TESZTELÉSI DOKUMENTÁCIÓ

Készítette:

Szigili Edit Mária

Konzulens:

Farkas Zoltán

Miskolc

2023.

TESZTELÉSI DOKUMENTÁCIÓ

A játékprogram tesztelése különösen fontos, mivel a játékok általában elég összetettek és a felhasználói élmény nagyon fontos. Jelen dokumentumban rögzítésre kerültek az alkalmazás működésénél alkalmazott tesztelési folyamatok. Ezeket röviden összefoglalva a teszttervben felsoroljuk.

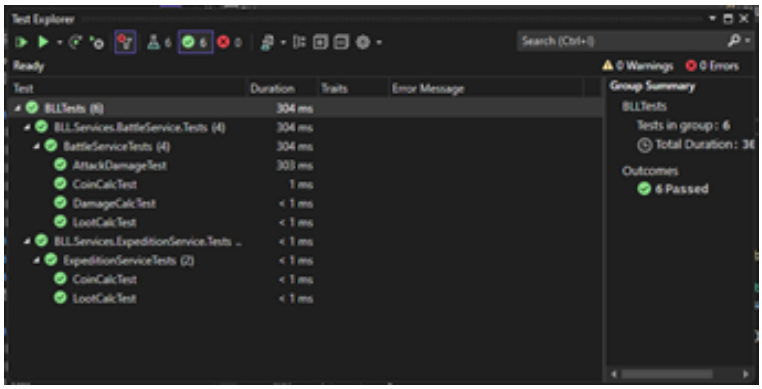
Teszterv:

- NUnit tesztek, és eredmények a Visual Studio Test Explorer felületén a backend alkalmazás tesztelésére.
- Lighthouse chrome bővítmény a FrontEnd teljesítményének elemzésére.
- Manuális tesztek az alkalmazás működése közben.

NUnit tesztek a Visual Studio Test Explorer felületén:

6 Teszt került kiértékelésre, mindegyik sikeresen lefutott. Ezek bemutatása a következők:

Test Explorer:



mockRandomProvider:

```
[TestClass()]
public class BattleServiceTests
{
    // Random mock
    private readonly Mock<IRandomProvider> mockRandomProvider = new();
```

A battle service és az expedition service tesztelt metódusai a c# random osztályára támaszkodnak. Véletlenszerűen generált adatokat nem lehet tesztelni, mert hamis eredményt adhat a teszteset. Emiatt meg kellett változtatnunk a random osztály működését, hogy az általunk specifikált adatot adja a tesztek alatt vissza. Így tudjuk biztosítani, hogy a tesztek eredménye mindig konstans lesz, és megbízhatóak lesznek.

A Battle tesztjei:

AttacDamageTest:

```
[TestMethod()]
public void AttackDamageTest()
{
    //Arrange
    int strength = 13;
    int agility = 12;
    int churchLevel = 2;
    int practiceRangeLevel = 1;

    // Az általad megadott számot fogja visszaadni mindig randomként
    // Random mock
    mockRandomProvider.Setup(randomProvider =>
        randomProvider.GetRandomNumber(It.IsAny<int>(), It.IsAny<int>())).Returns(10);

    //Act
    BattleService battleService = new(null, null, null, null, mockRandomProvider.Object);
    int result = battleService.AttackDamage(strength, agility, churchLevel, practiceRangeLevel);


    //Assert
    Assert.AreEqual(18, result);
}
```

Beállítjuk a strength, agility, churchLevel és practiceRangeLevel értékét. Beállítjuk a véletlenszerű szám generáló Mock-ot, hogy mindig a 10-es számot adja vissza. Létrehozunk egy battleService példányt, amelynek az első négy paraméterének null értéket adunk át, és a Mock-olt számot adjuk át az ötödik paraméterként. Majd meghívjuk az AttackDamage metódust a battleService példányon az előbbi bemenő adatokkal. Végül ellenőrizzük, hogy az eredmény megfelelő-e, azaz ebben az esetben 18.

