**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 7](#_Toc131072852)

[2. Feladatspecifikáció 8](#_Toc131072853)

[3. Fejlesztői dokumentáció 8](#_Toc131072854)

[3.1. Általános ismertetés 8](#_Toc131072855)

[3.2. Fejlesztői környezet 11](#_Toc131072856)

[3.3. Visual Studio 2022 Community Edition 11](#_Toc131072857)

[3.4. MonoDevelop 12](#_Toc131072858)

[3.5. Entity Framework 6 12](#_Toc131072859)

[3.6. SQL Server Express LocalDB 13](#_Toc131072860)

[3.7. Az Adatbázis 17](#_Toc131072861)

[3.7.1. Items tábla 18](#_Toc131072862)

[3.7.2. Types tábla 19](#_Toc131072863)

[3.7.3. MobDrops tábla 20](#_Toc131072864)

[3.7.4. Furnace tábla 21](#_Toc131072865)

[3.7.5. Brewing tábla 22](#_Toc131072866)

[3.7.6. CraftingTable tábla 23](#_Toc131072867)

[3.8. Osztályok, változók 26](#_Toc131072868)

[3.8.1. Globális változók 26](#_Toc131072869)

[3.8.2. MainWindow 26](#_Toc131072870)

[3.8.3. CrafingTree 31](#_Toc131072871)

[3.9. Továbbfejlesztési lehetőségek 32](#_Toc131072872)

[4. Felhasználói dokumentáció 33](#_Toc131072873)

[5. Irodalomjegyzék 34](#_Toc131072874)

[6. Mellékletek 34](#_Toc131072875)

[6.1. Ábrajegyzék 34](#_Toc131072876)

# 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:

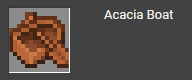
* Tárgyak kezelése és felvétele, típusokkal és dobási eséllyel
* Receptek felvétele
* Alap tárgyak megkeresése egy megadott tárgy receptjéhez

# 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.



1. 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

2. 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

3. 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.

4. 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.

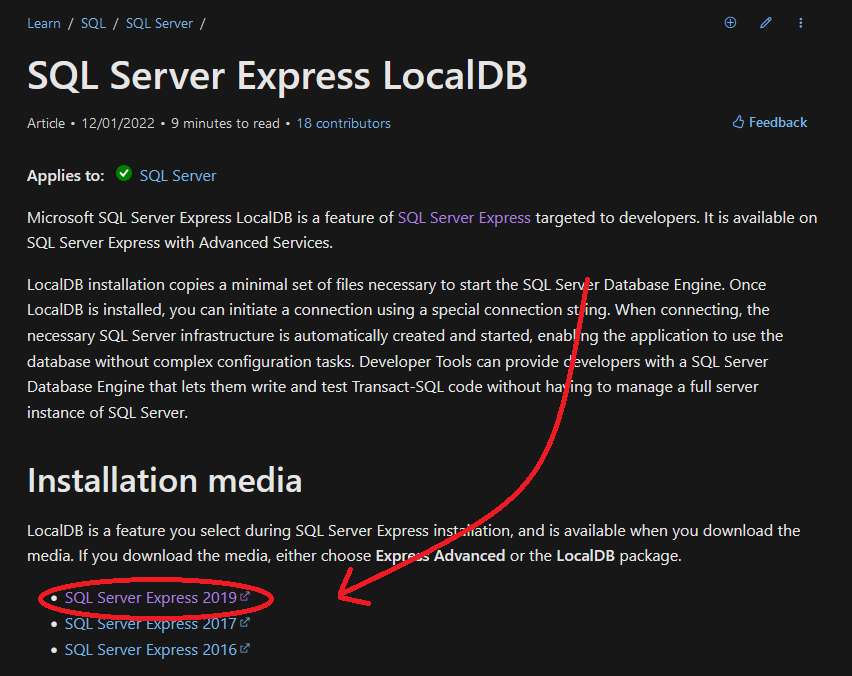
## SQL Server Express LocalDB

A LocalDB az egy Microsoft által fejlesztett és fenntartott SQL alapú adatbázis szerver. Mi ezt használjuk a programunk adatbázisaként, és ezen kezeli az Entity Framework 6 az adatainkat.

Mivel ez egy függősége a programnak, ezért installálni kell, amit a következőképen lehet elvégezni:

* Rákeresünk arra, hogy „LocalDB”, és az első Microsoft oldalt megnyitjuk, vagy beírjuk a keresőbe a következő linket:

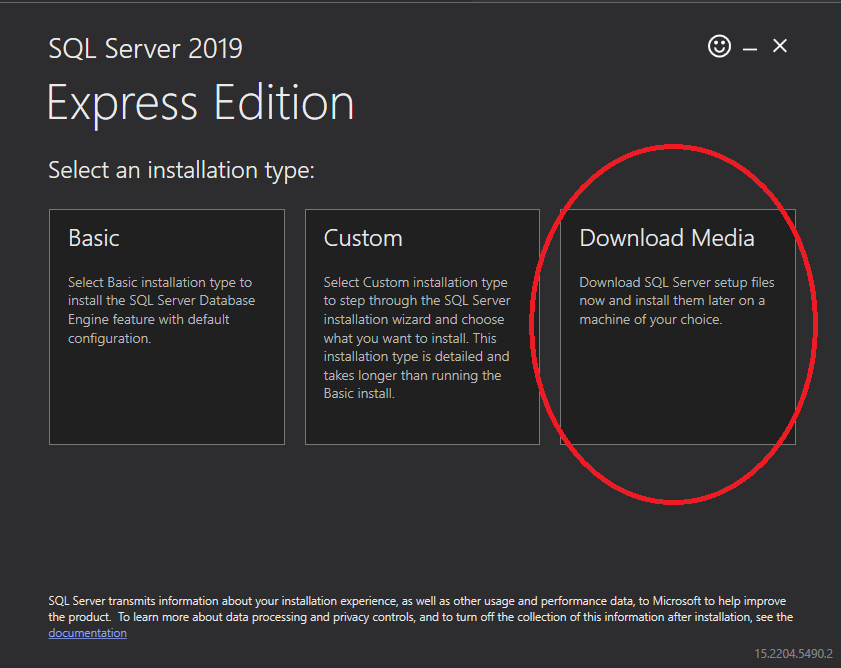
<https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb?view=sql-server-ver16>

