DEBRECENI SZC BEREGSZÁSZI PÁL TECHNIKUM ÉS KOLLÉGIUM

BAG Raktárrendszer weboldal és játék

Készítette:

Fekete Ákos, Horváth Gergely, Béres Balázs Ákos

Debrecen

2024

Tartalomjegyzék

[1. Asztali játék leírása 1](#_Toc165070964)

[1.1 A játékhoz használt technológiák 1](#_Toc165070965)

[1.1.1 Unity 1](#_Toc165070966)

[1.1.2 Blender 1](#_Toc165070967)

[1.1.3 Mixamo 1](#_Toc165070968)

[1.1.4 PHP 2](#_Toc165070969)

[1.2 Felhasználói dokumentáció 2](#_Toc165070970)

[1.2.1 A főmenü gombjainak funkciói 4](#_Toc165070971)

[1.2.2 A játék menete 5](#_Toc165070972)

[1.2.3 A felhasználói interfész 7](#_Toc165070973)

[1.3 A játék kódalapú működése 9](#_Toc165070974)

[1.3.1 Kiszállítási rendszer 9](#_Toc165070975)

[1.3.2 Az ellenségek működése 14](#_Toc165070976)

[2. Weboldal és mobil alkalmazás 18](#_Toc165070977)

[2.1. A weboldalhoz, és mobil apphoz használt technológiák: 18](#_Toc165070978)

[2.1.1 HTML 5: 18](#_Toc165070979)

[2.1.2. CSS: 18](#_Toc165070980)

[2.1.3. JavaScript 18](#_Toc165070981)

[2.1.4 PHP és Laravel 19](#_Toc165070982)

[2.1.5 IntelliJ IDEA Ultimate 20](#_Toc165070983)

[2.2 Felhasználói dokumentáció: 21](#_Toc165070984)

[2.2.1 Belépés/regisztráció/elfelejtett jelszó oldalak: 21](#_Toc165070985)

[2.2.2 A főoldal 22](#_Toc165070986)

[2.2.3 Bal oldali navigációs sáv 25](#_Toc165070988)

[2.2.4 Jobb oldali navigációs sáv 25](#_Toc165070989)

[2.2.5 Rendelések oldal 26](#_Toc165070991)

[2.2.6 Felhasználók oldal 30](#_Toc165070992)

[2.2.7 Hozzáadás oldal 32](#_Toc165070995)

[2.3 A webapplikáció kód alapú működése 34](#_Toc165070996)

[3. Összefoglalás 38](#_Toc165070997)

[3.1. A játék megvalósítása és személyes tapasztalatok 38](#_Toc165070998)

# Asztali játék leírása

A játék célja, hogy egy pár órás időűző alkalmi játék legyen, amelyben a játékosok versenyezhetnek egymás ellen, a legmagasabb pontszám eléréséért.

# A játékhoz használt technológiák

## Unity

A Unity az a játékmotor, amelyet ehhez a játékhoz használtam, ezt a motort a Unity Technologies fejleszti a mai napig. Ennek a motornak a segítségével egyaránt készíthetünk három, illetve kettő dimenziós játékokat is. Nagy előnye a Unity-nek az is, hogy képes nagy méretű adatbázisokat is kezelni. Rendelkezik integrált grafikus környezettel is, a játékfejlesztés elsődleges módszereként.

A Unity-t azrét választottam, mert sok olyan Indie játékot láttam, amiket szerettek az emberek és Unity-ben voltak elkészítve, ezért meghozta a kedvem a használatához.  
Nem volt túlságosan nehéz elsajátítani az alapokat sem, hiszen rengeteg oktató video van fent az interneten, ami a kezdőknek nagyon sokat segít.

## Blender

A Blender egy háromdimenziós, szabad, nyílt forráskódú grafikai program, melynek használatával készítettem a játékban látható összes modellt, használható még animációk készítésére és 3D nyomtatáshoz is.

A Blender-t azért választottam, mert rengeteg modellt már, legyen szó animációs filmeken keresztűl a játékokon át rengeteg helyen használják, ezért ehhez is rengeteg oktató videó van, ami megint az én javamat szolgálta kezdőként.

## Mixamo

A Mixamo az egy 3D számítógépes grafikai cég, amely web-alapú szolgáltatásokat nyújt háromdimenziós karakterek animálására. Lényege, hogy a csontozási folyamatot, amely elengedhetetlen az animáláshoz automatikusan elvégzi egy pár paraméter megadása után, ami igen felhasználó barát.

A Mixamo-t azért választottam, mert szükségem volt animációra a játékhoz és kezdőként nem értettem a Blender-ben való animáláshoz ezért, a mixamo segítségével animáltam meg az ellenségeket.

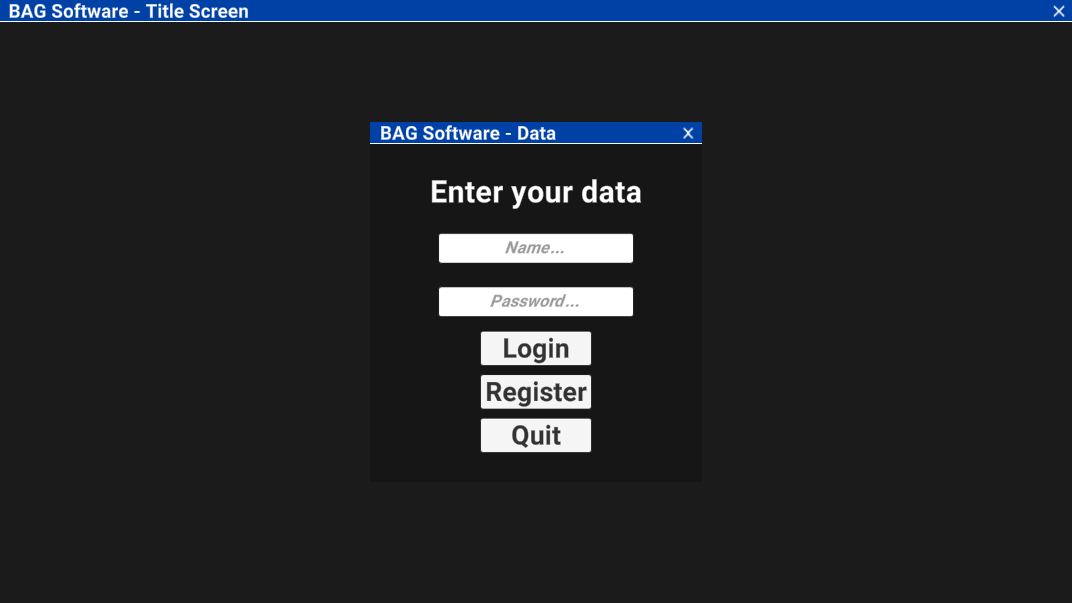
## PHP

A PHP az egy általános szerveroldali szkriptnyelv melyet dinamikus weboldalak készítésére használnak, mivel HTML oldalakba ágyazható. 1995 Rasmus Lerdorf fejlesztette ki, viszont napjainkban már a “The PHP Group” tartja fenn és fejleszti azt.

A PHP-t azért választottam a játékhoz, hogy ennek segítségével RestAPI-n keresztül tudjam a játékon belüli adatokat tárolni egy adatbázisban, így létrehozva egy felhasználói felületet, melyen a felhasználók tudnak regisztrálni, bejelentkezni és a játékon belüli pontjaikat el tudják menteni.

