék

1. **Specifikáció**
   1. **Játék leírása**

A játék témája: Egy középkori, többjátékos, stratégiai, idle farming játék. A játékos egy középkori város irányítója és célja minél magasabb szintre megerősíteni a városát (katonailag, tudományilag, társadalmilag).

A játék egy képzeletbeli világban játszódik, ahol az összes játékos városa található. A játékosok tudnak egymással interakcióba lépni (támadás, fosztogatás, kereskedelem, technológia-lopás, szövetség).

* 1. **Frontend megoldás**

A projekt “Frontend” része maga a Unity-s játék alkalmazás (C# programozás), illetve a szükséges adatok szerverről való lekérésének megvalósítása a kliens részére.

* 1. **Backend megoldás**

A Backend része a szerveren az adatbázis kezelése, illetve a kérések kiszolgálásának megvalósításából állna. Az adatbázist *MySQL*-lel képzeltük el egy szerveren.

A játék nem valós idejű lenne, vagyis a játékosok egymás akcióit nem közvetlenül a végrehajtások után láthatnák. Minden akció az adatbázist frissítené, amit a kliensek később lekérhetnének, de ez a megoldás nem lenne elegendő egy valós idejű kapcsolat megvalósításához, ráadásul egy ilyen játéknál nem feltétlenül lenne rá igény.

{…} – adatbázis kép

* 1. **Mobil megvalósítás**

A Unity motornak köszönhetően a játék mobil eszközökre is kompatibilis lenne, hasonló felhasználói élményt nyújtva, az asztali eszközökre fejlesztett játéktól annyi eltéréssel, hogy nem .*exe*, hanem .*apk* lenne a kiterjesztése.

* 1. **Weblap**

A játékunkat kísérné egy weblap is, amely mind mobil, mind asztali eszközökön használható lenne. Ezen az oldalon lehetne egy bemutatót találni a játékról, regisztrálni a rendszerbe, letölteni a játékot, valamint a felhasználói adatokat módosítani/törölni.

1. **Csapatmunka**
   1. **Tervek**

A projekt kezdetén felmerült a munkamegosztás kérdése. Mindhárman szeretnénk kivenni a részünket minden területen, azonban fontosnak tartottuk, hogy részekre bontsuk a feladatot. Mindannyian elvállaltunk egy egységet. Természetesen segítünk egymásnak mindenben, de így minden feladatrésznek van vezetője, aki felelősséget vállal a rész időben való megvalósításáért, minőségéért.

Alex legfőképpen a unity-vel szeretne foglalkozni, mivel korábbi projektjei alatt sok tapasztalatot szerzett benne.

Az adatbázist Balázs vállalta el. Ennek az oka, hogy mindig is érdekelte a hálózatok világa.

Laurát a weblap fejlesztés és a GUI design érdekli leginkább. A grafikai mindig is meghatározó része volt az életének, és ezt a programozással ötvözni szeretné későbbi tanulmányai során is.

* 1. **Projekt alatt**

Most, hogy nagyjából a projekt közepén tartunk, sok tapasztalatot szereztünk a közös munkák során. Szerencsésen tapasztaljuk, hogy a csapatmunka megfelelően működik. Viszont volt néhány dolog, amire nem gondoltunk a tervezésnél.

Alapvetően felosztottuk három egységre a munkát de mindegyik még mindig nagy és komplex volt. Ezt a következő módon orvosoltuk. Kisebb feladatokat, célokat tűztünk ki magunknak, hogy ezt könnyen nyomon követhessük a „*trello*” című weboldalt használtuk. Ez jelentősen megkönnyítette a munkát. Itt öt táblára osztottuk a projektet (általános, adatbázis, játék, weblap, dokumentáció).

Hogy párhuzamosan tudjunk dolgozni fontos volt a branchek használata. A githubot használtuk a weblap forráskódjának tárolására. A Unity-hez *Plastic SCM-t, (*ami egytöbbplatformos kereskedelmi elosztott verzióvezérlő eszköz) használtuk. Mind a kettővel könnyen és átláthatóan tudtuk kezelni a brancheinket.

Tapasztaltuk magunkon, hogy amikor elakadtunk valamiben több időre, nagyon jót tett egy kis környezet változás képen a Unity-ről Laravel-re és fordítva válltani, így újult erővel tudtunk nekilátni a munkának.