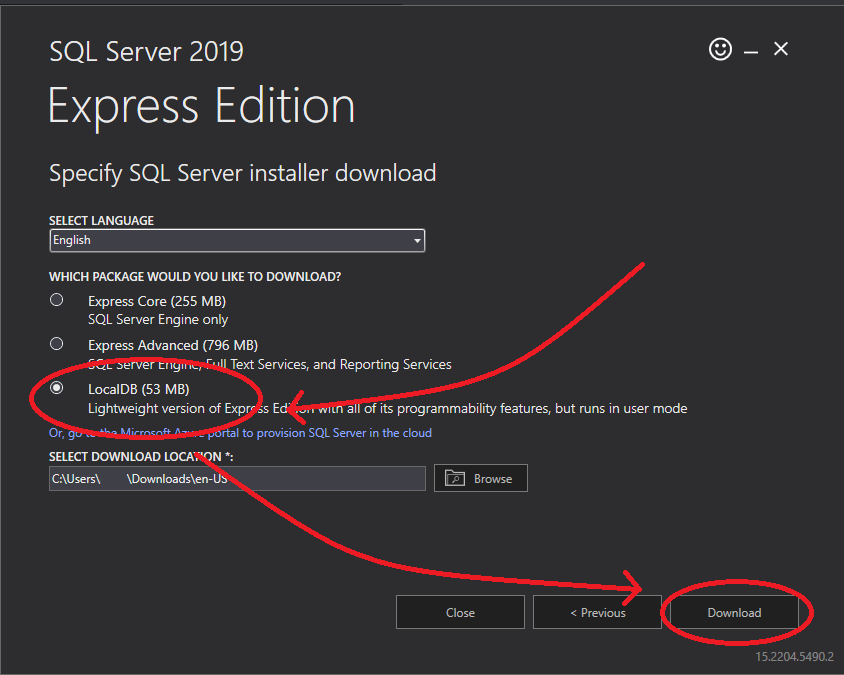
* Meg kell jelennie a következő oldalnak, ahol láthatunk egy „Installation media” menüpontot, az alatt egy „SQL Server Express 2019” nevű linket, és arra rányomunk.
* Ez elkezd nekünk letölteni egy .exe fájlt, amire majd ha kész van rákattintuk vagy a böngésző jobbfelső sarkában való menüből, vagy a „Letöltések” mappából.

A képen szöveg látható

Automatikusan generált leírás

* Ez feldob nekünk egy telepítőt, ahol rá kell kattintanunk a „Download Media” gombra, majd ki kell választanunk a „LocalDB” lehetőséget, és végül rákattintunk a „Download” gombra.



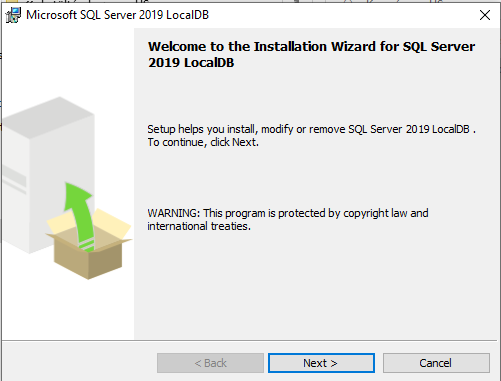


* Miután befejezte a letöltést, rákattintunk arra, hogy „Open Folder”, és ezután bezárhatjuk. A képen diagram látható

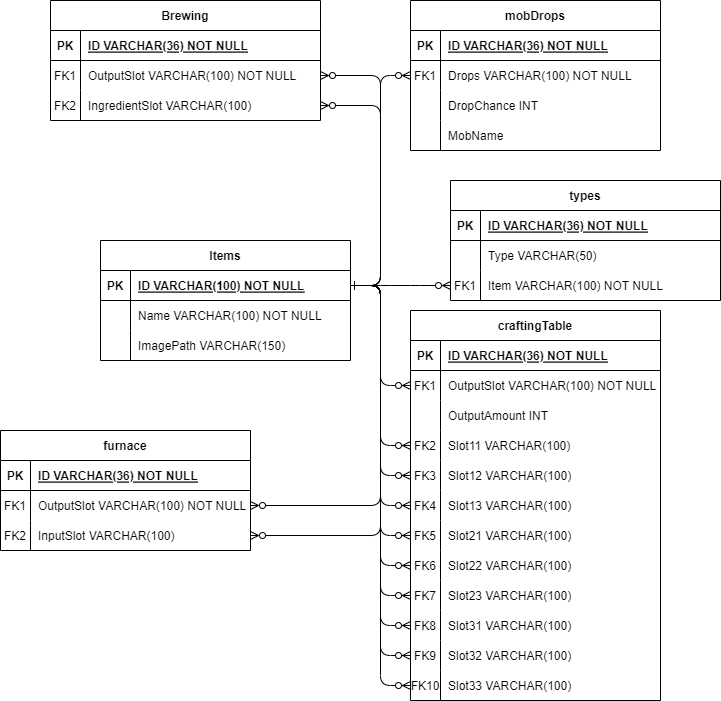
  Automatikusan generált leírás
* Az előbbi gomb megnyitott nekünk egy mappát, amiben van egy „SqlLocalDB.msi” nevű fájl. Ezt megnyitjuk, és feldob nekünk egy telepítőt.