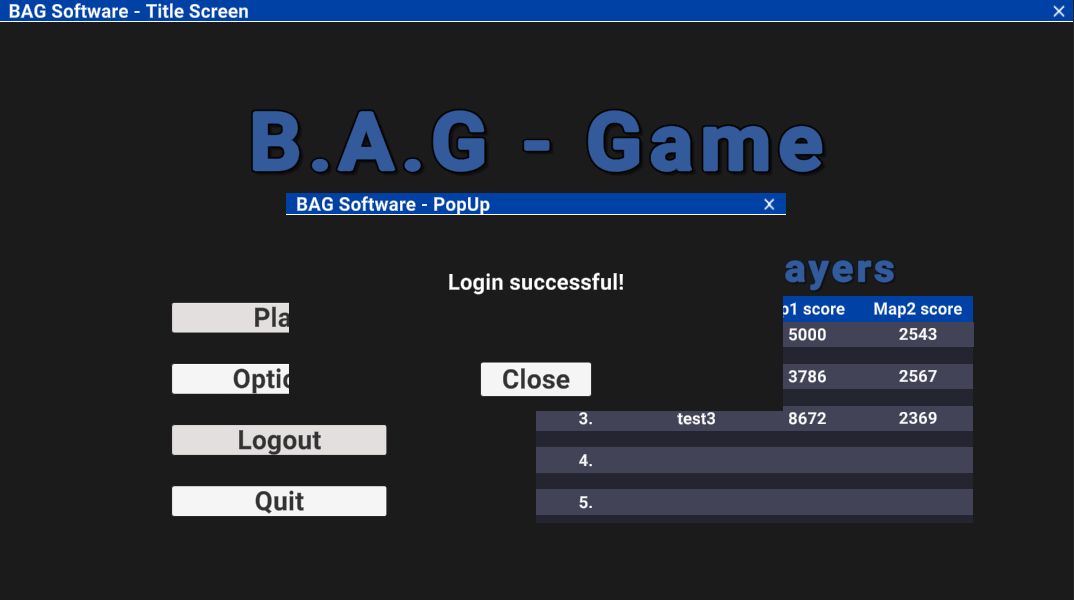
# Felhasználói dokumentáció

A játék elindítását követően a felhasználó egy a bejelentkezés és regisztrációs ablakot fog látni, ezen az ablakon kötelező készíteni-e egy fiókot, vagy egy már meglévő fiókkal bejelentkezni-e, ha ezt nem teszi meg a felhasználó nem lesz számára elérhető a játék.



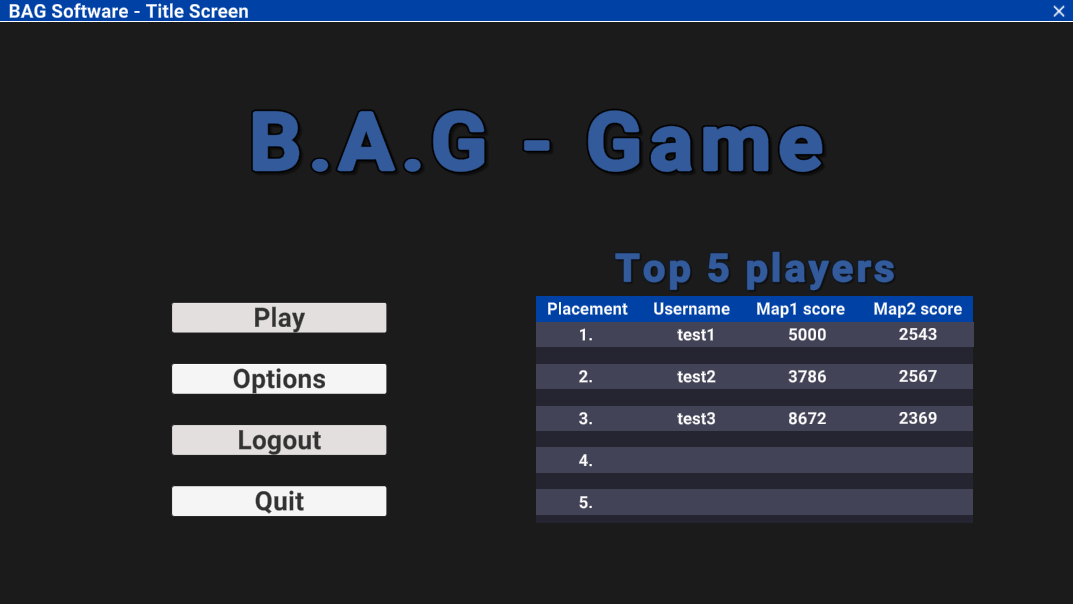
1. ábra: Bejelentkezési menü

Ezen az ablakon kell a felhasználónak kitölteni a *“Name”* és *“Password”* mezőket, majd a célnak megfelelően kell vagy a *“Register”* gombra kattintania a fiók regisztrálására vagy a *“Login”* gombra, hogy bejelentkezzen. *(1. ábra)*. Regisztráció után a felhasználónak be kell még jelentkezni, mert a *“Register”* gomb csak a fiók létrehozására szolgál. Amint ezen gombok valamelyikét a felhasználó megnyomja, megjelenik egy úgynevezett *“Pop-Up Window”* (felugró ablak), amely tájékoztatja a felhasználót arról, hogy a folyamat sikeres volt-e. *(2. ábra)*



2. ábra: Sikeres bejelentkezés esetén felugró ablak

A felugró ablakon a *"Close”* gombra kattintva el tudjuk az tüntetni. A következő ablak ezután, amit, a felhasználó lát az a *“Main Menu”* (Főmenü), ami a kép balfelső sarkában is látható. *(3. ábra)* Ezen a menüponton találhatók a következők:



3. ábra: A főmenü

* *“Play”* gomb
* *“Options”* gomb
* *“Logout”* gomb
* *“Quit”* gomb
* Egy *“Leaderboard”* amely felett a *“Top 5 players”* felirat található, ez az öt legmagasabb pontszámot elért játékos nevét és a két pályán elért pontját írja a képernyőre harminc másodperc és egy perc közötti frissítési idővel.

## A főmenü gombjainak funkciói

Elsősorban kezdjük is a *“Play”* feliratú gombbal, abban az esetben, ha erre a gombra kattintunk akkor egy következő ablakra jutunk, a *“Level Select”* nevezetű ablakra. Ezen az ablakon láthatja a játékos a három játszható pályát, amelyek a következők:

* *“Tutorial”* pálya: Ezen a pályán a játékos megtanulhatja a játék alapjait, mechanikáit és más fontos tudnivalót a játékmenettel kapcsolatban.
* A következő két pálya pedig már csak arra vár, hogy a játékos elsajátítsa a felépítésüket, megtanulja a pályán elhelyezett játékmenettel kapcsolatos tárgyak és célok helyeit, ezzel kisebb stratégiát létrehozva, a pontok maximalizálásának érdekében.

A következő gomb, ami a főmenüben található az az *“Options”* (beállítások) gomb. Itt a játékos beállíthatja az egérérzékenységét, három lebegőpontos pontossággal, emellett, ha elfelejtette volna az irányításokat, ezene a menüponton lehetősége lesz arra, hogy felfrissítse emlékezetét.

Az ezt követő gomb a főmenüben a *“Logout”* (kijelentkezés)gomb, ennek a gombnak egyszerű feladata van, hiszen csak annyit tesz, hogy a felhasználót kijelentkezteti fiókjából, majd újra megjeleníti a regisztrációs és bejelentkezési ablakot, hogy egy másik felhasználó létre tudjon hozni, egy új fiókot, vagy már egy meglévővel be tudjon jelentkezni.

Végül, található még a főmenüben, egy *“Quit”* (kilépés) gomb is, amely magától értetődően kattintáskor bezárja a játékot.

## A játék menete

A játék menete a következő módon zajlik. Amint a játékos rákattint egy adott pálya alatt *“Play”* gombra, bekerül a játékba. A pályától függ, hogy játékosunknak mi lesz a feladata.

Ha a játékos a *“Tutorial”* pályát választja, bekerül egy teszt pályára, ahol táblákról olvashatja le a játék tudnivalóit és menetét. A kezdőpálya egy kis méretű egyszerű felépítésű pálya, ezért könnyen navigálható. A táblák igen informatívak, ezért lehet, hogy az olvasás kicsit hosszadalmas, viszont fontos, hogy elolvassa a felhasználó a táblán lévő utasításokat, annak érdekében, hogy ne érezze magát elveszve a játék menete közben.

Az esetben, ha a játékos a *“Tutorial”* pályán kívül bármelyik másik pályát választja, a kettő játszható pályából, bekerül ténylegesen a játékba, ahol már a minél több pont elérése a cél és nem az alapok elsajátítása. Itt, ahogy bekerül a pályára a játékos, kap a képernyője tetején egy *“Objective: Deliver the last box..”* feliratú küldetést, ezt követően az a feladata a játékosnak, hogy a tőle jobb oldalt lévő szekrényekre beérkező dobozt kiszállítsa a megfelelő célhelyre, amit a játék egy *“Deliver Here!”* feliratú küldetésjelző ablakkal fog jelezni, ami a célhely pontos helyét mutatja a pályán.

