|  |  |
| --- | --- |
|  | Eötvös Loránd Tudományegyetem  Informatikai kar  Média- és Oktatásinformatikai tanszék |

**Egyedi alkalmazás fejlesztése kvízjátékokhoz**

|  |  |
| --- | --- |
| *Témavezető:*  Dr. Illés Zoltán  habilitált egyetemi docens | *Szerző:*  Ömböli Csaba  programtervező informatikus BSc |

*Budapest, 2022*

Tartalomjegyzék

[1. Témabejelentő 3](#_Toc99903562)

[2. Bevezetés 4](#_Toc99903563)

[2.1. A dolgozat tartalma 4](#_Toc99903564)

[2.2. Mindent vagy semmit 4](#_Toc99903565)

[2.2.1. Az eredeti vetélkedő 4](#_Toc99903566)

[2.2.2. Saját szabályok 5](#_Toc99903567)

[3. Felhasználói dokumentáció 7](#_Toc99903568)

[4. Fejlesztői dokumentáció 7](#_Toc99903569)

[4.1. Főbb tervezési irányvonalak 7](#_Toc99903570)

[4.2. Architekturális döntések 9](#_Toc99903571)

[4.3. Navigáció 9](#_Toc99903572)

[4.3.1. Shell 10](#_Toc99903573)

[4.3.2. PageService 10](#_Toc99903574)

[4.3.3. NavigationViewService 10](#_Toc99903575)

[4.3.4. NavigationService 11](#_Toc99903576)

[4.4. Az Adatelérési réteg 11](#_Toc99903577)

[5. Összegzés 11](#_Toc99903578)

[6. További fejlesztési lehetőségek 11](#_Toc99903579)

[7. Hivatkozások 12](#_Toc99903580)

# Témabejelentő



# Bevezetés

## A dolgozat tartalma

A szakdolgozat keretében elkészített program alapjaiban a témabejelentőhöz illeszkedik, de a tervezési, és implementálási fázis során felmerülő új szempontok, körülmények és nehézségek miatt helyenként minimálisan eltérhet attól.

Az elkészült dolgozat 2 önálló részre bontható:

**Indító-alkalmazás**

Tartalmaz egy olyan indító-alkalmazást, amely keretet ad más programok rendszerezéséhez, és lehetővé teszi, a felhasználónak, hogy amennyiben az eszközén több játékkal rendelkezik, egy helyről kiválaszthassa, hogy melyiket szeretné elindítani. A program egyszeri konfigurálásával egy helyről, gyorsan indíthat el kvízjátékokat, kiválasztva a rendelkezésre állók közül a neki leginkább tetszőt.

**Mindent vagy semmit program**

A másik fő komponens, egy olyan alkalmazás, ami segítséget nyújt a Mindent vagy semmit televíziós vetélkedőben megismert játékmenet előkészítésében, zökkenőmentes lebonyolításában, és dokumentálásában. Az alkalmazás belekerült a korábban említett indító-alkalmazásba, de önállóan is használható.

A szakdolgozat két alkalmazást tartalmaz, de mivel mindkettőt ugyanolyan tervezési szempontok mentén, nagyon hasonlóan került megvalósításra, ezért a továbbiakban általánosan mindkettőre vonatkozóan írom a tudnivalókat. Ha valamely szempontnál a két alkalmazás eltér, akkor térek ki külön egyesével rájuk.

## Mindent vagy semmit

### Az eredeti vetélkedő

„A vetélkedő 1997-ben indult az akkor újnak számító TV2 képernyőjén, és minden hétköznap az esti műsorsávban jelentkezett. A játék fődíja egy autó volt, ezen kívül pénz- és tárgynyeremények sokasága talált gazdára minden adásban. Az egész az amerikai Joepardy című kvízműsorra épült. Minden adásban három játékos versenyzett, a nap győztese pedig eldönthette, hogy megtartja-e azt a pénzt, amit nyert és kiszáll, vagy visszatér a következő adásra. Aki egy bizonyos számú játékban részt vett az vihette haza a főnyereményt, az autót.” (Vollai, 2019)

A Vágó István nevével fémjelzett műsorban több játékmódban, általános műveltségi kérdésekre válaszolva juthattak közelebb a nyeremények elnyeréséhez a játékosok.

