WEB-ES ALKALMAZÁSOK- GUZEBAL-WEBALKAL-1

Microsoft PowerApps – Calculator applikáció

SCHILLER VIKTOR – GWOQXX

MÁSODIK ÉVFOLYAM – 2. SZEMESZTER

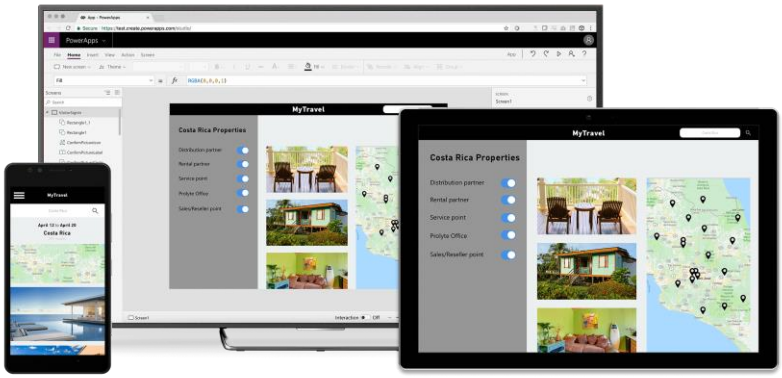
NJE GAMF MŰSZAKI ÉS INFORMATIKAI KAR

Tartalom

1. Microsoft Power Apps

1.1. Egyszerű alkalmazás készítése

A Power Apps alkalmazások, szolgáltatások és összekötők gyűjteménye, illetve egy adatplatform amely gyors alkalmazásfejlesztéshez kínál környezetet saját üzleti igényekhez igazodó egyéni alkalmazások létrehozásához. A Power Apps használatával gyorsan létrehozhat olyan egyéni üzleti alkalmazásokat, amelyek csatlakozni tudnak adataihoz akár a mögöttes adatplatformon (Microsoft Dataverse), akár különféle online és helyszíni adatforrásban tárolva, például SharePoint, Excel, Microsoft 365, Dynamics 365, SQL Server stb.

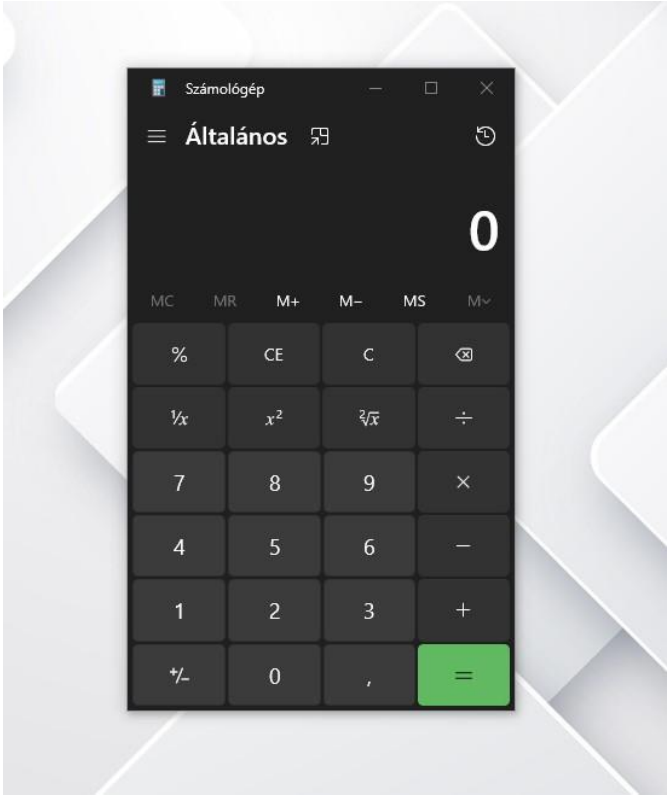


1. ábra. Microsoft PowerApps platform

A Power Apps használatával háromféle alkalmazást hozhatunk létre: vászonalapú (canvas), modellvezérelt és portál. Én a Canvas App fejlesztése mellett döntöttem. Egy ingyenes regisztráció után lehetőségünk van az Egyetemi Teams-es email cím mellett fejleszteni, kipróbálni a fejlesztőfelületet. Egy számológép alkalmazás lekódolása mellett döntöttem, mivel a Vizuális Programozás tantárgyra mind MITAppinventor, mind pedig .NET Maui nyelveken ezt programoztam le.