Ezt követően, ha a játékos kiszállítja a dobozt a megfelelő helyére, az az beledobja az elsötétített nyílásban ténylegesen elkezdődik a játék. Ekkor a játék története szerint az elhagyatott épületben a kiszállítási rendszer újraindul, ezzel a környéken lévő zombik nyugalmát megzavarva.

Ezek után a játékos *“UI”* (felhasználói interfészen) megjelenik egy visszafelé számláló időzítő és egy doboz számláló szöveg 0/5 vagy 0/10 formátumban, ami a játék nehézségétől függ. Ekkor a felhasználó feladata igen egyszerű, vissza kell rohannia a pálya kezdetéhez és az ott polcokra érkező dobozokat ki kell szállítania a megadott időkorláton belül amilyen gyorsan csak tudja.

A visszaszámláló és a dobozszámláló a következő képen működik:

* Ha a játékos jó helyre szállít ki egy dobozt, a számláló értéke növekedik eggyel a visszaszámláló pedig szintén növekedi, mivel dobozonként jár több idő az egész kiszállításra, emellett a játékost dobozonként (ha a kiszállítási hely megfelelő), jutalmazza a játék pontokkal (a képernyő balalsó sarkában látható) és fizetőeszközzel (a képernyő jobbalsó sarkában látható).
* Ellen esetben, ha a játékos a dobozt rossz helyre szállítja ki, pontot fog veszíteni a meglévő pontjaiból, nem kap fizetőeszközt és további kiszállítási időben sem részesül.

A játékosnak kiszállítás mellett más módja is van fizetőeszköz beszerzésére, mint például küzdelemmel a zombikkal szemben.

Küzdelem a zombikkal a következő képen zajlik:

* Ha a játékos nekidob egy dobozt a zombinak a zombi reagál erre és akció képtelenné válik egy rövid ideig, elég időt adva, hogy a zombiról visszapattanó dobozt e játékos elkapja és még egyszer megdobja vele a zombit.
* A zombi halálakor pedig növel a játék egy a játékos számára rejtett ölésszámlálót és fizetőeszköz jutalomban is részesül.

Bizonyos ölésszám után pedig elérhetővé válik a játékos számára egy *“Upgrade Shop”* (fejlesztési bolt)menü, ami automatikusan megnyílik, akárhányszor eléri azt a számot.  
Itt a felhasználó a játék menete közben szerzett fizetőeszközét tudja elkölteni fejlesztésekre. A fejlesztések szintjének nincs határa, viszont áruk folyamatosan emelkedik vásárlásonként, emellett áruk nem közös, viszont minden fejlesztés ára azonos mennyiséggel növekszik. A fejlesztések még emellett egyszeri fejlesztések, ami azt jelenti, hogy egy kör végén elvesznek és egy új körben újra kell őket vásárolni.

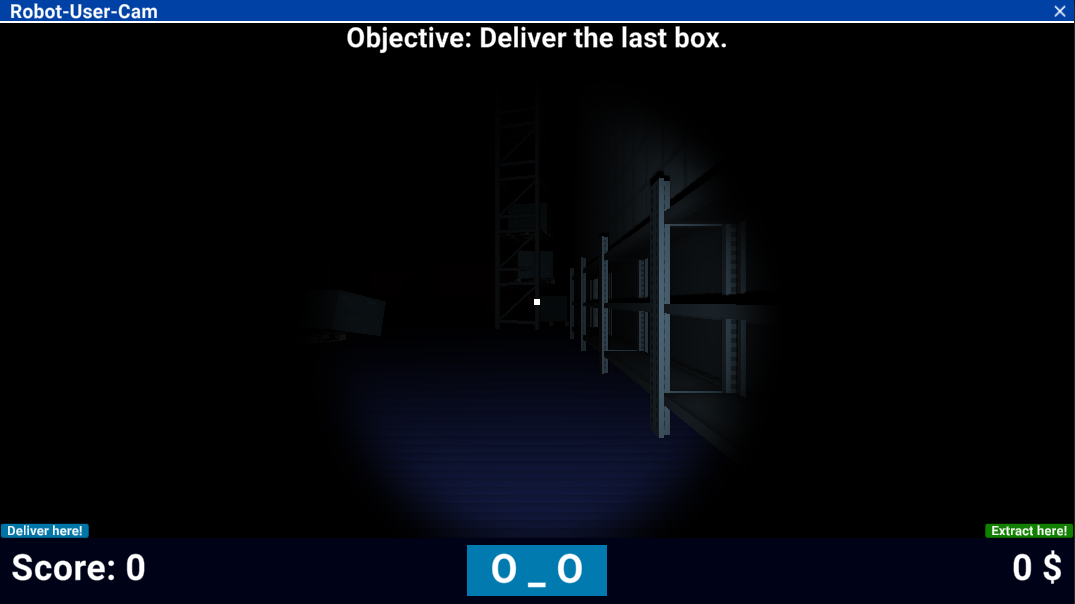
A fejlesztések a következők:

* *“Speed”*: Ez növeli a játékosnak az alap sebességét, ezzel megkönnyítve a pályán való közlekedést, vagy nehezítse azt, ha a játékos túl magasra fejleszti azt ki.
* *“Strength”*: Ez a fejlesztés a játékos dobási erejét növeli, annak érdekében, hogy a játékos könnyű szerrel tudja a dobozokat közelebb juttatni a célhoz, és hogy messzebbről tudjon végezni a zombikkal, hiszen minél erősebben dob a játékos annál pontosabban ér célba a doboz.
* *“Health”*: A *“Health”* fejlesztés a játékos maximális életerejét fogja növelni, ennek hatására több sebzést lesz képes kibírni. Ezen felül a játékos életét a maximális életének ötven százalékával megfelelő mennyiséggel is visszatölti, hiszen nincsen a játékban más mód az életerő visszaszerzésére.

Végezetül, ha eltelt egy bizonyos idő elérhetővé válik egy rövid időre az úgynevezett *“Extraction”* (Evakuációs) pont.

Az evakuációs pont lényege, hogy elmentse a pályán elért pontokat rekord ként. Ha a játékos elveszíti a játékot mielőtt képes lenne evakuálni a pályáról minden elért pontja elveszik és kezdheti a körét az elejétől.

## A felhasználói interfész



4. ábra: Heads Up Display

A *”Heads Up Display”* (HUD) mutatja a játékban a fontos információkat a játékos számára. *(4. ábra)* Részei a következők:

* *“Objective”*: Ez írja le az éppen aktuális feladatát, a játékosnak, de igazán csak egy sejtelmes felírat, ami csak sugallja azokat. Ez a felírat a játék futása során egy alkalommal fog megváltozni.
* *“Score”*: Ez a felírat minden alkalommal frissül, amikor a játékos pontot szerez és feladata, az, hogy a játékossal közölje az általa elért összes pontszámát.
* *“O \_ O”*: Ez az arc reprezentálja a játékos életpontjait, ha a játékos valamilyen módon megsérül a kék színű háttér fentről lefelé elkezd csökkenni és bizonyos százaléknyi életerő alatt az arc meg is változik.
* *“Deliver here!”*: Ez a felírat játék közben el van rejtve, akkor jelenik meg, amikor a játékos felvesz egy dobozt, akkor a felirat arra pozícióra ugrik, ahova ki kell a dobozt szállítani. Ha a dobozt a játékos eldobja, vagy leteszi ez a felírat újra eltűnik.
* *“Extract here!”*: Az evakuációs felírat akkor jelenik meg a képernyőn, ha a játékban teljesül a feltétele, ellenben mindig el van rejtve felhasználó elől.

Vannak olyan részei is a HUD-nak, melyek csak kiszállítási időben jellenek meg *(5. ábra)*:



5. ábra: Heads Up Display kiszállítás közben

* *“0/5”*: Ez a kiszállítási kvóta, ha ezt a számot nem üti meg a felhasználó az idő lejárta előtt, akkor veszít a meglévő fizetőeszközéből, pontjaiból és a még nem kiszállított dobozok is megsemmisülnek, annak érdekében, hogy ne lehessen előre halmozni dobozokat több kiszállítási időszakokra.
* *“155,12”*: Ez a kiszállításra szánt idő, amely egy visszaszámláló, ha ez az idő nullát üt akkor az itt felsorolt két HUD elem eltűnik.