### Saját szabályok

Középiskolai éveim alatt megismerkedtem egy az eredeti vetélkedő szabályaival szinte teljesen megegyező változattal. Az általam elkészített program ennek a variánsnak a szabályrendszerét követi.

Ahhoz, hogy a játékot ebben a formában lehessen játszani, szükség van egy fizikai nyomógomb rendszerre, aminek segítségével eldönthető, hogy ki adhat választ az adott kérdésre.

**A meglévő rendszer technikai részletei**

A jelenlegi rendszer[[1]](#footnote-1) egy hálózati áramforrásra kötött központi egységből, és az ehhez csatlakoztatott négy darab nyomógombból áll. A központi egység biztosítja a gombok az számára az áramellátást, és kezeli az azok lenyomásából származó input jeleket. Ha egy nyomógombot lenyomunk, a gomb előtt található lámpa világítani kezd, a központi egység pedig (logikai áramkörök segítségével) biztosítja, hogy eztkövetően más gombok lenyomására ne világítson a saját lámpájuk, tehát mindig csak az elsőként lenyomott gombhoz tartozó lámpa világítson. A központi egységen található „nullázó” gomb lenyomásával alaphelyzetbe állíthatjuk a rendszert. (Ha van olyan lámpa, ami épp világít, az kialszik, és rendszer ismét inputra vár.)

**A játék menete**

A játékot az eredeti vetélkedőhöz hasonlóan egy műsorvezető/játékmester vezeti. Neki (általában kinyomtatott papíralapú, vagy Word formátumban) rendelkezésére állnak a játékban felhasználásra kerülő kérdéssorok.

A játékosok számától, és az alkalomtól függően lehet egyéni, vagy csapatos játék. Egyéni játék esetén mindenki egyedül áll ki a gombokhoz és versenyez, míg csapatos játéknál az egész csapat egyszerre használja ugyanazt a nyomógombot válaszadási szándékának jelzéséhez. (Az egyszerűség kedvéért a továbbiakban nem különböztetünk meg egyéni játékost, mivel az felfogható egy egyszemélyes csapatként is.) A csapatokat valamilyen szabály szerint[[2]](#footnote-2) körökre osztjuk. A körökre osztásnak fontossági sorrendben a következő preferenciái szoktak lenni:

1. minden csapat legalább n kört játsszon a döntőt nem számolva, (n > 0, n E N)
2. az egy körben szereplő csapatok száma minél egyenletesebb legyen (pl: 9 csapat, és n = 1 esetén rossz megoldás 2 négyfős és 1 egyfős kör kialakítása. Az optimális megoldás 3 háromfős kör.)
3. minél kevesebbszer játsszon egymás ellen két csapat

A körök kialakítása után a csapatok elfoglalják a helyüket egy-egy nyomógomb mögött, és a játék kezdetét veszi.

Egy kör két szakaszból áll. A csapatok feladata mindkét szakaszban az, hogy minél több pontot szerezzenek, a kérdésekre adott helyes válaszaikkal, fontos azonban, hogy a helytelen válaszokért a megválaszolt kérdés pont értékével megegyező levonás jár. (A csapatok akár negatív összpontszámot is elérhetnek.) Az első szakaszban ún. tematikus kérdéssorok kérdéseire lehet válaszolni. Egy kérdéssor 5 témából áll. Minden téma 6 egyre nehezebb kérdést tartalmaz, sorrendben a következő pont értékben: 1000, 2000, 3000, 4000, 5000, 8000. Mindig az előző kérdésre jó választ adó csapat kérheti a következő kérdést, ha ilyen nincs, akkor a legtöbb ponttal rendelkező csapat kérhet. Pont egyenlőség esetén a játékmester dönti el, hogy ki kérhet. A kiválasztott kérdést a játékmester elmondja, a csapatok pedig nyomógombjaik lenyomásával jelezhetik válaszadási szándékukat. (Bármelyik csapat nyomhat, nem csak az amelyik a kérdést kérte.) Mindig az a csapat adhat választ, amelyiknek világít a lámpája.