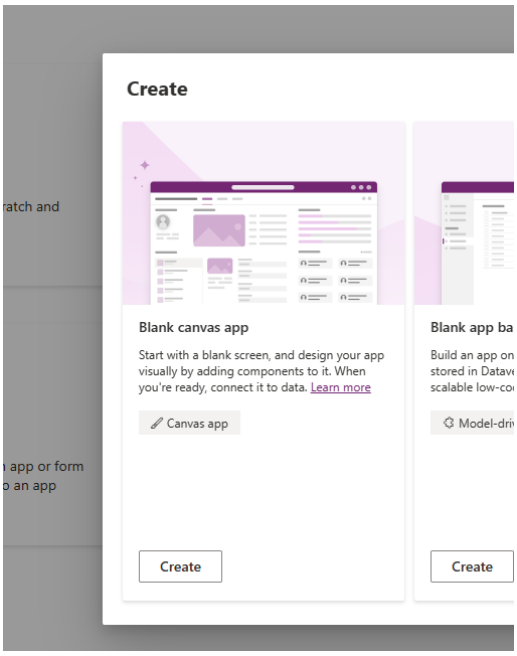
2. Elrendezés kialakítása

Első lépésben kerestem egy minta számológép alkalmazás kinézetet, ami sablonként fog szolgálni az egységek, gombok, kijelző, a komplett GUI elrendezését illetően, amolyan beépített Windows-osat, letisztult design-al, használható felülettel. Íme:



2. ábra. A számológép-minta (Win10 default Calculator)

Először létrehoztam egy új projektet a menüsorban a *+Create > New Blank Canvas App* menüpontnál. Legyen a projekt neve Calculator. Ezután a képernyő kinézetét kellett felépítenem a *Screen* nézetben.



3. ábra. Új Canvas App létrehozása

2.2 Kijelző ablak létrehozása

Kialakítom a képernyő tartalmát. A tervezésnél meg kellett határoznom a változókat, amik szerepelni fognak a kalkulátorban és annak kijelzőjén. A '_Result' = az eredmény változója, ami int típus, az '_Operandus' a műveleti jel változója, ez sztring típusú, a '_Valtozo' dallamos nevet adtam az értéknek, amit a felhasználó megad, mint kiszámolandó számot, tehát int típus, illetve a '_decimal', ami a tizedesvessző és helyiértékek helyét szabályozza. Az alkalmazás megfelelő méretei, szélesség, magasság, nyújtás, szín oldalsávon történő beállítása után még megadtam az OnVisible funkcióját is:

```
UpdateContext({_Result:0;_Operandus: "";_Valtozo:0;_decimal:0})
```

Ez a függvény az App indításakor az _Result, _Operandus, _Valtozo és _decimal változók értékeit nullára állítja, ezt fogja megjeleníteni az alkalmazás start helyzetben.

Ezekután az ablak felső részén két „felirat" típusú kisebb kijelző lett elhelyezve, a felső kisebb karakterekkel az épp beütött változót és a műveletet, míg az alsó, nagyobb betűméretű az eredményt hivatott mutatni. Így jöttek létre a 'Kijelzo_felso' és 'Kijelzo_Result' feliratok. Ezek 'Text' kódjai a következők voltak:

```
If(_Operandus="#";Round(_Valtozo;3)&"";!IsBlank(_Operandus);Round(_Result;3)&"  "&
_Operandus &" "&Round(_Valtozo;3);Round(_Result;3))
```

Dióhéjban annyit jelent, hogy a szám változó beírása után, ha nem adunk meg műveleti jelet, a szám a felső sorban kiíródik három tizedesjegyre kerekítve megadva. Ha azonban van műveleti jel, akkor a _Result, _Operandus és _Valtozo változók értékeit egy kifejezésbe állítja össze egy sztringben összefűzve.