A képen szöveg látható

Automatikusan generált leírás

* Végig megyünk a telepítő lépésein, megvárjuk amíg befejezi, és készen van, már lehet használni. 

## Az Adatbázis



## Items tábla

Az Items tábla tartamazza a következő oszlopokat:

* ID: A játékban szereplő ID, pl.: „minecraft:acacia\_boat”
  + 100 karakter maximum hossz
  + Nem lehet üres
* Name: A játékban megjelenő neve a tárgynak, pl.: „Acacia Boat”
  + 100 karakter maximum hossz
  + Nem lehet üres
* ImagePath: egy relatív útvonal a program .exe fájlától számítva a tárggyal asszociált képhez, pl.: „/ImageResources/Items/acacia\_boat.png”
  + 150 karakter maximum hossz
  + üres lehet, ha nincs hozzá kép
  + NEM egy képet tartalmaz

Az Items tábla a játékban található tárgyak eltárolására van kitalálva, ezeket használja a program majdnem mindegy funkciója.

A következőképen lehet felvenni új elemeket:

* C#, Entity Framework 6:

Items item = new Items(){ ID = "minecraft:item", Name = "Item", ImagePath = "/ImageResources/Items/item.png" };

* SQL:

INSERT INTO Items(ID, Name, ImagePath)

VALUES("minecraft:item", "Item", "/ImageResources/Items/item.png");

## Types tábla

A Types tábla tartalmazza a következő oszlopokat:

* ID: Egy automatikusan generált GUID (Globálisan Egyedi Identifikáció) token, nem a felhasználó által beírt adat.
  + 36 karakter hossz összesen
  + Nem lehet üres
  + Gép által generált
* Type: A típus neve, ami szerint csoportosítva vannak tárgyak
  + 150 karakter maximum hossz
  + Nem lehet üres
* Item: A tárgy, ami kapcsolódik a típushoz
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)

A Types tábla a játékban megtalálható típusokat tárolja, és a majd később továbbfejlesztett verzióban fontos szerepet fog játszani a CraftingTable típus barkácsolási receptjeiben.

A következőképen lehet felvenni új elemeket:

* C#, Entity Framework 6:

Items item = new Items(){ ID = "minecraft:item", Name = "Item", ImagePath = "/ImageResources/Items/item.png" };

Types type = new Types() { ID = Guid.NewGuid().ToString(), Item = item, Type = "type" };

* SQL:

INSERT INTO Types(ID, Item, Type)

VALUES("{generálj le egy GUID-ot és rakd ide}", "minecraft:item", "type");

## MobDrops tábla

A MobDrops tábla tartalmazza a következő oszlopokat:

* ID: Egy automatikusan generált GUID (Globálisan Egyedi Identifikáció) token, nem a felhasználó által beírt adat.
  + 36 karakter hossz összesen
  + Nem lehet üres
  + Gép által generált
* MobName: A játékban szereplő lény neve, ami dobhatja a megadott tárgyat
  + 50 karakter maximum hossz
* DropChance: Mekkora az esély, hogy a megadott lény dobni fogja a tárgyat
  + 0-100 skálájú egész szám
  + Egész számok százalékká válnak, pl.: 1 -> 1%
* Drops: Milyen tárgyat dob a megadott lény
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)

A MobDrops tábla a játékban megjelenő lényeknek azon tárgyait tárolja, amit esélyesen dobhatnak halálukkor.

A következőképen lehet felvenni új elemeket:

* C#, Entity Framework 6:

Items item = new Items(){ ID = "minecraft:porkchop", Name = "Porkchop", ImagePath = "/ImageResources/Items/porkchop.png" };

MobDrops mobDrop = new MobDrops() { ID = Guid.NewGuid().ToString(), MobName = "Pig", DropChance = 100, Drops = item };

* SQL:

INSERT INTO Types(ID, MobName, DropChance, Drops)

VALUES("{generálj le egy GUID-ot és rakd ide}", "Pig", 100, "minecraft:porkchop");

## Furnace tábla

A Furnace tábla tartalmazza a következő oszlopokat:

* ID: Egy automatikusan generált GUID (Globálisan Egyedi Identifikáció) token, nem a felhasználó által beírt adat.
  + 36 karakter hossz összesen
  + Nem lehet üres
  + Gép által generált
* OutputSlot: Milyen tárgy az eredménye a megadott receptnek
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* InputSlot: Milyen tárgyból jön létre a kiadott tárgy
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)

A Furnace tábla a játékban lévő kemencében tett recepteket tárolja. Egy az egyhez átalakítás történik, szóval egy tárgyból csak egy tárgy készülhet.