A második szakaszban „villámkérdésekre” lehet választ adni. Itt a játékmester sorban egymás után tesz fel véletlenszerű témákból kérdéseket. Ebben a szakaszban az összes kérdés 3000 pontot ér. Mivel itt nincsenek témák, amikből választani lehetne, ezért egy válsz elhangzása után a játékmester automatikusan folytatja a következő kérdéssel. Fontos, hogy míg a tematikus szakaszban a játékvezető megengedőbb, azzal kapcsolatban, hogy mennyi idő telik el a gomb lenyomása, és a válasz elhangzása között, addig a villámkérdéseknél a válasznak azonnal kell jönnie, különben a választ helytelennek ítéli meg.

A tematikus és a villámkérdéses szakasz lejátszására is korlátozott idő áll rendelkezésre. A csapatok által az egyes körökben megszerzett pontszám összeadódik, ez képzi majd az összpontszámot.

Az egyes körök végeztével kezdődik a sorban utánuk következő, egészen addig, amíg az összes kört le ne játszották a résztvevők. Ekkor következik a döntő kör, melynek szabályai megegyezni a korábbiakkal. A döntő résztvevői az első 4 legmagasabb összesített pontszámmal rendelkező csapat, akik a döntőben mind 0 pontról indulnak. A döntő győztese nyeri a játékot.

**A jelenleg használt segédprogram**

A játék során használt kérdéssorok megjelenítéséhez, a csapatok pontszámainak kijelzéséhez, és az idő méréséhez jelenleg egy általam ismeretlen személy által, a 2000-es években készített program van használatban. A programot egy ember kezeli, aki egyrészt figyeli a játékosok által kért kérdést, és mindig azt jeleníti meg a képernyőn, másrészt figyeli hogy a játékmester elfogadta-e a választ, és ez alapján adja meg/vonja le a pontokat a csapatoktól.

Az általam megvalósított Mindent vagy semmit program ennek a jól bevált működését veszi alapul, és ezen program hibáiból tanulva kísérel meg az elődjénél jobb felhasználói élményt nyújtani (A részleteket lásd a későbbi fejezetekben.)

# Felhasználói dokumentáció

# Fejlesztői dokumentáció

## Főbb tervezési irányvonalak

Középiskolás koromtól kezdve, amikor barátaimmal még heti szinten aktívan használtam a kvízes rendszert (kiváltképp a programot), egészen mostanáig megvan bennem a szándék, hogy legyen egy olyan alkalmazás, ami jobban megfelel az eredeti program készítése óta eltelt évek támasztotta igényeknek. Nekem is volt sok ötletem ezzel kapcsolatban, de hogy a tényleges felhasználói igényeket felmérve tudjak nekiállni a tervezésnek, egy ehhez kapcsolódó kérdőív szétküldésével gyűjtöttem visszajelzéseket. Nem érkezett be annyi válasz, amennyire számítottam, de úgy vélem a felmérés, még így is reprezentatív.

Mivel a készítendő alkalmazásnál a funkciót tartottam elsődleges fontosságúnak, ezért az 1. és 2. ábrán látható visszajelzések fontos útmutatásként szolgáltak, hogy milyen mélységben kell „hozzányúlnom”, vagyis módosítanom, esetleg kivennem, az eddig használt funkciókhoz.



1. ábra A régi program funkcionalitására irányuló kérdésre kapott válaszok aránya

A képen szöveg látható

Automatikusan generált leírás

2. ábra A régi program esetleges haszontalan funkcióiról érdeklődő kérdés, és az erre kapott válaszok.

A meglévő funkciók átdolgozására, és újak hozzáadására igen sok javaslat érkezett, amelyek együttes megvalósítása (néhány általam kitalált funkcióval együtt) túlmutat a szakdolgozat keretein, ám többségük a jövőben vélhetően megvalósításra kerül. (Lásd: További fejlesztési lehetőségek c. fejezet.)