A kommunikációhoz *Discord*-ot használtunk. Mindannyiunk számára már korábban is ismert volt a program, így nem okozott fennakadást, tudtunk a feladatainkra koncentrálni. {…}(discord checkin – pozitívan hatottunk egymásra mivel láttuk, hogy valaki épp dolgozik)

* 1. **Visszatekintés, értékelés**

{…}

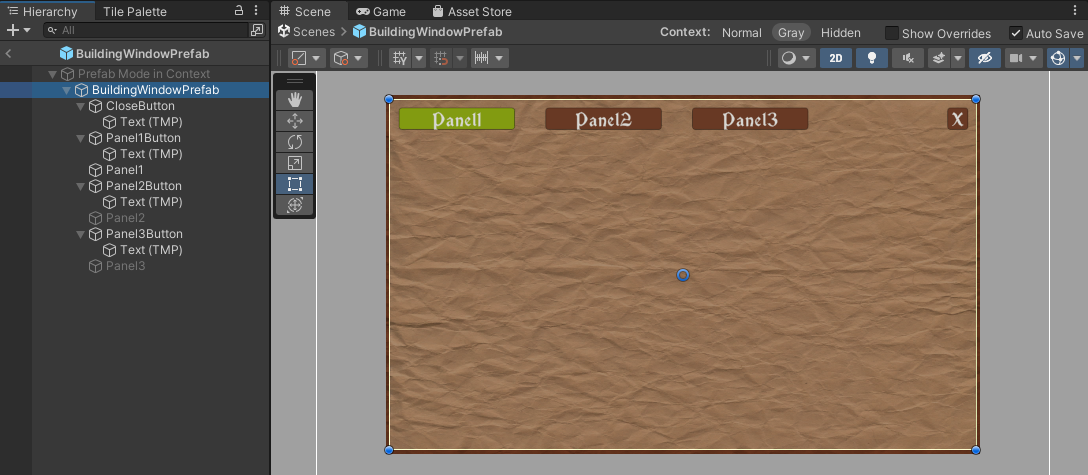
1. **Sablonok használata**
   1. **Prefabok**

A frontend kinézet kiépítését egységes sablonok, úgy nevezett „prefabok” létrehozásával kezdtük el, a játék egységes kinézete és a munkafolyamat gyorsítása érdekében. A Unity által könnyen létrehozható prefabok olyan előre elkészített játékobjektumok, amelyeket testre szabásuk után bármennyiszer felhasználhatóak. Szerkesztése esetén az összes belőle létrehozott elem frissül, ezért könnyen módosítható.

Az összes többször előforduló GUI elemre létrehoztunk egy-egy ilyen sablont. (szöveg, gomb, beviteli mező, csúszka, stb…).

Példa:

Mivel a játékban minden egyes épületnél felugrik egy ablak, ezért érdemes volt létrehoznunk erre egy prefabot:



Hogy a képen is jól látható egy prefab tartalmazhat egyéb sablonokat, ebben az esetben az ablak gombokat és paneleket. A bezáró gombok funkcióját elég volt egyszer megírni, hiszen ezt megtehettük magán a prefabon.

Egy ablak megnyitását és bezárását egy függvénnyel oldottuk meg, ami egy bekért logikai változó alapján állítja a megjelenést. Ezzel egyszerre az épületek elérhetőségét is változtatjuk.

{…} – forráskód kép

Tehát egy új *BuildingWindow* létrehozásánál már nincs egyéb dolgunk a *CloseButton*-nal, egyből neki is lehet állni a GUI felület kialakításához.

* 1. **Layouts szerkezet**

A prefabokhoz hasonlóan a laravelben is sablonokat hoztunk létre. A minden oldalon ismétlődő részeket a könnyű átláthatóság érdekében külön .*blade.php* állományokba rendeztük. Ezeket az *app.blade.php*-ba egyszerűen beimportálhatjuk @include segítségével. Ilyen fájlok a n*avigáció, lábléc,* a *css* linkek és *script*-ek beillesztése.

Ezzel így egyteljes weblapkép összeáll, tehát már csak az oldal <main> részét kell változtatnunk a route-tól függően a @yield segítségével.

1. **Webszerver**

A weboldal éles tesztelése és a játék kipróbálása miatt egy folyamatosan elérhető számítógépre volt szükségünk, hogy az adatbázis is elérhető legyen a nap bármelyik pillanatában. Erre egy *Synology DS418Play NAS (Network Attached Storage),* Linux alapú hálózati adattárolót választottunk. A webszervert telepíteni kellett a *Synology* saját linuxos csomagközpontjából, illetve a *PHP* egyes verzióit is (jelen esetben a 8.0-ás verziót használjuk, a telepítés pillanatában ez volt a legfrissebb támogatott verzió). Továbbá szükség volt még a *phpMyAdmin* és *MariaDB MySQL* adatbázis szerverre, illetve az *Apache HTTP* szerverre, melynek a 2.4-es verzióját használjuk. Tartalék szerverként a *NGINX szerver* is be lett állítva.