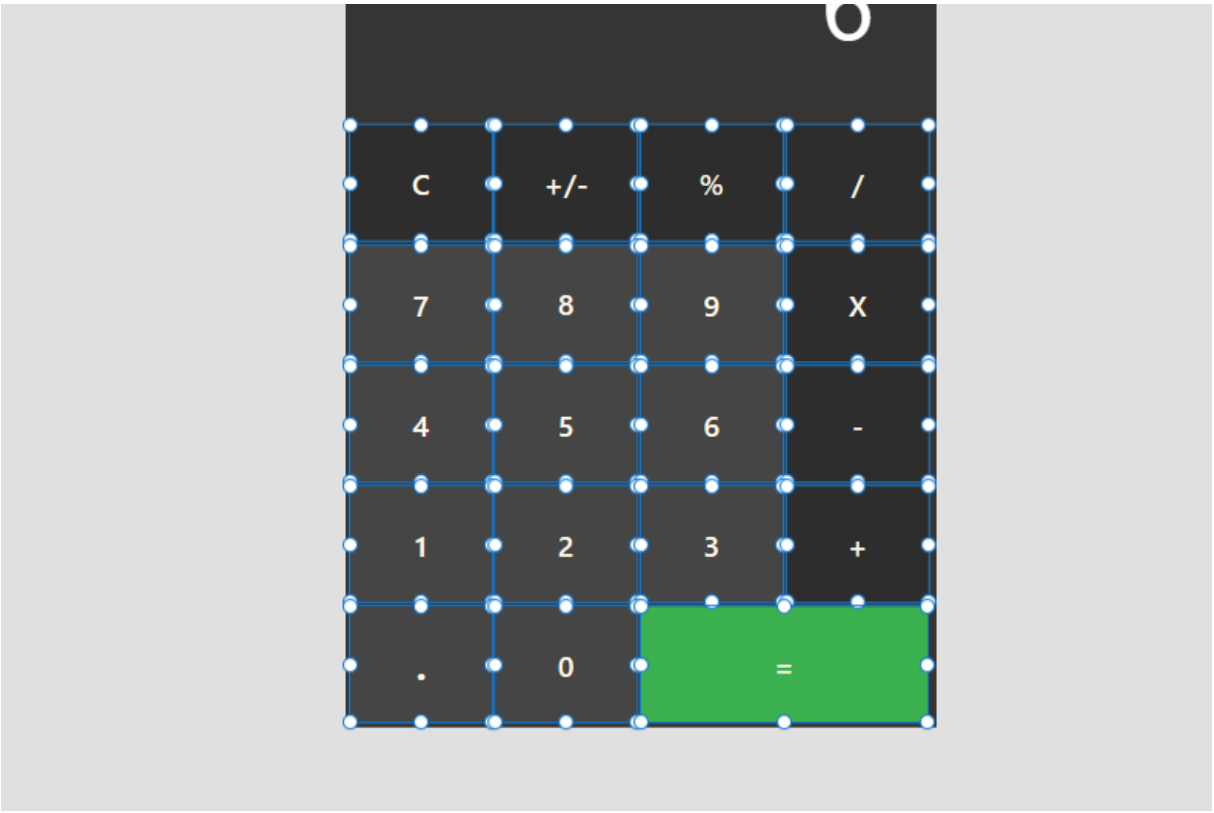
A 'Kijelzo_Result' Text típusú kódja:

```
Round(If(IsBlank(_Operandus) || _Valtozo=0; _Result;_Valtozo);6)
```

A kódrészlet visszaad egy számot, amely az _Operandus és az _Valtozo változók értékétől függ, de ha az _Operandus üres, vagy az _Valtozo értéke nulla, akkor az _Result változó értékét adja vissza.