# A játék kódalapú működése

## Kiszállítási rendszer

A kiszállítási rendszer a következő képpen működik:

private void Start()

{

minDeliveryLength = minDeliveryLength \* Difficulty.timeMultiplier;

maxDeliveryLength = maxDeliveryLength \* Difficulty.timeMultiplier;

table = GameObject.FindGameObjectsWithTag("Table");

rndSpawn = Random.Range(0, table.Length);

boxCount = Difficulty.boxPerWave;

ScoreManager.maxDelivery = boxCount;

StartCoroutine(SpawnLocation(boxSpawnInterval));

}

A *“void Start”* az a funkció, amely a játék indításakor fog lefutni kizárólag egyszer! Itt a kezdőértékeket szokás először beállítani. Itt található a *“minDeliveryLength”* változó a minimum kiszállítási időt tárolja, a *“maxDeliveryLength”* változó pedig logikusan a maximális kiszállítási időt tárolja, ezt a két értéket pedig beszorozzuk a *“Difficulty”* szkript-ben lévő idő többszörözővel.  
Ezt követően megkeressük a *“table”* tömbbe a *“Table”* Tag-el ellátott játékelemeket, azért, mert a dobozok ezeknek az elemeknek a pozícióit használja a szkript a dobozok létrehozásakor.

E-mellett létezik egy*“rndSpawn”* változó is, ami kap egy véletlen szerű számot a *“table”* tömb változó határain belül, így lehetővé teszi azt, hogy a dobozok különböző helyeken is létre tudjanak jönni.  
A következő változó*“boxCount-nál”* beállítjuk nehézségtől függően, hogy összesen mennyi doboz lehet egy kiszállítás során.  
Végül pedig elindítunk egy *”Coroutine”*-t *“SpawnLocation”* néven egy “boxSpawnInterval” változóval, amely így néz ki:

private IEnumerator SpawnLocation(float time)

{

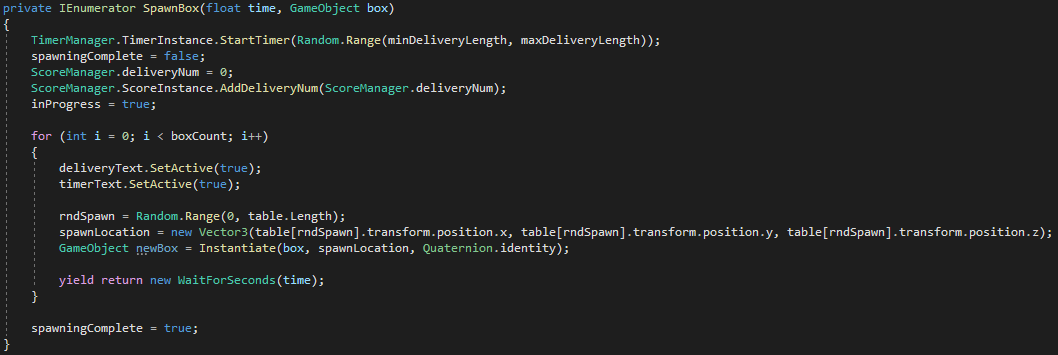
yield return new WaitForSeconds(time);

rndSpawn = Random.Range(0, table.Length);

StartCoroutine(SpawnLocation(boxSpawnInterval));

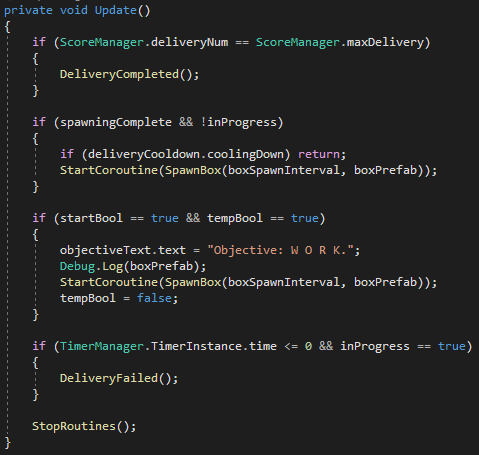
}

A *”Coroutine”*-t abban az esetben használjuk, ha valamit szeretnénk többször lefuttatni, de viszont nincs szükségünk arra, hogy az adott funkció gyorsan és minden frame-ben lefusson. Ebben az esetben vár egy kevés időt a dobozok létrehozási helyének a kiválasztásánál a játék nehogy azok egymásban jöjjenek létre. Végül pedig elindítjuk önmagában még egyszer, hogy a játék végéig fusson, vagy míg valami le nem állítja azt.



6. ábra: Doboz létrehozó ”Coroutine”

A következő ábrán a doboz létrehozásának *”Coroutine”*-ja látható *(6. ábra)*. Itt első lépésben indítunk egy időzítőt, amelynek feladata lesz visszaszámolni a *”minDeliveryLength”* és *”maxDeliveryLength”* közötti véletlenszerűen kiválasztott értékből. Utána hamisra állítjuk a *”spawningComplete”* változót hiszen még nem vagyunk kész a dobozok létrehozásával. Beállítjuk a *”ScoreManage.deliveryNum”* nevű változót nullára, mivel még nem szállíthattunk ki dobozt, mert még nem jöttek létre. Azt követő sorral lefrissítjük a kiszállítások számát. Majd igazra állítjuk az inProgress nevű változót, mert folyamatban lesz a kiszállítás, ha elkezdenek létrejönni a dobozok.



7. ábra: Kiszállítás szkript Update funkciója

Ezen az ábrán láható az *“Update”*, ez kezeli a játékban az egész kiszállításnak a többszöri lefutását, hogy a játék az örökkévalóságig tudjon futni *(7. ábra)*. Négy különböző feltételünk van és azért nincsenek összekötve, mert így biztosabban különbséget tud tenni a kód melyik esetben melyik fusson le.

Az első feltétel azt nézi, hogy megegyezik-e a kiszállított dobozok száma *“ScoreManager.deliveryNum”* a kiszállítási kvótával *“ScoreManager.maxDelivery”*. Abban az esetben, ha ez igaz végrehajtódik a *“DeliveryCompleted();”* funkció, melynek az a feladata, hogy a képernyőn megjelenő időzítőt és kvótaszámlálót elrejtse, és állítson be egy késleltetést, hogy legyen a játékosnak ideje visszaérni-e a kiszállítandó dobozok helyéhez.

A második feltétel azért felelős, hogy ha a dobozok létrejöttek *“spawningComplete”*, és nincsen éppen aktív kiszállítás *“!inProgress”*, akkor hozzon létre még dobozokat a megadott időközönként.

A harmadik feltétel a játék kezdetét kezeli, ha a kezdési bool *“startBool”* igaz, és az ideiglenes bool is *“tempBool”*, abban az esetben kezdődjön el a játék (a *“tempBool”* azért szükséges, hogy ez csak egyszer fusson le).

Az utolsó feltétel pedig végül azt nézi meg, hogy kifutottál-e az időböl *“TimerManager.TimerInstance.time <= 0”* és hogy folyamatban van-e egy kiszállítás *“inProgress == true”*, ha igen akkor a kód végrehajtja a *“DeliveryFailed();”* funkciót, ennek feladata, hogy megállítsa az időzítőt, a kiszállítási folyamatot, elrejtse az időzítőt és a kvóta számlálót, vonjon le pontot és fizetőeszközt, dobassa el a játékossal a dobozt, ha van a kezében, mert ez problémákat okozhat, emellett keresse meg az összes ki nem szállított dobozt és törölje ki azokat, majd indítson el egy kiszállítási késleltetést.

private void DeliveryFailed()

{

TimerManager.TimerInstance.StopTimer();

deliveryText.SetActive(false);

timerText.SetActive(false);

inProgress = false;

int moneyLoss = -Random.Range(50, 350);

ScoreManager.ScoreInstance.AddCred(moneyLoss);

PickUp.pickedUp = false;

boxes = GameObject.FindGameObjectsWithTag("Box");

for (int i = 0; i < boxes.Length; i++)

{

if (boxes[i].GetComponent<BoxProperties>() != null)

{

Destroy(boxes[i]);

}

}

deliveryCooldown.StartCooldown();

}

Az ellenségek létrehozását nem részletezném hiszen majdnem ugyan ezen az elven működik annyi különbséggel, hogy kiszállítástól független jönnek létre az ellenségek, és számok folyamatosan növekszik.