A megvásárolt [*houseofswords.hu*](https://houseofswords.hu/) domain címet a domain szolgáltató oldalán átirányítottuk a hálózati adattároló hálózatára, melyen az otthoni routeren a megfelelő beállításokkal ki kellett oldali a használatos portokat, jelenleg a 80, 81, 443, 3306 portokat (email szerverhez egyéb portokat is használunk).

A *NAS*-on futó webszerveren meg kellet adni a weboldalunk fő fájljához vezető útvonalat (*index.php*), illetve a portokat be kellett állítani, hogy melyeket használja. Továbbá a *HTTP szervert* és a *PHP* verzióját és engedélyeit is be kellett állítani (HTTP 2.4-es verzió, PHP 8.0-ás környezet teljes körű hozzáféréssel).

A *Laravel* keretrendszert, melyet a weboldalhoz és a backendhez egyaránt használunk, parancssorból kellett telepíteni, melyet a *PuTTY* programmal *SSH*-n keresztül értünk el (Ehhez a szokásos 20-as portot használtuk. Csak belső hálózatról érhető el a konzol).

A /usr/local/bin mappába kellett telepíteni a Composer csomagközpontot a többi program mellé a következő paranccsal:

sudo curl -s http://getcomposer.org/installer | sudo php80

Elérési út beállítása a Composer parancsnak:

sudo vi composer

Beillesztettük a fájlba az elérési utat (path):

#!/bin/bash

php80 /usr/local/bin/composer.phar $\*

sudo chmod --reference=composer.phar composer

Végül ellenőriztük a telepítés sikerességét:

composer –version

(2.4.4-es verzió)

Ezután elő kellett készíteni a weboldalt egy minta telepítésével:

composer create-project --prefer-dist laravel/laravel <appnév>

Illetve egyéb függőségeket is telepítettünk pl: php80 artisan ui bootstrap –auth (Bootstrap CSS és JavaScript keretrendszer).

Mivel a telepítés egy admin jogokkal rendelkező felhasználó alatt történt a megfelelő biztonság érdekében és hogy a HTTP felhasználó írni és olvasni is tudja a számár szükséges mappákat a mappahozzáférésnél a weboldalt tartalmazó mappának a jogosultságait és tulajdonosát módosítani kell!

Nagyon fontos, a Bootstrap cache és storage mappáinak a tulajdonosát és hozzáférési beállításait módosítani kell a megfelelő működés érdekében egy sima felhasználóra.

- sudo chown -R $USER:http bootstrap/cache

- sudo chown -R $USER:http storage

- chmod -R 775 bootstrap/cache

- chmod -R 775 storage

Hogy az NGINX tartalék webszerverrel is működjön a weboldal a következő beállítást kellett elvégezni, hogy ne csak a főoldal, hanem az egyéb fájlokra történő routolás is működjön:

sudo vi /etc/nginx/app.d/server.webstation-vhost.conf

Ebbe a fájlba hozzá kellett adni a következő sort :

location / { try\_files $uri $uri/ /index.php?$query\_string; }

Ezután újra kellett indítani az NGINX webszervert a következő paranccsal:

sudo nginx -s reload

Hasznos parancsoka cache törlésére:

- php80 artisan view:clear

- php80 artisan route:clear

- php80 artisan cache:clear

- php80 artisan config:cache

Ezzel beállításra került a NAS-on a webszerver és a Laravelles project. Viszont a weboldal frissítése és fejlesztése közben rájöttünk, hogy rendkívül macerás és nehézkes, ha a GitHubon lévő brancheket külön-külön másoljuk fel a NAS-ra. Ez kompatibilitási és adatvesztési problémákhoz vezetett és újra kellett csinálnunk néhány funkciót, mivel felülíráskor elvesztek a korábban írt sorok. Erre a problémára azt sikerült kitalálnunk, hogy a NAS-ra is telepítettük a Git-et és a beépített feladatidőzítő segítségével percenként lefuttatunk egy git pull parancsot. Ezzel tökéletesen kiküszöböltük az adatvesztést. A GitHub segítségével a különböző brancheken tudtunk dolgozni a különböző részfeladatokon, és a különbségeket a fő mappába mergeljük ami percenként fetchelve van a NAS-sal. Amint tesztelve és ellenőrizve lett egy részmechanika main branchre történő mergelés után egy percen belül láttuk is a változtatásokat a szerveren és élesben lehetett használni. Ez az automatizálás rengeteg időt megspórolt nekünk.

{…}-források, dőlt formázás

1. **Email szerver**
2. **Pathfinder**
3. **Források**

1. Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot. [↑](#footnote-ref-1)