2.3 Gombsor létrehozása

Ezek után létrehoztam a kijelző alatti területen egy „táblázatos elrendezés” -t a számológép gombjainak helyt adva. Az oszlopok száma: 4, a sorok száma: 5 lett, a mintakép alapján.



4. ábra. A számológép-gombsor-táblaelrendezés kialakítása (alsó rész)

A gombsor megalkotásakor célszerűen a funkciónak -számomra megfelelő- elnevezéssel láttam el a gombok nevét. Az egyenlőségjel két-oszlop összevonásnyi széles lett, a jobb olvashatóság miatt. Az adott relációk a következők voltak:

A négy alapművelet: Összeadás; Kivonás; Osztás; Szorzás;

kiegészítve: Tizedesjegy; Százalék; Előjelváltás; Törlés – gombok.

3. A button-okhoz tartozó programkód

A gombok 'Tulajdonságok' menüjében ki lett alakítva a háttérszín, méret, elnevezés, betűszín, a rádiusz-azaz a gombfelület görbülete-, és ami a legfontosabb, az 'OnSelect' kód művelet, amit a 'Speciális' fül alatt kellett beállítani.

„C" – törlés gomb = UpdateContext({_Result:0;_Operandus: "";_Valtozo:0;_decimal:0})

_Result: 0

_Operandus: üres karakterlánc (""), azaz nincs értéke

_Valtozo: 0

_decimal: 0

A kódrészlet tehát ezeknek a változóknak az értékét állítja be a fentebb említett értékekre.

+/- Előjelváltó gomb = If(IsBlank(_Operandus);UpdateContext({_Result:-_Result});UpdateContext({_Valtozo:-_Valtozo}))

Ez a kódrészlet végrehajtja az adott változók értékeinek megváltoztatását (előjelet vált), ha az _Operandus üres, illetve ha nem üres, akkor a másik változó értékének megváltoztatását hajtja végre. Az UpdateContext függvényeknek köszönhetően az új értékek későbbi számításokban vagy logikai feltételekben felhasználhatók.

%, +, -, x, / - műveleti jel gombok = UpdateContext({_Result:If(_Operandus="#"; _Valtozo;
_Operandus="+"; _Result+_Valtozo;
_Operandus="-"; _Result-_Valtozo;
_Operandus="*"; _Result*_Valtozo;
_Operandus="/"; _Result/_Valtozo;
_Operandus="%"; _Result*(_Valtozo/100);
_Result);
_Valtozo:0;
_decimal:0;
_Operandus:"%"})

Ez egy olyan UpdateContext függvény, amelyben a változók értékeit állítottam be különböző feltételek szerint.

Az első változó, amelyet beállítottam, az ’_Result’ változó, és az értéke attól függ, hogy milyen az ’_Operandus’ változó értéke. A ’_Operandus’ változó az az operátor, amely az előző eredményhez hozzáadódik, kivonódik, szorzódik, osztódik vagy százalékszorzó lesz.

Ha az ’_Operandus’ változó értéke "#" (üres karakterlánc), akkor az ’_Result’ értéke ’_Valtozo’ lesz. Ez azt jelenti, hogy ha nincs műveleti operátor, akkor a ’_Valtozo’ értéke lesz az eredmény.

Ha az ’_Operandus’ értéke "+" (pluszjel), akkor az ’_Result’ értékéhez hozzáadják a ’_Valtozo’ értékét. Ez az összeadást jelenti.

Ha az ’_Operandus’ értéke "-" (mínuszjel), akkor az ’_Result’ értékéből kivonják a ’_Valtozo’ értékét. Ez a kivonást jelenti.

