**Záródolgozat**

Tanuló neve: Tóth Dániel Mátyás

Osztálya: 5/13.s

Elkészítési éve: 2022/23

**Záródolgozat adatlap**

**Záródolgozat készítőjének neve:**

Tóth Dániel Mátyás

**Elérhetősége(i):**

[dani.toth2003@gmail.com](mailto:dani.toth2003@gmail.com)

[furryvagyokihhi@gmail.com](mailto:furryvagyokihhi@gmail.com)

+36-30-603-1890

**A záródolgozat témája:**

Egy C#-ban fejlesztett program, aminek a feladata hogy tároljon a Minecraft nevű játékból tárgyakat és az elkészítési módjukat, és annak az adatait kezelje.

**Záródolgozat címe:**

MCraftingTree

**Konzulens tanár:**

Szegedi Barnabás

Kelt: Budapest, 2023.

.................................................................. .................................................................

a záródolgozat készítőjének aláírása a konzulens tanár aláírása

EREDETISÉG NYILATKOZAT

Alulírott tanuló kijelentem, hogy a záródolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült záródolgozatomban található eredményeket az intézmény egy példányban archiválhatja.

Budapest, 20

tanuló aláírása

**Záródolgozati konzultáció igazoló lap**

Alulírott Szegedi Barnabás konzulens tanár aláírásommal igazolom.

Tóth Dániel Mátyás nevű tanuló konzultációkon való részvételével.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dátum** | **Téma** | **Tanuló aláírása** | **Tanár aláírása** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tartalom

[1. Bevezetés 6](#_Toc130895106)

[2. Feladatspecifikáció 8](#_Toc130895107)

[3. Fejlesztői dokumentáció 9](#_Toc130895108)

[3.1. Általános ismertetés 9](#_Toc130895109)

[3.2. Fejlesztői környezet 12](#_Toc130895110)

[3.3. Visual Studio 2022 Community Edition 13](#_Toc130895111)

[3.4. MonoDevelop 14](#_Toc130895112)

[4. Felhasználói dokumentáció 15](#_Toc130895113)

[5. Irodalomjegyzék 15](#_Toc130895114)

[6. Mellékletek 15](#_Toc130895115)

[6.1. Ábrajegyzék 15](#_Toc130895116)

# Bevezetés

A programom készítésétéhez az inspiráció akkor jött, mikor óra közben játszottam a Minecraft nevű játékkal, és egy barátomnak teszteltem egy modifikált verzióját. Sokkal több tárgy, ellenség és barkácsolási recept volt benne, mint az alap játékban, ezért rengeteg mindent nem tudtam, hogy hogy kell megcsinálni, és hogy milyen tárgyakba kerülne. Annyit tudtam, hogy melyik tárgyat kell megszereznem ahhoz, hogy „győzzek”, amihez egy nagyon komplex recept tartozott. Ekkor volt egy gondolatom, hogy: „Hmm, de jó lenne hogyha lenne egy program vagy egy modifikáció ahhoz, hogy megmutassa nekem a szükséges alapanyagokat ehhez a tárgyhoz…”, és úgy döntöttem, hogy megcsinálom.