DamageCalcTest:

```
[TestMethod()]
public void DamageCalcTest()
{
    //Arrange
    double multiply = 1.5;
    int strength = 13;
    int churchLevel= 2;

    // Az általad megadott számot fogja visszaadni mindig randomként
    // Random mock
    mockRandomProvider.Setup(randomProvider =>
        randomProvider.GetRandomNumber(It.IsAny<int>(), It.IsAny<int>())).Returns(10);

    //Act
    BattleService battleService = new BattleService(null, null, null, null, mockRandomProvider.Object);
    int result = battleService.DamageCalc(multiply, strength, churchLevel);

    //Assert
    Assert.AreEqual(18, result);
}
```

Beállítjuk a multiply, strength, és churchLevel értékét. Beállítjuk a véletlenszerű szám generáló Mock-ot, hogy mindig a 10-es számot adja vissza. Létrehozunk egy battleService példányt, amelynek az első négy paraméterének null értéket adunk át, és a Mock- olt számot adjuk át az ötödik paraméterként. Majd meghívjuk az DamageCalc metódust a battleService példányon az előbbi bemenő adatokkal. Végül ellenőrizzük, hogy az eredmény megfelelő-e, azaz ebben az esetben is 18.

LootCalcTest:

```
[TestMethod()]
public void LootCalcTest()
{
    //Arrange
    int intellect = 13;

    // Az általad megadott számot fogja visszaadni mindig randomként
    // Random mock
    mockRandomProvider.Setup(randomProvider =>
        randomProvider.GetRandomNumber(It.IsAny<int>(), It.IsAny<int>())).Returns(80);

    //Act
    BattleService battleService = new BattleService(null, null, null, null, mockRandomProvider.Object);
    int result = battleService.LootCalc(intellect);

    //Assert
    Assert.AreEqual(145,result);
}
```

Beállítjuk az intellect értékét. Beállítjuk a véletlenszerű szám generáló Mock-ot, hogy mindig a 80-as számot adja vissza. Létrehozunk egy battleService példányt, amelynek az első négy paraméterének null értéket adunk át, és a Mock- olt számot adjuk át az ötödik paraméterként. Majd meghívjuk az LootCalc metódust a battleService példányon az előbbi bemenő adatokkal. Végül ellenőrizzük, hogy az eredmény megfelelő-e, azaz ebben az esetben 145.

CoinCalcTest:

```
[TestMethod()]
public void CoinCalcTest()
{
    //Arrange
    int intellect = 13;

    // Az általad megadott számot fogja visszaadni mindig randomként
    // Random mock
    mockRandomProvider.Setup(randomProvider =>
        randomProvider.GetRandomNumber(It.IsAny<int>(), It.IsAny<int>())).Returns(150);

    //Act
    BattleService battleService = new BattleService(null, null, null, null, mockRandomProvider.Object);
    int result = battleService.CoinCalc(intellect);

    //Assert
    Assert.AreEqual(215, result);
}
```

Beállítjuk az intellect értékét. Beállítjuk a véletlenszerű szám generáló Mock-ot, hogy mindig a 150-es számot adja vissza. Létrehozunk egy battleService példányt, amelynek az első négy paraméterének null értéket adunk át, és a Mock- olt számot adjuk át az ötödik paraméterként. Majd meghívjuk az CoinCalc metódust a battleService példányon az előbbi bemenő adatokkal. Végül ellenőrizzük, hogy az eredmény megfelelő-e, azaz ebben az esetben 215.

Expedíció tesztjei:

ExpeditionServiceTest mockRandomProvider:

Az expedition service tesztelt metódusainál is ugyanazt a mockRandomProvidert használjuk, amelynek szükségessége és működése már a battle service-nél előzetesen bemutatásra került.

LootCalcTest:

```
[TestMethod()]
public void LootCalcTest()
{
    //Arrange
    int intellect = 13;
    int difficulty = 2;

    // Az általad megadott számot fogja visszaadni mindig randomként
    // Random mock
    mockRandomProvider.Setup(randomProvider =>
        randomProvider.GetRandomNumber(It.IsAny<int>(), It.IsAny<int>())).Returns(80);

    //Act
    ExpeditionService expeditionService = new ExpeditionService(null, null, mockRandomProvider.Object);
    int result = expeditionService.LootCalc(intellect, difficulty);

    //Assert
    Assert.AreEqual(145, result);
}
```

Beállítjuk az intellect és a difficulty értékét. Beállítjuk a véletlenszerű szám generáló Mock-ot, hogy az mindig a 80-as számot adja vissza. Létrehozunk egy expeditionService példányt, amelynek a Mock adatát adjuk át harmadik paraméterként. Majd meghívjuk a LootCalc metódust az expeditionService példányon az előbbi bemenő adatokkal. Végül ellenőrizzük, hogy az eredmény megfelelő-e, azaz ebben az esetben 145.