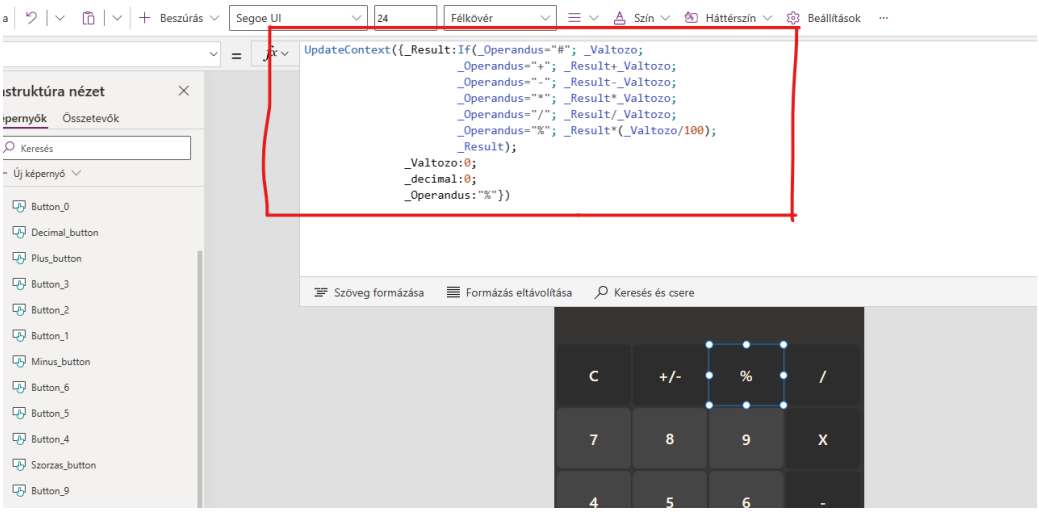
Ha az ’_Operandus’ értéke "*" (szorzójel), akkor az ’_Result’ értékével megszorozzák a ’_Valtozo’ értékét. Ez a szorzást jelenti.

Ha az ’_Operandus’ értéke "/" (osztásjel), akkor az ’_Result’ értékét elosztják a ’_Valtozo’ változó értékével. Ez az osztást jelenti.

Ha az ’_Operandus’ értéke "%" (százalékjel), akkor az ’_Result’ értékével megszorozzák a ’_Valtozo’ százalékos értékével. Ez a százalékos szorzást jelenti.

A második változó, amelyet beállítottam, az ’_Valtozo’ variáns, és az értéke 0. Ez azt jelenti, hogy ha a kódban újabb műveletre van szükség, akkor a ’_Valtozo’ változóban lévő értéket használják fel.

A harmadik változó, amelyet beállítottam, az ’_Operandus’ változó, és az értéke az aktuális operandus. Ez azt jelenti, hogy ha a felhasználó a következő műveletnél nem ad meg újabb műveleti operátort, akkor az adott műveletet végzi el.



5. ábra: Műveleti jelek kódjai

= - Egyenlőségjel gomb = UpdateContext({_Result:If(_Operandus="#"; _Valtozo;

_Operandus="+"; _Result+_Valtozo;

_Operandus="-"; _Result-_Valtozo;

_Operandus="*"; _Result*_Valtozo;

_Operandus="/"; _Result/_Valtozo;

_Operandus="%"; _Result*(_Valtozo/100);

_Result);

_Valtozo:0;

_decimal:0;

_Operandus:""})

Itt minden ugyanaz maradt, mint a műveleti jelek változóinál, azzal a különbséggel, hogy az '_Operandus' változót üres sztringgé alakítja, ezáltal nem hajt végre újabb relációt, a '_Result' változóba bekerül az épp aktuális eredmény és megjeleníti azt.

. – Tizedesvessző gomb = UpdateContext({_decimal: 1})

A változó értékét 1-re állítja (_decimal > 0).