## Az ellenségek működése

Az ellenség egy általános úgynevezett *“State Machine”*-t használ, amivel tud váltakozni viselkedési fázisok között. A viselkedési fázisok használata lehetővé teszi azt, hogy minden viselkedési fázist külön szkriptben hozzunk létre, így könnyen bővíthető vagy csökkenthető a viselkedési állapotok száma.

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

8. ábra: AiStateMachine felépítése

Először, egy létrehozott class-ban létre hozok egy tömböt, amiben letároljuk az összes létező állapotot, majd megadunk egy *“AiAgent”* alapú változót, amely a Unity-ben egy karakter által végrehajtható lépések halmaza, és a megszüntetési feltételeinek listája *(8. ábra)*.

* Az első funkció feladata, hogy az adott játék elem agent-je legyen egyenlő az ebben a szkriptben talált agent-el, majd a *“states”* tömböt töltse fel annyi fázissal, ammenyi darab van összesen.
* A *“RegisterState()”* funkcióval lehet több fázist felvenni.
* A *“AiState GetState()”* funkció segítségével letudjuk kérni az index-el megegyező számon lévő fázist.
* Update-on belül megnézzük az aktuális fázisát az ellenségnek és ha változás van akkor frissítjük azt.
* A *“ChangeState()”* funkción belül pedig megnézzük, hogy az agent befejezte-e, a fázist, amiben éppen volt *“GetState(currentState)?.Exit(agent);”*, ha igen akkor kérünk egy új fázist és beléptetjük az agent-et abba a fázisba *“currentState = newState;*  
  *GetState(currentState)?.Enter(agent);”*.

Az álltalam használt fázisok listája:

public enum AiStateId

{

Chase,

Wander,

Search,

Attack,

Stunned,

Idle

}

Végül pedig létre kell hoznunk egy *” interface”*-t, mely annyit tesz, hogy unifrmizálja a fázisok felépítését, így minden fázisnak kötelezően rendelkeznie kell ezekkel a funkciókkal.

public interface AiState

{

AiStateId GetId();

void Enter(AiAgent agent);

void Update(AiAgent agent);

void Exit(AiAgent agent);

}

* *“void Enter(AiAgent agent);”*: Az ebbe a részbe írt kód akkor fut le, ha az agent, belép ebbe a fázisba. Hasonlít a *“void Start”-*hoz. Itt lehet érdemes kezdőértékeket beállítani a fázison belüli *“Update”* számára.
* *“void Update(AiAgent agent);”*: Amíg az adott fázisban van az agent az itt beírt kód fog lefutni addig amíg valami arra nem kényszeríti, hogy kilépjen az adott fázisból.
* *“void Exit(AiAgent agent);”*: Az ideírt kód abban az esetben fut le, ha az adott fázisból az agent valami ok folytán kilép.

# Weboldal és mobil alkalmazás

A weboldal célja, hogy van egy raktárrendszer, és ahhoz van egy admin kezelőrendszer. Lehet látni egyes csomagokat, azoknak a kiszállítási státuszát, illetve meg is lehet változtatni a státuszt. Ezen felül meg lehet tekinteni az alkalmazottakat, azok személyes adatait. Amennyiben valaki elfelejti a belépési jelszavát, a kapcsolattartói e-mail címre küld egy jelszó helyreállítási e-mailt a rendszer, amivel vissza tudja állítani a jelszót.

## A weboldalhoz, és mobil apphoz használt technológiák:

### HTML 5:

HTML 5 a legújabb verziója a Hypertext Markup Language-nak, ami az internetes oldalak struktúráját határozza meg. Az HTML5 számos új elemet és attribútumot vezetett be, amelyek támogatják a multimédia és a grafikus tartalmak integrálását közvetlenül az HTML kódba. Így például az <audio> és <video> elemek lehetővé teszik a médiafájlok könnyű beágyazását, míg az új szemantikus elemek, mint a <article>, <section>, <nav>, és <footer> segítenek jobban strukturálni az oldalakat és elősegítik a jobb SEO (keresőoptimalizálás) és webes hozzáférhetőség eredményeit.

### 2.1.2. CSS:

CSS (Cascading Style Sheets) az a nyelv, amely meghatározza, hogyan jelenjenek meg az HTML dokumentumok elemei a képernyőn. CSS3, a CSS legújabb verziója, magában foglal számos fejlesztést, mint például a rugalmas dobozok (flexbox), rácsrendszerek (grid systems), átmenetek (transitions), animációk, és a válaszadó design elemeket, amelyek kulcsfontosságúak a modern webfejlesztésben. Ezek a funkciók lehetővé teszik a fejlesztők számára, hogy dinamikusabb és vizuálisan vonzóbb weboldalakat hozzanak létre.

### 2.1.3. JavaScript

JavaScript egy dinamikus programozási nyelv, amelyet széles körben használnak az interaktív weboldalak fejlesztésére. JavaScript segítségével a fejlesztők hozzáadhatnak dinamikus funkciókat a weboldalakhoz, mint például animációk, felugró ablakok, űrlap-validációk, és még sok más. JavaScript modern keretrendszerei, mint az Angular, React és Vue.js, tovább emelik a webfejlesztés hatékonyságát és összetettségét, lehetővé téve gazdag felhasználói élmények létrehozását.

### PHP és Laravel

PHP (Hypertext Preprocessor) egy széles körben használt szerveroldali szkriptnyelv, amelyet kifejezetten webfejlesztéshez terveztek. A PHP előnye, hogy könnyen tanulható és rugalmas, támogatást nyújt a legtöbb webes szerverhez és adatbáziskezelő rendszerhez, így ideális választás dinamikus weboldalak és online alkalmazások létrehozásához.

Laravel egy modern PHP keretrendszer, amelyet Taylor Otwell hozott létre, és célja, hogy a webalkalmazás-fejlesztés folyamatát egyszerűbbé és elegánsabbá tegye. Laravel egyike a PHP közösség legnépszerűbb és legaktívabb keretrendszereinek, amely a kifejezetten tiszta, egyszerű és olvasható kód írását teszi lehetővé.

Kulcsjellemzők és előnyök:

1. MVC architektúra: Laravel az MVC (Model-View-Controller) architektúrát követi, amely elősegíti az alkalmazás logikának és a felhasználói felületnek a szeparálását. Ez növeli az alkalmazás karbantarthatóságát és bővíthetőségét.
2. Blade sablonmotor: Laravel egyik kiemelkedő funkciója a Blade sablonmotor, amely intuitív és rugalmas. Blade lehetővé teszi az adatok zökkenőmentes integrálását a HTML-be, miközben tiszta és rendezett kódot tart fenn.
3. Eloquent ORM: Laravel saját objektum-relációs leképezője (ORM), az Eloquent, egyszerűvé teszi az adatbázis-műveletek végrehajtását. Az Eloquent használatával a fejlesztők PHP kód segítségével képesek interaktálni az adatbázissal, így elkerülhető a bonyolult SQL lekérdezések írása.
4. Migrációk és seederek: Laravel migrációs rendszere segít a verziókontrollált adatbázis-sémák kezelésében, lehetővé téve az adatbázis-séma megosztását és alkalmazását a fejlesztési csapat minden tagja között anélkül, hogy manuálisan kellene frissíteni az adatbázis-struktúrát.
5. Teljeskörű biztonság: Laravel külön figyelmet fordít a biztonságra, beépített védelmet nyújtva az SQL-injekció, a keresztoldali szkriptelés (XSS) és a keresztoldali webhely-kérés hamisítás (CSRF) ellen. Ezek a funkciók automatikusan aktívak, így a fejlesztők kevesebb időt kell hogy fordítsanak a biztonsági aggodalmak kezelésére.