A következőképen lehet felvenni új elemeket:

* C#, Entity Framework 6:

Items inputItem = new Items(){ ID = "minecraft:porkchop", Name = "Porkchop", ImagePath = "/ImageResources/Items/porkchop.png" };

Items outputItem = new Items(){ ID = "minecraft:cooked\_porkchop", Name = " Cooked Porkchop", ImagePath = "/ImageResources/Items/cooked\_porkchop.png" };

Furnace recipe = new Furnace() { ID = Guid.NewGuid().ToString(), OutputSlot = outputItem, InputSlot = inputItem };

* SQL:

INSERT INTO Furnace(ID, OutputSlot, InputSlot)

VALUES("{generálj le egy GUID-ot és rakd ide}", "minecraft:cooked\_porkchop", "minecraft:porkchop");

## Brewing tábla

A Brewing tábla tartalmazza a következő oszlopokat:

* ID: Egy automatikusan generált GUID (Globálisan Egyedi Identifikáció) token, nem a felhasználó által beírt adat.
  + 36 karakter hossz összesen
  + Nem lehet üres
  + Gép által generált
* OutputSlot: Milyen tárgy az eredménye a megadott receptnek
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* IngredientSlot: Milyen hozzávalóból jön létre a kiadott tárgy
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)

A Brewing tábla a játékban lévő alkímiai asztalban tett összes receptet tartalmazza. A célja, hogy főzeteket hozzon létre vizes üvegekbe.A következőképen lehet felvenni új elemeket:

* C#, Entity Framework 6:

Items ingredientItem = new Items(){ ID = "minecraft:ghast\_tear", Name = "Ghast Tear", ImagePath = "/ImageResources/Items/ghast\_tear.png" };

Items outputItem = new Items(){ ID = "minecraft:regeneration\_potion ", Name = "Regeneration Potion", ImagePath = "/ImageResources/Items/potion.png" };

Brewing recipe = new Brewing() { ID = Guid.NewGuid().ToString(), OutputSlot = outputItem, IngredientSlot = ingredientItem };

* SQL:

INSERT INTO Brewing(ID, OutputSlot, IngredientSlot)

VALUES("{generálj le egy GUID-ot és rakd ide}", "minecraft:cooked\_porkchop", "minecraft:porkchop");

## CraftingTable tábla

A CraftingTable tábla tartalmazza a következő oszlopokat:

* ID: Egy automatikusan generált GUID (Globálisan Egyedi Identifikáció) token, nem a felhasználó által beírt adat.
  + 36 karakter hossz összesen
  + Nem lehet üres
  + Gép által generált
* Slot11: Egy tárgyat tárol, a barkácsasztal bal-felső réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot12: Egy tárgyat tárol, a barkácsasztal közép-felső réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot13: Egy tárgyat tárol, a barkácsasztal jobb-felső réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot21: Egy tárgyat tárol, a barkácsasztal bal-közép réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot22: Egy tárgyat tárol, a barkácsasztal középső réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot23: Egy tárgyat tárol, a barkácsasztal jobb-közép réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot31: Egy tárgyat tárol, a barkácsasztal bal-alsó réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot32: Egy tárgyat tárol, a barkácsasztal közép-alsó réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* Slot33: Egy tárgyat tárol, a barkácsasztal jobb-alsó réséhez tartozik
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* OutputSlot: Milyen tárgy az eredménye a megadott receptnek
  + Egy Items típus, lásd az [Items táblát](#_Items_tábla)
* OutputAmount: Mennyi tárgyat ad a recept
  + Egy természetes szám, felső határa 4 294 967 295

A CraftingTable tábla a játékban szereplő összes barkácsoló receptet tárolja. Ez a fő lényege a programnak, hisz ezeknek az alap tárgyait lehet megkeresni.

A következőképen lehet felvenni új elemeket:

* C#, Entity Framework 6:

Items component1 = new Items(){ ID = "minecraft:flint", Name = "Flint", ImagePath = "/ImageResources/Items/flint.png" };

Items component2 = new Items(){ ID = "minecraft:iron\_ingot ", Name = "Iron Ingot", ImagePath = "/ImageResources/Items/iron\_ingot.png" };

Items outputItem = new Items(){ ID = "minecraft:flint\_and\_steel ", Name = "Flint and Steel", ImagePath = "/ImageResources/Items/flint\_and\_steel.png" };

CraftingTable recipe = new CraftingTable() { ID = Guid.NewGuid().ToString(), Slot11 = component1, Slot12 = component2, Slot13 = nullItem, Slot21 = nullItem, Slot22 = nullItem, Slot23 = nullItem, Slot31 = nullItem, Slot32 = nullItem, Slot33 = nullItem, OutputSlot = outputItem, OutputAmount = 1};

* SQL:

INSERT INTO Furnace(ID, Slot11, Slot12, Slot13, Slot21, Slot22, Slot23, Slot31, Slot32, Slot33, OutputSlot, OutputAmount)

VALUES("{generálj le egy GUID-ot és rakd ide}", "minecraft:flint", "minecraft:iron\_ingot", "-1", "-1", "-1", "-1", "-1", "-1", "minecraft:flint\_and\_steel", 1);

## Osztályok, változók

## Globális változók

* ctx: Egy Context változó, amin keresztül el lehet érni az adatbázist
* nullItem: Egy Items típus, ami arra van használva hogy jelölje az üres réseket a recept típusokban.
* switchScreen: Egy string típusú változó, arra van használva, hogy lementse melyik recept ablakon vagyunk.
* ImagePath: Egy string típusú változó, arra van használva hogy el lehessen érni a legutóbbi lementett tárgy képének a helyét.
* DialogHostKey: Egy string típusú változó, arra van kitalálva, hogy lementse melyik tárgy funkció gombjára kattintottunk.
* RecipeID: Egy string típusú változó, arra van használva, hogy lementse a legutóbbi betöltött recept ID-jét.