1,2,3,4,5,6,7,8,9,0 = Számjegy gombok = UpdateContext({

_Valtozo:If(_decimal>0;_Valtozo+If(_Valtozo<0;-1;1)*10^-_decimal*9; _Valtozo*10+If(_Valtozo<0;-1;1)*9);

_decimal: If(_decimal>0; _decimal+1; 0);

_Operandus: If(IsBlank(_Operandus); "#"; _Operandus)

})

Ha a '_decimal' változó nagyobb, mint 0, akkor először kiszámolja a hozzáadandó számot, ami a (10^-_decimal) (10 alapú hatvány, ahol a kitevő a negatív _decimal érték) szorozva az adott számmal. Ha a '_Valtozo' negatív, akkor a számot -1-gyel szorozza, hogy az eredmény is negatív legyen. Ha a '_decimal' változó nem nagyobb, mint 0, akkor a már tizedessé konvertált '_Valtozo' értékét „visszateszi" egy tizedesjeggyel, és megszorozza a beütött számmal.

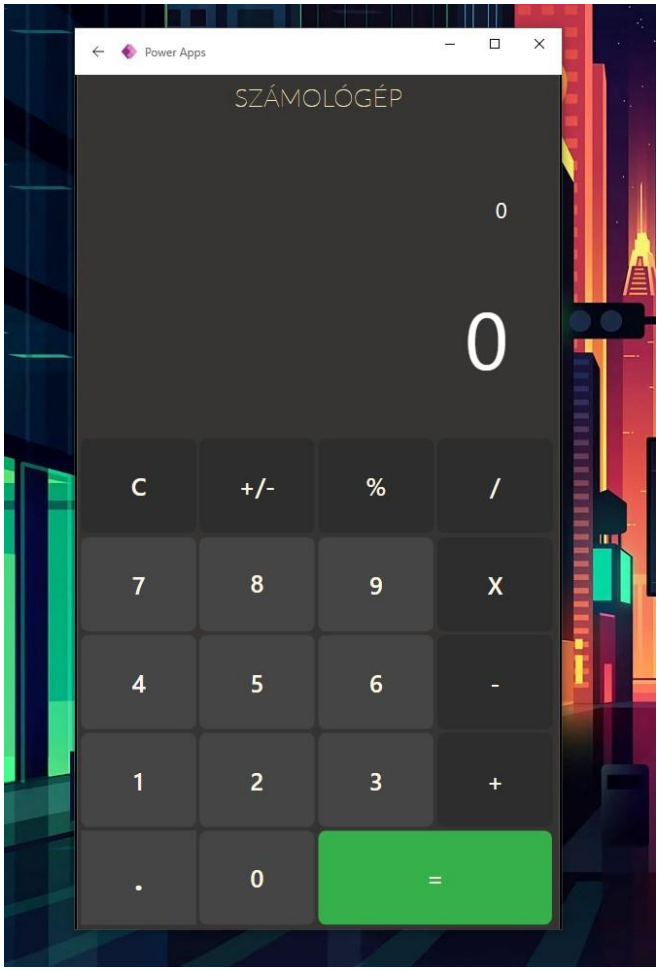
Ezután beállítja a _decimal változót is. Ha a _decimal nagyobb, mint 0, akkor azt növeli 1-gyel, hogy az új számjegyeket hozzáadhassa a már tizedesjegy '_Valtozo'-hoz. Így század, ezred, stb. értékek kerülnek a kijelzőre. Ha nem, akkor a '_decimal' értékét 0-ra állítja.

Végül a _Operandus változót is beállítja. Ha az üres, akkor beállítja az értékét "false" karakterre, ha nem üres, akkor a megadott '_Operandus' értéket használja, azaz a felhasználó folytatja a számolást valamilyen műveleti jellel.