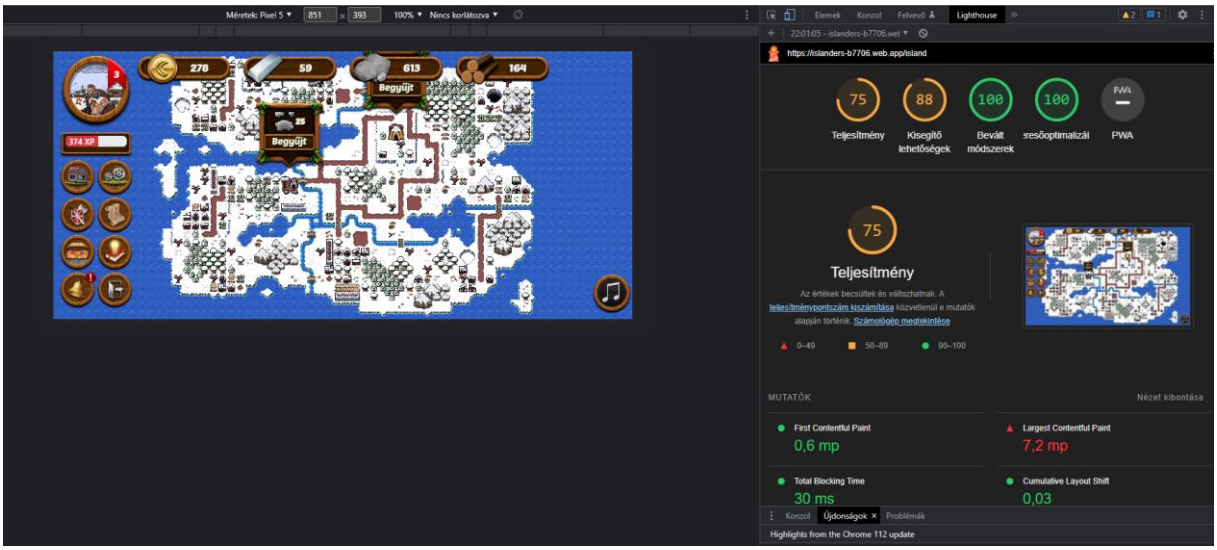
CoinCalcTest:

```
[TestMethod()]
public void CoinCalcTest()
{
    //Arrange
    int intellect = 13;
    int difficulty = 2;

    // Az általad megadott számot fogja visszaadni mindig randomként
    // Random mock
    mockRandomProvider.Setup(randomProvider =>
        randomProvider.GetRandomNumber(It.IsAny<int>(), It.IsAny<int>())).Returns(150);

    //Act
    ExpeditionService expeditionService = new ExpeditionService(null, null, mockRandomProvider.Object);
    int result = expeditionService.CoinCalc(intellect, difficulty);

    //Assert
    Assert.AreEqual(215, result);
}
```

Beállítjuk az intellect és a difficulty értékét. Beállítjuk a véletlenszerű szám generáló Mock-ot, hogy az mindig a 150-es számot adja vissza. Létrehozunk egy expeditionService példányt, amelynek a Mock adatát adtuk át harmadik paraméterként. Majd meghívjuk a CoinCalc metódust az expeditionService példányon az előbbi bemenő adatokkal. Végül ellenőrizzük, hogy az eredmény megfelelő-e, azaz ebben az esetben 215.

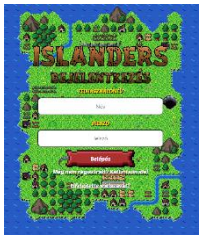
Lighthouse tesztelés

Lighthouse chrome bővítménnyel végig teszteltük a FrontEnd teljesítményét:



A Lighthouse egy nyílt forráskódú, automatizált eszköz, amely segít az éles weboldalak teljesítményének és minőségének tesztelésében, javításában. A Lighthouse a Google által fejlesztett, és ingyenesen elérhető, nyílt forráskódú eszköz, amely a böngészőben futtatva méri a weboldalak teljesítményét és használhatóságát, legyen az nyilvános vagy hitelesítést igénylő weboldal.
A Lighthouse tesztek számos aspektusát tesztelik, például az oldal betöltési sebességét, a teljesítményt, a hozzáférhetőséget, a használhatóságot, az SEO-t, a kisegítő lehetőségeket, a progresszív webalkalmazásokat és az általános biztonságot. Az eszköz a tesztek eredményeit jelentéseken keresztül mutatja be, amelyek összefoglalják az egyes tesztek eredményét, és javaslatokat tesznek a weboldal optimalizálására.