## MainWindow

MainWindow osztály tartalmazza az összes funkciót és gombot, ami a fő ablakon szerepel.

Funkciók:

* NewBitmapImage:
  + Egy, a programtól relatív, útvonalat fogad be egy képhez string formátumban.
  + Ezt az útvonalat befogadja, megkeresi a képet, és csinál belőle egy új BitmapImage típusú képet, amit visszaad.
  + A lényege ennek a funkciónak, hogy megakadályozza az Image.Load() metódust, ami megakadályozza, hogy lehessen törölni tárgyakat.
  + Biztosra kell menni hogy nem egy teljes útvonalat adunk meg (pl.: C:/users/pista/MCraftingTree/ImageResources/Items/acacia\_boat.png) hanem egy relatív útvonalat
  + példa:

string path = "/ImageResources/Items/acacia\_boat.png";

BitmapImage bmImage = NewBitmapImage(path);

Ez nekünk visszaadja a következő képet:



* LoadItems:
  + Feltölti a bal oldali adatrácsot (ItemsDG) tárgyakkal
  + Két változata van, az első:
    - Nem fogad be semmilyen adatot, lekéri magának az adatbázisból
    - Aszinkron, hogy a program indításánál ne lassítja indításnál
  + A második:
    - Befogad egy string típusú változót, amit megnézz, hogy szerepel-e az adatbázisban.
    - Nem aszinkron, nem szükséges a jelenlegi mennyiségű adattal.
  + ImageUidSearch:
    - Add\_Recipe gombban van használva, megkeresi egy megadott résnek az Uid-jét, ahol a tárgynak az ID-je van tárolva
    - A beviteli adat egy Border típus, mert a rések legkülsőbb eleme, amit lát a program az egy Border
  + CheckForRecipes:
    - A feladata, hogy megnézze, hogy egy tárgynak vannak-e vele asszociált barkácsoló receptjei
    - Befogad egy Items típusú változót
    - Visszaad egy Items típusú listát
  + CheckForSmelting:
    - A feladata, hogy megnézze, hogy egy tárgynak vannak-e vele asszociált kemencés receptjei
    - Befogad egy Items típusú változót
    - Visszaad egy Items típusú változót

Gombok:

* SwitchScreen:
  + Váltogatja a recept ablakokat
  + switchScreen változót beállítja a megfelelő ablak ID-jére
* Add\_Recipe:
  + Hozzáad az adatbázishoz egy receptet, attól függően hogy milyen ablakon vagyunk
  + Használja a switchScreen változót, hogy eldöntse melyik táblához adja hozzá a receptet
  + Ha a kiadandó tárgy rése üres, nem menti le
  + Ha valamelyik rés a beviteli oldalon üres, hozzáadja a nullItem változót a helyére
* Delete\_Recipe:
  + Töröl egy jelenleg betöltött receptet
  + Használja a switchScreen változót, hogy eldöntse melyik táblából törölje
  + Használja a RecipeID változót, hogy eldöntse melyik receptet kell törölni
  + Újratölti a tárgyak adatrácsát, hogy frissüljenek a receptek.
* NumberOnly
  + Nem egy gomb, hanem a Crafting ablakban lévő beviteli mezőben történő minden gomb lenyomáskor ellenőrzi, hogy át lehet-e alakítani a megadott szöveget egy számmá.
  + Ha üres, akkor nem történik semmi.
* SwitchItem:
  + Jobboldalon váltja a menüket az alapanyagok és a tárgykezelő között.
* DialogHost\_Button\_Prep:
  + Beállítja a DialogHostKey változót, hogy melyik gombra kattintottunk („Add” vagy „Alter”).
  + Ha megnyomjuk az „Alter” gombot, akkor előkészíti a felugró ablakot azzal, hogy belerakja a megváltozandó adatnak a tulajdonságait
* MobDropEnable:
  + A felugró ablakban láthatóvá teszi a MobDrop részt.
* Add\_Or\_Alter\_Item:
  + Használja a DialogHostKey változót, hogy eldöntse melyik gombot nyomtuk meg.
  + Ellenőrzi, hogy az „ItemName” meg az „ItemID” üres-e, ha igen akkor hibád dob.
  + Kettő fő része van, az első:
    - Létrehoz egy Items típusú változót az ItemName és ItemID adataiból
    - Ha a type nem üres, létrehoz egy Types változót
    - Ha a MobName nem üres, létrehoz egy MobDrops változót
    - Ha nem üres az ImagePath változó, hozzáadja az Items típusú változóhoz
    - Hozzáadja az adatbázishoz a létrehozott változókat a megfelelő táblákhoz
  + A második:
    - Frissíti a DialogHost\_Button\_Prep-ben betöltött adatokat
* Delete\_Item:
  + Az tárgyak adatrácsból egy kiválasztott tárgyat töröl az adatbázisból
  + Ha egy másik tárgy nem használja a képet amit ez használ, törli a képet az ImageResources-ból
* OpenFile:
  + Megnyitja a Fájlkezelőt, és engedi hogy kiválassz egy képet vagy gif-et, amit akarsz hogy asszociálva legyen a feltöltendő tárggyal
  + Feltölti a megadott képet az ImageResources/Items útvonalú mappába
  + Beállítja az ImagePath változót a relatív útvonalra a képhez (pl.: ImageResources/Items/acacia\_boat.png)
* Search\_Items:
  + Beolvassa a tárgy adatlista feletti mező értékét
  + Ha üres, akkor meghívja a LoadItems-t adat nélkül
  + Ha nem üres, meghívja a LoadItems-t a megadott szöveggel
  + A Search\_Items\_Click kattintásra, a SearchBar\_KeyDown Enter lenyomására ezt hívja meg
* Item\_Source:
  + A tárgy adatrácsból ha elkezdünk húzni egy képet, akkor ez hívodik meg, és be tudjuk dobni egy Item\_Destination-al rendelkező résbe
* Item\_Destination:
  + A húzott tárgyat be lehet dobni ide, lemásolja a képet és a tárgy ID-jét
* Remove\_Image:
  + Ha egy tárggyal rendelkező résre jobb klikkelünk, eltünteti a tárgyat onnan.
* Load\_Recipe:
  + Ha egy tárgyra kétszer kattintunk a tárgy adatrácsban, három dolog történhet:
    - Ha nincs vele asszociált recept, nem csinál semmit
    - Ha csak egy recept van asszociálva vele, betölti a megadott recept ablakba a receptet
    - Ha több recept is van asszociálva vele, feldob a recept ablak fölé egy adatrácsot, ami tartalmazza az összes vele asszociált receptet, ezek közül lehet rákattintani egyre, ami meghívja a RecipeDG\_Chosen funkciót
  + Használja a switchScreen változót, hogy eldöntse melyik recept ablakon vagyunk és mit tud betölteni
* RecipeDG\_Chosen:
  + A recept adatrácsból kiválasztott receptet betölti a megadott recept ablakba
  + Használja a switchScreen változót, hogy eldöntse milyen ablakon van
* Add\_To\_Tree:
  + Megnézi egy betöltött receptnek az alapanyagait, ezt átalakítja egy CraftingTree típussá, és ezt betölti egy adatrácsba a BaseMaterials menüpontban