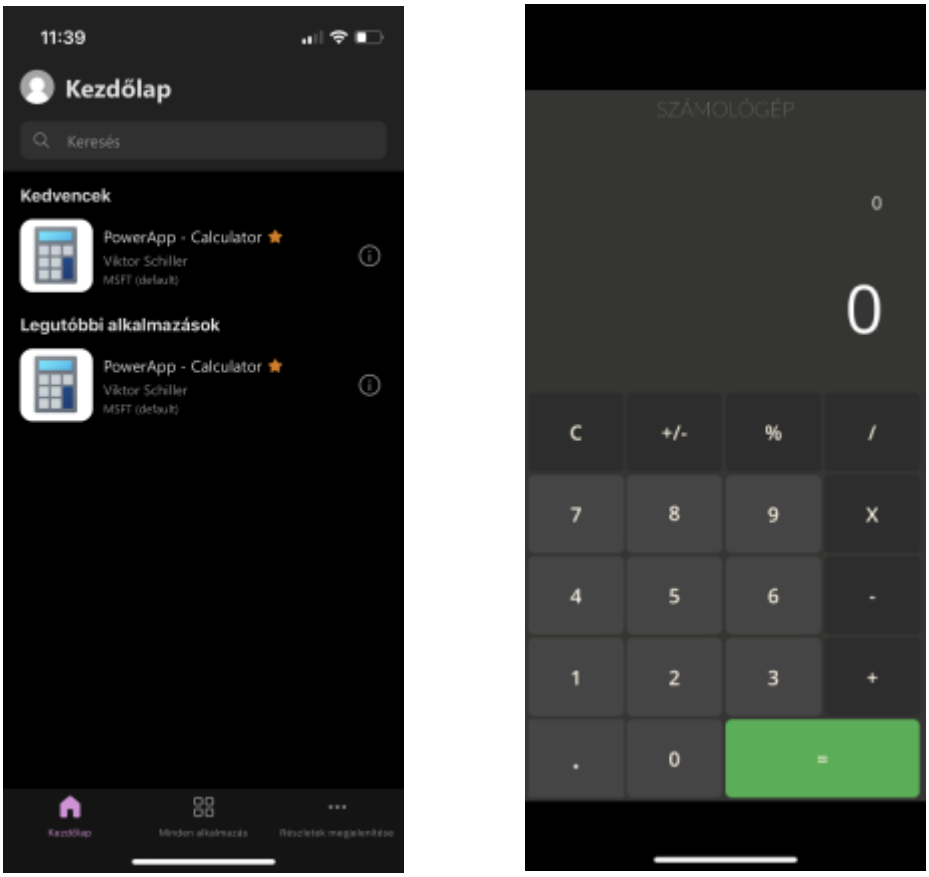
Eltérések a számjegy gombok kódban csak a '_decimal' változóval való szorzásokban vannak, annak megfelelően, hogy a felhasználó éppen melyik számjegyet üti be.

4. Összegzés

A PowerApps funkcióival Win10-en és IOS-en is lefutott a projekt, mindkét esetben zökkenőmentesen. Egyedi ikont adtam még az alkalmazáshoz. A számológép számítási teljesítménye természetesen rendkívül alap, mindössze a négy fő műveletet plusz százalék és az előjelváltás funkciót tartalmazza, de úgy lett megírva, hogy később tovább bővíthető legyen, akár tudományos számológéppé is. Design-ja módosítható, de jelenleg is letisztult, és próbáltam követni a 'minta' fotón szereplő Kalkulátor kinézetét. Az eredmény magáért beszél:



6. ábra: Windows10 22H2 operációs rendszeren



7. ábra: IOS 16.4.1 operációs rendszeren

FELHASZNÁLT IRODALOM & KÉPEK

- Microsoft Power Apps dokumentáció Microsoft Power Apps-dokumentáció - Power Apps | Microsoft Learn
- GitHub alkalmazás schillerviktor (Schiller Viktor) (github.com)
- PowerApps Tricks & Tipps Top 10 Power Apps Tips Of 2020 - Matthew Devaney
- Főcím kép https://learn.microsoft.com/hu-hu/power-apps/media/powerapps-intro.png