### IntelliJ IDEA Ultimate

IntelliJ IDEA Ultimate egy fejlett integrált fejlesztői környezet (IDE) a JetBrains-től, amely támogatást nyújt Java, Kotlin, PHP, és számos más programozási nyelv fejlesztéséhez. Az IntelliJ IDEA Ultimate kiadása számos speciális funkciót kínál, mint például a webfejlesztési keretrendszerek és technológiák támogatása, adatbázis-kezelés és verziókövetés integráció. Az IDE kiváló eszközöket kínál kódanalízishez, refaktoráláshoz és hibakereséshez, amelyek jelentősen javítják a fejlesztési folyamatok hatékonyságát.

## Felhasználói dokumentáció:

### 2.2.1 Belépés/regisztráció/elfelejtett jelszó oldalak:

A weboldal betöltése után az admin panelbe bejelentkezést kér (egy felhasználónevet és egy jelszót). A jelszó az adatbázisban titkosítva van tárolva, ezzel növelve a felhasználók biztonságát. Még be lehet pipálni egy olyat, hogy ’Emlékezz rám’, amivel megjegyzi az app a böngészőt, és nem fog kijelentkezésig kérni belépési adatokat. Kötelező bejelentkezni, anélkül nem lehet továbbmenni a főoldalra, ahol mindent lehet kezelni, és megtekinteni. (9. ábra)

A képen szöveg, képernyőkép, Betűtípus, Márka látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, embléma látható

Automatikusan generált leírásHa nincs még a felhasználónak fiókja, akkor van egy gomb, amivel regisztrálni lehet, megadva a teljes nevét, egy felhasználónevet, az e-mail címét, telefonszámát, és egy jelszót, amit meg kell ismételni, ezzel elkerülve az elírást. Ezzel a felhasználónévvel lehet később bejelentkezni. (10. ábra)

A bejelentkezés részen még található egy elfelejtett jelszó rész is, ahol a kapcsolattartó e-mail címet beírva kap a felhasználó egy linket, amivel vissza tudja állítani a jelszavát. (11. ábra). Amennyiben létezik az adatbázisban a megadott e-mail cím, akkor kiküldi a A képen szöveg, képernyőkép, Betűtípus, embléma látható

Automatikusan generált leíráscímre a jelszó helyreállító linket, és e-mait. A képen szöveg, képernyőkép, Betűtípus, embléma látható

Automatikusan generált leírás(12-13. ábra)

9. ábra: Bejelentkezési felület

10. ábra: Regisztrációs felület

12. ábra: Jelszó helyreállítása oldal sikeres oldalkitöltés esetén

11. ábra: Jelszó helyreállítása oldal

### A képen szöveg, képernyőkép, Betűtípus látható Automatikusan generált leírás2.2.2 A főoldal

13. ábra: A jelszót helyreállító e-mail

Sikeresen belépve az oldal ide irányít át bennünket, ahol láthatunk több mindent is. Bal felső sarokban látható a cégnek a logója, amire ha kattintunk, kinyit egy oldalsó menüt, aminek a segítségével navigálhatunk az oldalon *(14. ábra)*. Az oldal közepén található egy kis leírás, hogy hogyan navigálhatunk az oldalon.

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírásAz oldal jobb oldalához érve látható egy felhasználó ikon (egy szürke ember fej), és ha arra rámegyünk kinyit egy navigációs sávot, ahol elvégezhetünk több dolgot is *(15. ábra)*.

15. ábra: Jobb oldali menü

14. ábra: Bal oldali menü

### 2.2.3 Bal oldali navigációs sáv

A bal oldali navigációs sávra visszatérve, akkor jelenik meg, ha rámegyünk a bal oldali raktárrendszer logóra, a weboldal tetején. Miután megnyílt, láthatunk több opciót is:

* 1. - Főoldal
  2. - Rendelések
  3. - Felhasználók
  4. - Hozzáadás

Ezek különböző funkcióval ellátott, raktárrendszert kezelő részek, ahova ez a kis sáv navigálhat el minket. Bezárni ezt a menüt úgy lehet, hogy rányomunk a szürke ’X’ gombra, ezáltal bezáródik az oldalsó menü.

### Jobb oldali navigációs sáv

A jobb oldali navigációs sáv nagyon hasznos a bal oldali után. A menü kinyitása után láthatunk több dolgot is:

1. - A sötétebb részen látszik ugyanaz a kép, amit a menü megnyitása előtt láttunk.
2. - Mellette látjuk annak a fióknak a teljes nevét, amelyikkel bejelentkeztünk.
3. - Alatta a világosabb részen van 3 darab gomb.
4. - Az első a Személyes adatok
5. - A második a Jelszó módosítása
6. - A harmadik pedig a kijelentkezés

Ha rámegyünk a személyes adatokra, látjuk az összes olyan információt, amit megadtunk az oldalra való regisztráláskor, valamint látjuk a hatáskörét a belépett fióknak. Alatta látható egy kiírás, hogy amennyiben valamelyik adat nem stimmel, akkor kattintsunk ’ide’. Ha rámegyünk erre az írásra, átirányít az adatok módosításához. Bezárni ezt az ablakot a ’Bezárás’ gombbal tudjuk *(16. ábra)*.

Ezen gomb alatt találhatjuk meg a jelszó módosítását, amire ha rámegyünk bedob egy kis ablakot, animációval, és itt meg tudjuk változtatni a régi jelszavunkat valami újra. 3 darab beviteli mező található itt. ’Régi jelszó’, ’Új jelszó’, és ’Új jelszó még egyszer’. Itt amennyiben megegyezik a régi jelszó a belépéshez használttal, és az új jelszó mind a két helyen megegyezik, meg fogja a program változtatni a jelszót. Ha véletlen nyitottuk meg ezt az ablakot, akkor csak kattintsunk a ’Bezárás’ gombra, viszont ha szeretnénk véghez vinni a módosításokat, akkor nyomjunk a ’Küldés’ gombra. Ezt követően bezáródik ez a Form *(17. ábra)*.

A képen szöveg, képernyőkép, szám, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírásA jelszó módosítása alatti gomb a kijelentkezés, ami nevéből adódóan kijelentkezteti az adott felhasználót, és visszadobja a bejelentkezési részhez. Bezárni ezt az oldalsó menüt úgy tudjuk, hogy rámegyünk a szürke ’X’ gombra.

### 

17. ábra: Jelszó módosítása Modal

16. ábra: Személyes adatok Modal

### 2.2.5 Rendelések oldal

Mikor ezt az oldalt meglátjuk, látható egy táblázat, és benne adatokkal, némelyik végén gomb is található, amelyek különböző funkciókat látnak el. Ennek az oldalnak a lényege az adatok ellenőrzése, a rendelések megtekintése, elfogadása, elutasítása, és szükség esetén módosítása. *(18. ábra)*

1. Rendelések Listázása
   1. Funkció: **Rendelések megjelenítése egy listában, ahol lehetőség van különböző műveletek végrehajtására.**
   2. Felület: **Egy táblázat, ahol a rendelések adatai láthatóak.**
   3. Elemei:
      1. Rendelések - **az oldal címe.**
      2. Szűréses keresés: **szűrési lehetőségek a rendelések között.**
      3. Táblázat oszlopai: **ID, Áru neve, Áruszolgáltató, Jelenleg raktáron (db), Rendelt (db), Ár (1x), Rendelés dátuma, Rendelés állapota.**
      4. Műveletek: **Ikonok a rendelések törlésére, módosítására és elfogadására.**
2. Rendelés Visszavonása *(19. ábra)*
   1. Funkció: **Lehetőség van egy meglévő rendelés törlésére.**
   2. Felület: **A felhasználó egy pop-up ablakban kap megerősítést a rendelés törléséhez.**
   3. Elemei:
      1. Rendelés visszavonása - **ablak címe.**
      2. Szöveg: **"Biztos szeretnéd törölni az alábbi rendelést?"**
      3. ID: **a rendelés azonosítója (ezen az ablakon 0).**
      4. Név: **a termék neve (Színváltós lámpa).**
      5. Rendelt mennyiség: **megrendelt darabszám (20 db).**
      6. Gombok: **Visszavonás (piros), ami a törlést indítja, és Vissza (kék), ami visszalépést tesz lehetővé.**
3. Rendelés Módosítása *(20. ábra)*
   1. Funkció: **Meglévő rendelés részleteinek módosítása.**
   2. Felület: **Egy modális ablak, ahol a felhasználó módosíthatja a rendelés adatait.**
   3. Elemei:
      1. Rendelés módosítása - **ablak címe.**
      2. Jelenleg kiválasztott rendelés: **ID és termék neve.**
      3. Rendelés státusza: állapotváltás gomb.
      4. Mennyiség: **a rendelt darabszám beállítása.**
      5. Áruszolgáltató: **a terméket szállító cég neve (Alza.hu).**
      6. Ár: **kiszámított teljes ár.**
      7. Rendelés dátuma: **a rendelés kezdetének dátuma.**
      8. Gomb: **Mentés (zöld), amivel a módosításokat lehet menteni, és Vissza (kék), ami bezárja a pop-up ablakot.**
4. Rendelés Elfogadása *(21. ábra)*
   1. Funkció: **Új vagy meglévő rendelés elfogadása.**
   2. Felület: **Egy pop-up ablak, ahol a felhasználó elfogadhat egy rendelést.**
   3. Elemei:
      1. **Rendelés elfogadása - ablak címe.**
      2. Szöveg: **"Biztos elfogadod az alábbi rendelést?"**
      3. ID: **a rendelés azonosítója.**
      4. Név: **a termék neve.**
      5. Rendelt mennyiség: **megrendelt darabszám.**
      6. Gombok: **Elfogadás (zöld), ami a rendelés elfogadását hajtja végre, és Vissza (kék), az pedig szintén visszaléptet a modal megnyitása előtti állapothoz.**