Manuális tesztek az alkalmazás működése közben Google Chrome webböngészőben.



1. Teszt: A játék megnyitása a localhost-on a http://localhost:3000.
   Várt eredmény: A felület megfelelően megjelenik, a belépési oldallal.
   A teszt sikeres.

2. Teszt: A felület regisztrációs részének meghívása a belépési oldalról.
   Várt eredmény: A regisztációs oldalra navigál a felület.
   A teszt sikeres.

3. Teszt: Regisztráció 1. Regisztráció az elvárt adatokkal.
   Várt eredmény: Az adatok beírása után a regisztráció megtörténik, ezt egy felugró ablak jelzi, a folytatáshoz szükséges információkkal.
   A teszt sikeres.

4. Teszt: Regisztráció 2.: Regisztrációnál hibásan visszük be az adatokat: A felhasználónévben ékezetet és a jelszóba 7 karaktert írunk.
   Várt eredmény: Az adatok beírása után a regisztráció nem történik meg. A Hibás adatbevitelről a felületen megjelenik a hiba kiírása mindkét hiba helyénél, és mindaddig nem tűnik el ameddig az elvárt paramétereknek megfelelő adat nem kerül a helyére.
   A teszt sikeres.

5. Teszt: Regisztráció 3. Regisztrációnál hibásan visszük be az adatokat: Már regisztrált emailcímmel próbálunk újra regisztrálni
   Várt eredmény: Az adatok beírása után a regisztráció nem történik meg, mivel már létező email címmel próbálunk új játékot kezdeni. ezt egy felugró ablak jelzi, a folytatáshoz szükséges információkkal.
   A teszt sikeres.

6. Teszt: Első belépés a játékba.: Az Email visszaigazolás után belépés a játékba.
   Várt eredmény: A megfelelő adatok beírás után belép a felületre, és egyből a szigetválasztási lehetőségre irányít át az oldal.
   A teszt sikeres.

7. Teszt: A sziget kiválasztása.: A 4 sziget közül kiválasztjuk a kívánt szigetet, az aktívvá válik, majd a sziget mentése gombbal elmentjük.
   Várt eredmény: A sziget kiválasztása után belép a játékba, a szigethez tartozó profilkép megjelenik a Saját Profil helyén, az alapanyagok megkapják a kezdeti értékeiket, a szigeten a mozgó emberkék elindulnak a sziget útjain, az értesítésbe belekerül a szigetválasztás értesítője, ezt a felkiáltójel jelzi az értesítés ikonján.
   A teszt sikeres.

8. Teszt: Saját profil megtekintése.
   Várt eredmény: A saját profilban a felhasználónév, email cím, és a jelszó módosítás lehetősége jelenik meg a bevitt adatoknak megfelelően.
   A teszt sikeres.

9. Teszt: Sziget menedzselés 1.: Képességpontok kiosztásának tesztelése.: Elérhető képességpontok szétosztása. Visszavonása, újra szétosztása, mentése.
   Várt eredmény: Az elérhető képességpontok rendben átíródnak a képességekhez, és vissza. A mentés után a két gomb inaktívvá válik.
   A teszt sikeres.

10. Teszt: Sziget menedzselés 2.: Épületek.: Információ.
    Várt eredmény: Minden épületnél megjelenik a „?” ikonra kattintva a megépítéséhez szükséges alapanyagszükséglet.
    A teszt sikeres.