Először is, ki kellett találnom, hogy hogyan és mivel hozom létre, és mi kell ahhoz, hogy úgy működjön, ahogy azt én akarom. Ideálisan Java-ban lett volna jó megírni, hiszen a játék is Java 8, majd Java 17-ben íródott, és könnyen tudtam volna importálni az osztályfájlokat magából a .jar fájlból, de helyette a C# mellett döntöttem, mert ahhoz valamilyen szinten értek is, Java-val ellentétben. Megterveztem az adatbázis alapjait draw.io-ban, hogy könnyen tudjak változtatni az oszlopokon, és hogy ne kelljen szenvednem azzal, hogy valaminek a tárolására nem jó. A grafikai felülethez WPF-et használtam, mert gyorsabb, modernebb és több funkcionalitást nyújt a WinForms-hoz képest. Emellé még Material Design-t is használtam, ami a Bootstrap-hez hasonlóképpen egy felhasználói felület keretrendszer, hogy jobban nézzen ki a program, mint ahogy én azt meg tudnám írni nélküle. Kitaláltam, hogy akarom hogy a program kinézzen, milyen funkciói legyenek, és nekikezdtem megírni a programot, az adatbázissal kezdve. Entity Framework 6-ot használva megcsináltam a terveim alapján, és nagyon alap funkciókkal teszteltem, hogy be tud-e olvasni és lehet-e rá írni. Ezután megírtam, hogy lehessen hozzáadni tárgyakat az adatbázishoz, hogy dobja-e a megadott tárgyat egy ellenség, ha igen melyik és mekkora eséllyel, a tárgyakkal asszociált típus, tárgyakból készített receptek, legyen az kemencében kisütés, alkímiában készült főzetek, vagy csak egyszerűen barkácsolások. Ezután kidolgoztam, hogy megjelenjenek a tárgyak egy listában, a velük asszociált képekkel együtt, megcsináltam, hogy be lehessen tölteni recepteket azzal, hogy behúzol egy képet a barkácsasztal menüjébe a program közepén, és hogy kiszámolja hogy milyen tárgyak kellenek a megadott receptnek az elkészítéséhez, és mennyi belőlük. Végül hozzáadtam alap adatokat az adatbázishoz, tárgyak, recepteket és típusok amik megjelennek az alap játékban is, modifikációk nélkül.

Végül csak a tesztelés és ellenőrzés maradt, amik sikeresek voltak.

# Feladatspecifikáció

A szoftver lényege a Minecraft játékból tárgyak és receptek felvétele és kezelése, hogy tudjuk hogy a játékban a mennyibe kerülnek megadott tárgyak. Ezt egy egyszerűbb felülettel igyekeztem megoldani.

Funkciói:

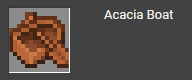
* Tárgyak kezelése és felvétele, típusokkal és dobási eséllyel
* Receptek felvétele
* Alap tárgyak megkeresése egy megadott tárgy receptjéhez

# Fejlesztői dokumentáció

## Általános ismertetés

Először ismerkedjünk meg az alap fogalmakkal:

* Modifikáció: Játékosok által készített változtatások a játékban, például extra receptek és/vagy tárgyak, legtöbbszőr CurseForge-ról vagy Modrinth-ról szerezhetőek be, de saját magunk is tudunk csinálni, ha ismerjük a Java-t.
* Tárgy: egy Minecraft-ból származó elem, amivel tudjuk befolyásolni a világot azzal, hogy letesszük a világba, felveszünk velünk valamit a világból, vagy csinálunk belőlük másik tárgyakat amik képesek az ezelőtt felsoroltakra.



. ábra

* Recept: Egy megadott felhasználói felületen milyen tárgyakat kell felhasználni ahhoz, hogy más tárgyakat lehessen létrehozni belőlük. Ezeknek van 3 fajtája:
  + Barkácsoló asztal: egy háromszor hármas felületen lehet azonnal más tárgyakat létrehozni recepttől függően, vagy egy előre megadott alakzatban, vagy alakzat nélkül. Van egy megadott mennyisége hogy mennyi tárgyat kapsz egy megadott receptből.

A képen diagram látható

Automatikusan generált leírás

. ábra

* + Kemence: Egy az egyhez átalakítás egy tárgyból a másikba, nem azonnali, de viszont ezt nem reprezentálom a programban. Használható ha valamit ki kell sütni/beolvasztani.

A képen szöveg, clipart, névjegykártya látható

Automatikusan generált leírás

. ábra

* + A képen szöveg, csatlakozó látható

    Automatikusan generált leírásAlkímia asztal: Egy alkímiai hozzávalót belerakunk egytől háromig terjedő vízzel/főzettel teli üvegbe, és belefőzzük. Leginkább főzetek készítéséhez használatos.