**A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás**

19. ábra: Rendelés visszavonása Modal

21. ábra: Rendelés elfogadása Modal

18. ábra: Rendelések oldal

**A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás**

20. ábra: Rendelés módosítása Modal

### 2.2.6 Felhasználók oldal

Miután ez az oldal megnyílt, a táblázatban látható az összes dolgozó, manager, és a raktárvezető is, illetve nekik a személyes adataik. *(22. ábra)*

1. Felhasználók Listázása
   1. Funkció: **Egy felhasználói táblázat megjelenítése, szűrési és műveleti lehetőségekkel.**
   2. Felület: **A felhasználók adatait tartalmazó lista, szűrési lehetőségekkel a fejlécen.**
   3. Elemei:
      1. Szűréses keresés: **lehetőség a felhasználók szűrésére különböző kritériumok szerint.**
      2. Táblázat oszlopai: **ID, Felhasználók, Jogosultság, Kapcsolattartás kezdetének dátuma, Elérhetőség és Műveletek.**
      3. **Az Elérhetőség oszlop tartalmazza a felhasználó telefonját és e-mail címét.**
      4. **A Műveletek oszlopban ikonok találhatók a felhasználói profilok megtekintésére, szerkesztésére és törlésére.**
2. Felhasználó Módosítása *(23. ábra)*
   1. Funkció: **Egy kiválasztott felhasználó adatainak szerkesztése.**
   2. Felület: **Egy felugró ablakban jelennek meg a szerkeszthető adatmezők.**
   3. Elemei:
      1. Felhasználónév: **a felhasználó egyedi azonosítója.**
      2. Teljes név: **a felhasználó teljes neve.**
      3. Jogosultság: **legördülő menü a felhasználói jogosultságok kiválasztásához.**
      4. Telefonszám és E-mail cím: **kapcsolattartási adatok.**
      5. Gombok: **Mentés a módosítások elmentéséhez, Vissza a kilépéshez.**
3. Felhasználó Törlésének Megerősítése *(24. ábra)*
   1. Funkció: **Egy felhasználó törlésének megerősítése.**
   2. Felület: **Egy megerősítő ablak, amely információt kér a felhasználótól.**
   3. Elemei:
      1. Szöveg: **"Biztos vagy benne, hogy törölni szeretnéd ezt a felhasználót?"**
      2. Megjelenített adatok: **felhasználónév, teljes név, jogosultság.**
      3. A képen szöveg, szoftver, Weblap, Webhely látható

         Automatikusan generált leírásGombok: **Igen a törlés megerősítésére, Nem a művelet megszakítására.**

**22. ábra:** Felhasználók oldala

**A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás**

**23**. ábra Felhasználók módosításának a Modal-ja

### **A képen szöveg, képernyőkép, szoftver, Operációs rendszer látható Automatikusan generált leírás**

### 

24. ábra: Felhasználó törlésének a megerősítésének a Modal-ja

### 2.2.7 Hozzáadás oldal

Ezt az oldalt megnyitva, látható több dolog is. Itt is itt van a szűréses keresés. Itt rá lehet mind dolgozóra, mind pedig rendelésre keresni. Ez alatt 2 gomb található, az egyik felhasználót, a másik rendelést ad hozzá a rendszerhez. A gombok alatt látszik a 2 táblázat, ami friss adat esetén, vagy a szűréses keresést alkalmazva frissül. *(25. ábra)*

Itt pedig látható a teljes oldal felépítése:

1. Felhasználók listázása

Funkció: **Felhasználók megjelenítése táblázatos formában.**

Felület: **Szűrési lehetőségekkel rendelkező lista, ami tartalmazza a felhasználók alapadatait.**

Elemei:

ID: **Egyedi azonosító a felhasználónak.**

Felhasználó neve: **A felhasználó azonosítója vagy neve.**

Jogosultság: **A felhasználó szerepkörét jelző kategória.**

Elérhetőség: **A felhasználó telefonszáma és e-mail címe.**

További műveletek: **Ikonok a felhasználó szerkesztéséhez vagy törléséhez.**

2. Felhasználó hozzáadása

Funkció**: Új felhasználó felvételének űrlapja.** *(26. ábra)*

Felület: **Ablak, ahol megadhatók a felhasználó alapadatai.**

Elemei:

Felhasználónév: **Azonosító, amit a rendszerben használni fog.**

Felhasználó teljes neve: **A felhasználó valódi neve.**

Jogosultság: **Lehetséges jogosultsági szintek közül lehet választani.**

Telefonszám és E-mail cím: **Kommunikációhoz szükséges adatok.**

Gombok: **Felhasználó hozzáadása a létrehozás véglegesítéséhez, X a megszakításhoz.**

Rendeléskezelés:

1. Rendelés leadása

Funkció: **Új rendelés felvétele.** *(27. ábra)*

Felület: **Egyszerű űrlap, ahol a termék adatait lehet rögzíteni.**

Elemei:

Áru neve: **A megrendelni kívánt termék neve.**

Áruszolgáltató: **A szállítást vagy eladást végző cég neve.**

Rendelni kívánt mennyiség: **Megrendelni kívánt darabszám.**

Gomb: **Rendelés leadása a folyamat befejezéséhez, X a megszakításhoz.**

# **A képen szöveg, képernyőkép, Betűtípus, szám látható Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver, Betűtípus látható Automatikusan generált leírásA képen szöveg, szoftver, Weblap, Webhely látható Automatikusan generált leírás2.3** A webapplikáció kód alapú működése

27. ábra: Rendelés hozzáadás Modal

26. ábra: Felhasználó hozzáadása Modal

25. ábra: Hozzáadás oldal

A screen shot of a computer program

Description automatically generatedEz az ábra egy felhasználói profil szerkesztő űrlapot mutat be. Itt a felhasználó különböző adatait (pl. felhasználónév, teljes név) lehet szerkeszteni. Az @if direktíva használatával bizonyos részek csak akkor jelennek meg, ha a felhasználó megfelelő jogosultsággal rendelkezik, *például: "Raktárvezető" (28. ábra).*

*28. ábra: Felhasználó szerkesztésének az űrlapja (Modal)*

A screen shot of a computer program

Description automatically generatedAz alábbi ábra a felhasználói felület egy részét mutatja, ahol üzenetek jelennek meg. Ezek az üzenetek "toasts" néven ismertek, ami egy rövid, automatikusan eltűnő üzenetet jelent, amely információt nyújt a felhasználó számára. Itt látható a sikeres műveletek (zöld hátterű) és a hibák (piros hátterű) megjelenítésének a kódja *(29. ábra).*