## CrafingTree

A MainWindow osztályban szerepel, az egyetlen lényege, hogy fel lehessen használni az Add\_To\_Tree funkcióban az adatrács feltöltésére.

## Továbbfejlesztési lehetőségek

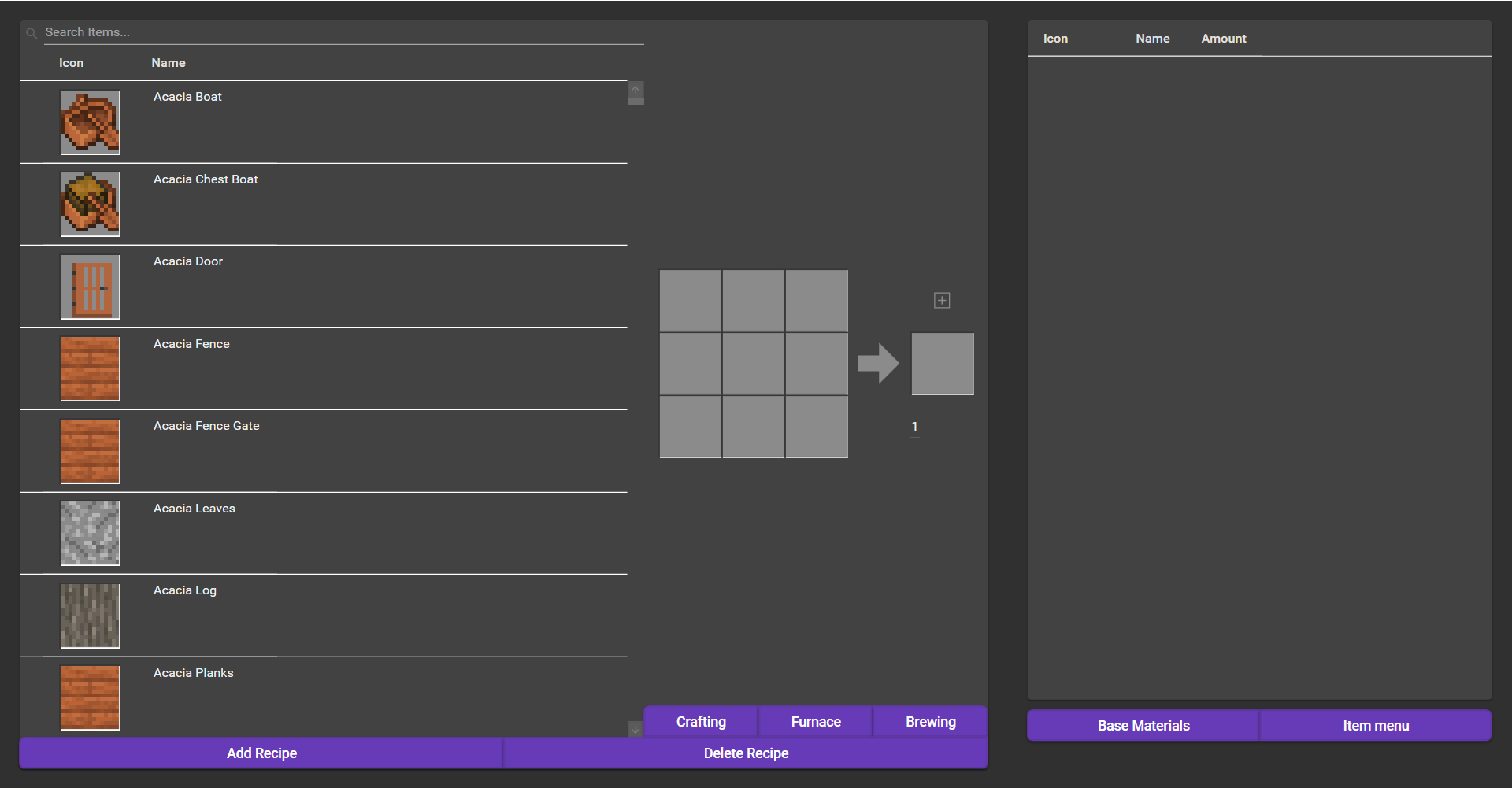
A nagyobb továbbfejlesztési lehetőségek a következőek:

* A receptek használják a típusokat tárgyak helyett ahol megfelelő, és váltogasson a különböző tárgyak között a betöltött recept.
* Lehessen beolvasni egy JSON fájlt, ami tartalmazza a recepteket és típusokat.
* Lehessen beolvasni Java osztályfájlokat, amik a tárgyakat tartalmazzák.
* Csináljon egy képet a Add\_To\_Tree funkció, ami mutatja az útvonalat hogy hogyan lehet megcsinálni egy megadott tárgyat.
* A MobDrops tábla is használatban legyen a Add\_To\_Tree funkcióban, és mutassa a lénynek a fejét a képben.
* Úgy jelenjenek meg a tárgyak és a lények, mint a játékban, azaz egy 3D-s objektumként.

# Felhasználói dokumentáció

## Mi az a MCraftintTree?

Az MCraftingTree az egy magánálló program a Minecraft nevű játékhoz kitalálva. A lényege az az, hogy meg lehessen keresni egy tárgynak az alapanyagait. Az alkalmazást lehet használni Minecraft tárgyak és receptek felvételére és kezelésére. A program szolgáltatja a felhasználót az alap, meg nem változtatott játék tárgyaival és receptjeivel, és ha a felhasználó egy felhasználók által készített modifikációnak tárgyait és receptjeit szeretné használni, azt magának kell felvennie az adatázisba.



## Rendszerkövetelmények

A minimum gépigény:

* CPU: Intel Core I5-2500 / AMD FX 6300
* Memória: 2 GB RAM
* Tárhely: 50 MB szabad
* DirectX: 9
* Grafikai kártya: Nvidia NVS 310 / AMD Radeon HD4670M

A program nem nagy gépigényű, és elfut mindenen, amin Windows 10 elfut, szóval a leírt minimum gépigények, a tárhely kivételével, mind a Windows 10 gépigényei.