. ábra

Ezt az alkalmazást túlnyomó többségben a Minecraft játékosai fogják  
használni, azon belül azok akik egyedi és/vagy másik játékosok által terjesztett modifikációkat használnak a játékhoz, és leginkább ahhoz fogják használni, hogy megnézzék milyen nyersanyagokba, azaz annál jobban le nem bonthatóbb tárgyakba, kerül egy megadott tárgy a hozzá asszociált receptet használva. Ezért észben kell tartani hogy mi a legfontosabb miközben változtatásokat hozunk létre a programhoz: a tárgyak és a receptek vannak az előnyben, minden más extra; Ha valami blokkolja a tárgyak és/vagy receptek rendes működését, azt vagy törölni vagy változtatni kell kiadás előtt.

## Fejlesztői környezet

Az általam használt fejlesztői környezet a Visual Studio 2022 Community Edition. Egy IDE *(Integrált Fejlesztői Környezet)* a C#-hoz és a .NET Framework-ös fejlesztéshez. Ideális a beépített funkciói, például az IntelliCode, az integrált Git hozzáférés és a NuGet bővítmények beszerzése miatt, de viszont csak Windows-on vagy Linux-on Wine-ban használható, de az utóbbin gyengén működik.

Ha egy Linux disztribúción szeretné fejleszteni, akkor a MonoDevelop-ot tudom ajánlani, de tesztelni a programot Wine-ban kell, mert a WPF nem támogatott Linux-on.

A használt adatbázis az SQL alapú LocalDB, amit Entity Framework 6-al kezelünk.

## Visual Studio 2022 Community Edition

A Visual Studio Community Edition az egy Microsoft által fejlesztett ingyenes IDE diákok, nyílt forráskódú és egyéni fejlesztők számára. Alapból nem képes adatokat tárolni, de be lehet szerezni hozzá csomagokat, amik képesek rá. Több szolgáltatást is nyújt, ami miatt előnyös a használatra:

* NuGet: Egy csomagkezelő, amit elsősorban olyan szoftverek csomagolására és megosztására van, amik .NET Framework-el lettek megírva.
* IntelliCode: Egy mesterséges intelligencia, amely képes kiegészíteni, átalakítani és befejezni kódjaidat egy sorban. Jelentősen meggyorsítja a kódírást.
* Git: Egy elosztott verziókezelő szoftver, amit arra használunk hogy több ember is tudja fejleszteni a programot több gépen, és hogy bárhonnan el lehessen érni a forráskódot. Ez a funkció bele van építve a Visual Studio- -ba, hogy gyorsan és könnyen fel lehessen nyomni a legutóbbi változtatásokat a Git repository-ba.

## MonoDevelop

A MonoDevelop az egy nyílt forráskódú IDE amit lehet használni C#-hoz. Hasonló módon a Visual Studio-hoz, nem tud kezelni magától adatbázist. Csak akkor javaslom használni ezt, ha a Visual Studio nem működik, vagy Linux-on akarja fejleszteni a programot.

Előnyei:

* Gyorsabb és kevesebb tárhelyet fogyaszt a Visual Studio-hoz képest.
* Képes kódokat kiegészíteni automatikusan.

Hátrányai:

* Kevésbé stabilabb, mint a Visual Studio.
* Nincsenek bővítmények a fejlesztő számára (Ez alatt nem NuGet csomagokat kell érteni).

## Entity Framework 6

Az Entity Framework 6 az egy ORM (Objektum-Relációs Leképző) a .NET Framework-höz. Mivel már nincs fejlesztve, de viszont támogatva van, ezért stabil, közel semmi változás nem történik vele, és csak biztonsági problémákat oldanak meg hozzá.

Előnyei:

* Engedélyezi hogy olyan programokat írjunk, amik relációs adatbázisokat befolyásol erősen típusos .NET objektumokkal.
* Sok féle leképezési fajta, mint például egy az egyhez, több az egyhez kapcsolatok, komplex típusok, öröklődés.

## LocalDB

# Felhasználói dokumentáció

# Irodalomjegyzék

# Mellékletek

## Ábrajegyzék

[1. ábra: Egy tárgy a programból 9](#_Toc130483132)