11. Teszt: Expedíció 1/1. Expedíció indítása Könnyű fokozaton.
    Várt eredmény: Az expedíció elindul, az eredmény felugró ablakban jelenik meg a képernyőn az expedíción történt események felvázolásával, erről kiküldi az értesítést. A kapott alapanyagok feltöltődnek a főoldalon látható készletbe. Elindul az 1 perces visszaszámlálás.
    A teszt hibára futott.:
    Hiba részletes leírása: A kapott alapanyagok nem töltődtek fel közvetlenül lefutáskor a főoldalon látható alapanyagok közé. Az expedícióból kilépve sem frissül a készletek mennysége, csak az oldal manuális frissítésével kerültek be a helyes értékek a helyükre.
    A hiba javításra került.
    Újratesztelés: Expedíció 1/2. Expedíció indítása Könnyű fokozaton.
    Várt eredmény: A várt eredmény az „Expedíció indítása Könnyű fokozaton.” első tesztelésénél rögzítve.
    A javítás után a teszt sikeresen lefutott, a hiba megszűnt.

12. Teszt: Expedíció 2. Expedíció indítása Normál fokozaton.
    Várt eredmény: Az expedíció elindul, az eredmény felugró ablakban jelenik meg a képernyőn az expedíción történt események felvázolásával, erről kiküldi az értesítést. A kapott alapanyagok feltöltődnek a főoldalon látható készletbe. Elindul az 1 perces visszaszámlálás.
    A teszt sikeres.

13. Teszt: Expedíció 3. Expedíció indítása Nehéz fokozaton.
    Várt eredmény: Az expedíció elindul, az eredmény felugró ablakban jelenik meg a képernyőn az expedíción történt események felvázolásával, erről kiküldi az értesítést. A kapott alapanyagok feltöltődnek a főoldalon látható készletbe. Elindul az 1 perces visszaszámlálás.
    A teszt sikeres.

14. Teszt: Épület lehelyezése.: A kiválasztott épület lehelyezése a térképen.
    Várt eredmény: Az épület építkezési ablaka a kiválasztott helyen megjelenik, 2 perctől visszaszámlál, majd megjelenik a kész épület, és elkezdi a termelést.
    A teszt sikeres.

15. Teszt: Csata 1/1: A csata oldalon felkínált ellenfelek közül a megfelelő kiválasztása.
    Várt eredmény: A lezajlott csata után a képernyőn megjelenik a csatában történt események felvázolása. Csata végeredménye is kiírásra kerül a szerzett alapanyagokkal, melyek feltöltődnek a főoldalon látható készletbe. Mindkét fél kap értesítést a csata kimeneteléről. A visszaszámlálás 10 percről elindul.
    A teszt hibára futott.:
    Hiba részletes leírása: Az ellenfél szintje maximum kettővel lehet magasabb, mint a játékosé.

    A hiba javításra került.
    Újratesztelés: Csata 1/2.: A csata oldalon felkínált ellenfelek közül a megfelelő kiválasztása.
    Várt eredmény: A várt eredmény a „Csata” első tesztelésénél rögzítve.
    A javítás után a teszt sikeresen lefutott, a hiba megszűnt.

16. Teszt: Épület fejlesztés
    Várt eredmény: A fejlesztés elindításakor az épület elkészüléséig a visszaszámláló ablak jelenik meg az épület helyén.
    A teszt sikeres.

17. Teszt: Piac 1.: Hirdetés feladása A csere termékek megadása, a hirdetés feladása.
    Várt eredmény: Az elcserélendő alapanyagok kiválasztása, mennyiségek megadásával. A hirdetés gombra kattintva a cserére felajánlott termék levonódik a készletből.
    A teszt sikeres.

18. Teszt: Piac 2/1.: Hirdetés törlése.
    Várt eredmény: A saját hirdetéseken belül az egyik hirdetés törlése. Az előzetesen levont alapanyagok jóváírása a készletben.
    A teszt hibára futott.:
    Hiba részletes leírása: A hirdetés törlésre kerül a saját hirdetések listából, de az alapanyagok nem kerülnek vissza a készletbe.
    A hiba javításra került.
    Újratesztelés: Piac 2/2.: Hirdetés törlése.
    Várt eredmény: A várt eredmény a „Hirdetés törlése” első tesztelésénél rögzítve.
    A javítás után a teszt sikeresen lefutott, a hiba megszűnt.

19. Teszt: Piac 3.: Csere.
    Várt eredmény: A csere gomb megnyomásával az elcserélt alapanyagok a főoldalon látható készletben rendezésre kerülnek. A kapott alapanyag bekerül a készletbe, a cserébe adott tételek levonásra kerülnek. A hirdetés feladója értesítést kap a csere megtörténtéről, és megkapja a cseretermékét.
    A teszt sikeres.