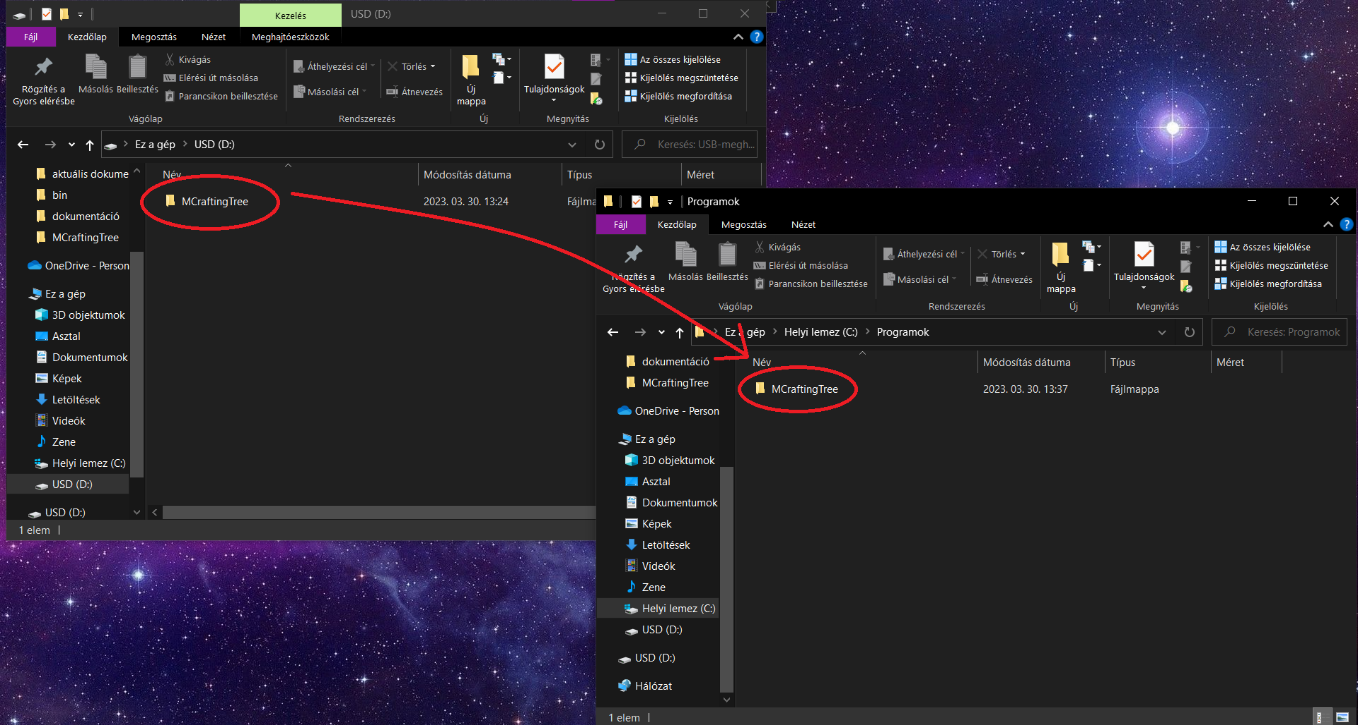
## Telepítés

A dokumentáció leghátsóbb oldalán van egy cd, az azon szerepelnek egy „MCraftingTree” mappa.

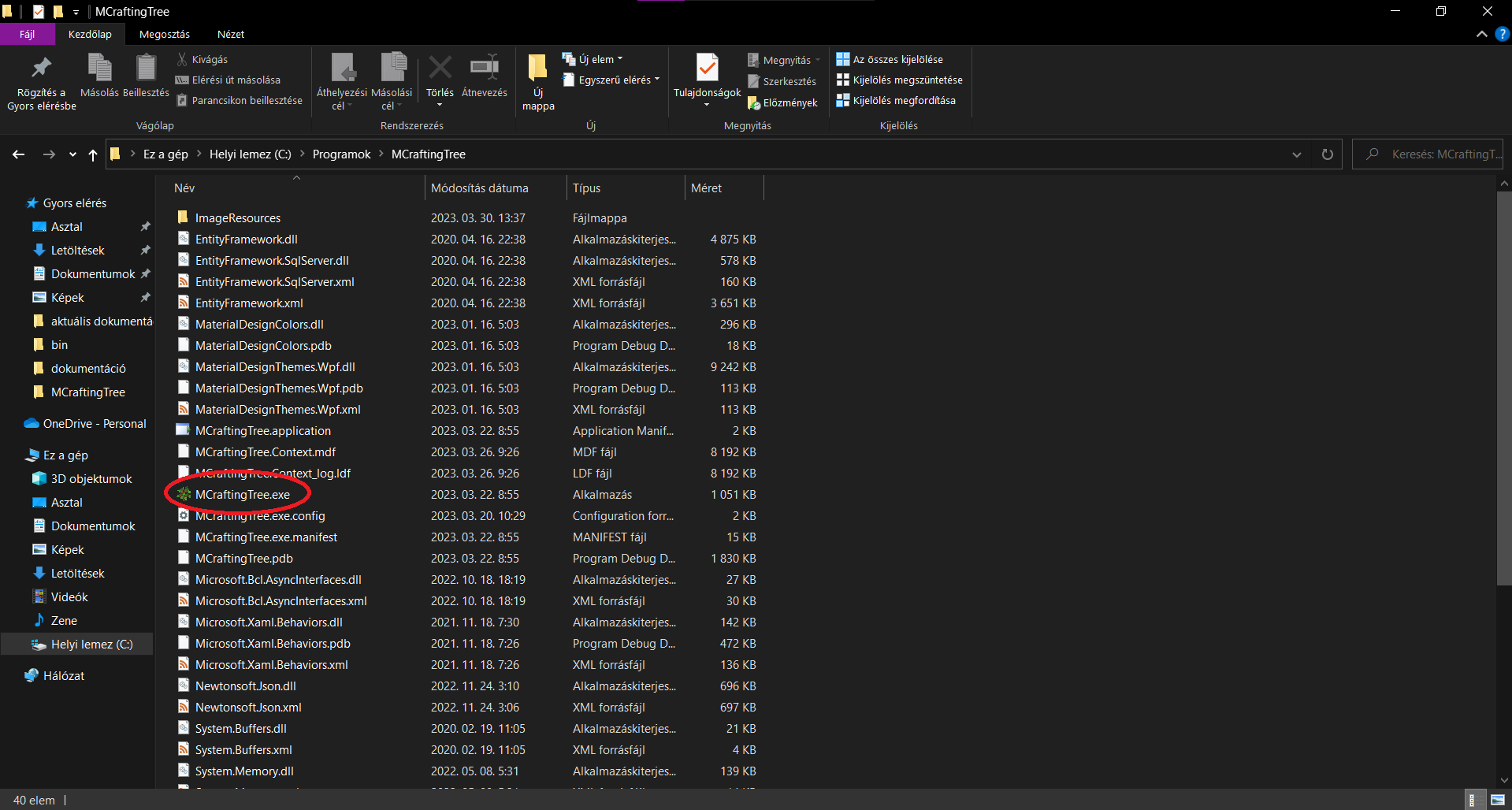
A képen szöveg látható

Automatikusan generált leírás

Erre rákkatintunk, és megnyomjuk a CTRL és C billentyűt, és egy ideális helyen megnyomjuk a CTRL és V gombot, vagy csak rákattintunk a mappára, és áthúzzuk a gépre.



Ezután el tudjuk indítani a programot a MCraftingTree.exe fájlal.



Ha a program nem indul el, telepíteni kell a LocalDB-t, amire a telepítési folyamat le van írva a Fejlesztői dokumentáció részben (Tartalomjegyzék 3.4.)

# Irodalomjegyzék

# Mellékletek

## Ábrajegyzék

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