*29. ábra: Toast-ok amikor módosítás történik*

A jelen lévő ábra egy olyan funkciót mutat, ahol egy felhasználó jelszavának visszaállításához szükséges token generálódik. Amennyiben a felhasználó létezik az adatbázisban, létrehoz egy token-t a jelszó visszaállításához és elküld egy e-mailt a felhasználónak a visszaállításhoz szükséges linkkel. Sikeres művelet esetén egy pozitív visszajelzést küld, ellenkező esetben egy hibaüzenetet ad vissza *(30. ábra).*

A computer screen shot of text

Description automatically generated

*30. ábra: Jelszó helyreállítása E-mail*

A negyedik ábra egy regisztrációs funkció kódját mutatja, ahol a felhasználó által megadott adatok validálása történik, majd a felhasználó létrehozása az adatbázisban. Ezután az újonnan létrehozott felhasználó eseménye generálódik (ez lehet pl. egy értesítés vagy statisztika frissítés), és a felhasználó automatikusan bejelentkezik *(31. ábra).*

A képen szöveg, képernyőkép látható

Automatikusan generált leírás

*31. ábra: Regisztrációs rész*

# Hibák és azok javítása

A weboldalon meglévő felhasználó hozzáadási rendszerrel volt egy probléma, melyet a fejlesztések végén nem vettünk észre, majd később lettünk rá figyelmesek. A probléma a jelszó lekezelésével kapcsolatban jelent meg amikor a raktárvezető vagy manager hozzáad egy felhasználót a rendszerhez. A rendszer elsőnek elfogadta semmi probléma nélkül a felhasználókat és létrehozta a fiókot jelszó nélkül. Később, egy SQL hibára lettünk figyelmesek mely jelezte, hogy a jelszó hozzáadása szükséges a felhasználóknak.

## Javítás leírása

A javítás a *“*UserController*”* nevű fájl *“*store*”* funkciójában történt. Elsőnek generálunk egy 20 karakteres jelszót a *“*Str::random(20)*”* függvény segítségével. Majd ezt *“*Hash::make($password)*”* függvény segítségével titkosítjuk majd adatbázisba elmentjük. Ezután a jelszót elküldjük a regisztrált személynek e-mailben a felhasználónévvel együtt.

public function store(Request $request)

{

$request->validate([

'name' => ['required', 'string', 'max:255'],

'username' => ['required', 'string', 'max:255', 'unique:'.User::class],

'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:'.User::class],

'pnumber' => ['required', 'regex:/^(\+\d{1,3}[- ]?)?\d{10}$/'],

]);

$password = Str::random(20);

User::create([

'name' => $request->name,

'username' => $request->username,

'email' => $request->email,

'pnumber' => $request->pnumber,

'password' => Hash::make($password),

]);

Mail::raw("A felhasználóneved: {$request->username}; A jelszavad: {$password}; Bejelentkezés után erősen ajánlott a jelszó módása biztonsági okokból.", function ($message) use ($request) {

$message->to($request->email)

->subject('Hozzá lettél adva a rendszerünkhöz!');

});

return redirect(route('add'));

}

# Összefoglalás

## A játék megvalósítása és személyes tapasztalatok

A játék megvalósítása a következőképpen zajlott. Eleinte nem teljesen tudtam, hogy mit is csinálok éppen a játék készítésekor, viszont akármikor elakadtam, a Unity hatalmas felhasználói körének hála, bármilyen problémám volt a fórumokon és oktatóvideókon segítségével mindig ki tudtam találni valamilyen működő megoldást a problémámra. Elsőként azzal kellett főként megismerkednem, hogy a játékmotor önmagában hogyan épül fel, az *“Editor”* ablakban egyes gombok milyen funkciókat látnak el és hogy objektum orientált alapú.

Ezt követően a munkamenet a következő volt:

* Először, létre kellett hozni egy játék béli Object-et. Ezekből épül fel az egész játék hiszen, hozzájuk lehet csatolni a szkripteket, melyek segítségével érhetjük el, hogy az adott tárgy képes legyen valamit végrehajtani, legyen szó mozgásról, számolásról, felhasználói felületen lévő gombról vagy akármiről, minden abból épül fel, hogy léteznek olyan Object-ek a játékban, melyek rendelkeznek valami szkripttel.
* Példaként az első játékelem, amit létrehoztam az egy kapszula volt, melyet szkript segítségével értem el, hogy mozgatható legyen, ez lett maga a játékos.

Amint készen voltak, a játék alap mechanikái, utána belekezdtem a játék kinézetének tervezésébe, mert egy jó játéknak szüksége van személyiségre.

Ennek a megvalósítására a már fent említett Blender nevű háromdimenziós grafikai és animátor programot használtam. Ez a felület is új volt számomra, de az oktatóvideók itt is könnyen elérhetőek, érthetőek és követhetőek voltak.

A Blenderen belül a munkafolyamat a következő képpen zajlott:

* Első lépésként kitöröltem a Blender által létrehozott alapvető elemeket majd létrehoztam a sajátom.
* Ezt követően mivel Voxel alapúak a modellek (kis négyzetekből felépülő modellek) be kellett állítanom a Blender-en belül használt mértékegységet egy bizonyos méretre, hogy kockára pontos modelleket tudjak létrehozni.
* Végül pedig ki kellett az adott modell felszínét csomagolni és a rá tartozó mintát megrajzolni, majd, ha azzal végeztem exportáltam a fájlt egy Unity által használt FBX formátumba, és egyszerűen behúztam a játék mappájába.
  1. A weboldal és a mobil alkalmazás megvalósítása, és személyes tapasztalatok

A projektünk egy integrált raktárkezelő rendszer létrehozását célozta, amely egy weboldal és egy mobil alkalmazás formájában került megvalósításra. A rendszer lehetővé teszi a raktárkészlet nyomon követését, a csomagok kiszállítási státuszának kezelését és az alkalmazottak adatainak adminisztrációját egyaránt.

**Technológiai részek:**

- **Frontend:** A felhasználói felület HTML5, CSS3 és JavaScript technológiák segítségével készült, melyek biztosítják az oldalak gyors betöltését és a különböző eszközökön történő megfelelő megjelenést. HTML5-t használtunk a dokumentumok strukturálására, CSS3-at a stílusok definiálására, míg a JavaScript biztosította az oldalak interaktivitását.

- **Backend:** A szerveroldali logika PHP nyelven íródott, Laravel keretrendszerrel kiegészítve. Laravel az MVC architektúrát követi, amely segít a kód karbantarthatóságában és a fejlesztési folyamatok egyszerűsítésében.

- **Fejlesztői környezet:** Az IntelliJ IDEA Ultimate IDE-t használtuk, ami támogatást nyújtott a PHP és több más nyelv fejlesztéséhez, valamint integrált eszközöket kínált a kód analízisére és hibakeresésre.

**Felhasználói dokumentáció és felületek:**

- **Bejelentkezés/Regisztráció/Elfelejtett jelszó:** Az adminisztrációs felületre történő belépéshez szükséges azonosítási folyamatok, beleértve a jelszó helyreállítási lehetőséget is, amely email-ben küldött linken keresztül érhető el.

- **Főoldal:** A főoldalra belépve azonnal áttekintést nyújt a raktárrendszer állapotáról, lehetőséget biztosítva a készlet, a csomagok állapotának szűrésére és frissítésére.

- **Navigációs sávok:** Mind a bal, mind a jobb oldali navigációs sávok segítségével gyorsan elérhetővé válnak a különböző funkciók, mint például a rendelések, felhasználók kezelése vagy új elemek hozzáadása.

**Személyes tapasztalatok:**

- A projekt során mélyebb ismereteket szereztünk a modern webfejlesztési technológiákról és az MVC architektúra alkalmazásáról. A Laravel keretrendszer használata különösen hasznos volt, hiszen számos előre megírt funkciót biztosít, amelyekkel gyorsabban és hatékonyabban tudtuk implementálni a szükséges funkcionalitásokat.

- Az IntelliJ IDEA Ultimate használata jelentősen megkönnyítette a fejlesztési folyamatokat, különösen a hibakeresés és a kód refaktorizálása során.

Összefoglalva, a projekt sikeresen biztosítja a vállalati raktárrendszer hatékony kezelését, javítva a napi munkafolyamatokat és támogatva a vállalat operációs tevékenységeit. Az alkalmazott technológiák és fejlesztési módszertanok lehetővé tették, hogy egy biztonságos, megbízható és skálázható rendszert hozzunk létre.