A funkció után a megjelenés volt számomra a következő prioritás. Véleményem szerint lehet egy alkalmazás bármennyire fejlett, és funkcionálisan tökéletes, ha nem frissítik a megjelenését, az évenkét változó UI/UX igények és trendek fokozatosan elavulttá teszik a felhasználók szemében. Bár a meglévő program elég egyedi a maga műfajában, de korábbi állításomat a 3. ábrán látható eredmény igazolni látszik.



3. ábra A régi program kinézetét érintő kérdés

Míg az előző két szempont talán felhasználóként a leglényegesebb, addig fejlesztőként a már a témabejelentőben is szereplő bővíthetőség kritikusan fontos. Mivel az alkalmazást a jövőben több területen tervezem továbbfejleszteni, ezért értékesebbnek tartok egy átláthatóan, és koherensen megírtat, egy extra funkciókkal telerakott és/vagy leg elkápráztatóbb UI megoldásokat használó, de kevésbé fejlesztő barát alkalmazásnál.

Összegezve: A fejlesztés során a prioritások sorrendben a következőként alakulnak:

1. Bővíthetőség, átláthatóság
2. Funkcionalitás
3. Kinézet

## Architekturális döntések

Az alkalmazás egy asztali alkalmazás. 2022-ben valóban szokatlan lehet egy dedikáltan asztali alkalmazást készíteni, ám ebben az esetben vannak olyan körülmények, amik ezt indokolttá teszik. A közösségünkben meglévő régi program (aminek az utódjaként ez készült) igénybevételeire, jelentős hányadban olyan környezetben kerül sor, ahol nincs, vagy csak akadozó internetkapcsolat áll rendelkezésre. Ezen okból kizárható a kizárólag online működtetett alkalmazás. A fennmaradó offline platformok közül a PC-s megvalósításon kívül minden más irreleváns lett volna.

A következő fontos döntés, a környezet kiválasztása volt. Mivel Windowson kívül más operációs rendszerre megvalósítani nem lett volna értelme, ezért kézenfekvő választás volt a .NET keretrendszer egy Windows specifikus szegmense.

A választott környezet eleinte az UWP[[3]](#footnote-3) volt, de idő közben a Microsoft kiadta az új MAUI[[4]](#footnote-4) rendszerbe illeszkedő WinUI 3[[5]](#footnote-5) (és ezzel együtt a Windows App SDK[[6]](#footnote-6)) első stabil verzióját. Ez az újszerű platform a korábban lefektetett szempontok közül a funkcionális helyességet nem befolyásolja, de a másik kettőben komoly előrelépés figelhető meg az UWP-hez képest. Általában elmondható, hogy egy újabb rendszer lévén jobban teljesíti az egyszerű bővíthetőséggel és fenntarthatósággal szemben támasztott követelményeinket, ám ezt csak fokozza, hogy a Microsoft megszűntette az UWP platform támogatottságát, a Windows App SDK javára, valamint hogy a Windows 11 operációs rendszer későbbi verzióiban várhatóan nem fognak futni az UWP-ra írt alkalmazások. A WinUI 3 tovább bővíti majd az előd amúgysem szegényes eszköztárát a felhasználói felület megvalósításához.

Ezek alapján az alkalmazás a következők szerint került megvalósításra:

* Windows App SDK-t és WinUI 3-at használ
* Szerkezete a platformhoz illő MVVM architektúrát követ
* A model és view-model részek C# nyelven, a view pedig XAML leíró nyelven kerültek megvalósításra
* Az alkalmazás .NET 5 és C# 9 szabványokat támogat

Az alkalmazás a Visual Studio 2019 fejlesztői környezeteben készült, és a további fejlesztői tevékenység zökkenőmentes folytatásához is ez javasolt. Régebbi és újabb verziók használata az esetlegesen fellépő átmeneti problémák miatt lassíthatja a fejlesztési folyamatot.

A projekt létrehozásakor a Visual Studioba kiegészítőként telepíthető Windows Template Studio-t vettem igénybe, amely néhány alapvető funkcióhoz szükséges sablonnal látta el az újonnan létrehozott alkalmazást. (Mivel a projektet eredetileg UWP projektként hoztam létre és csak később döntöttem a migrálásáról, ezért az akkor készült sablonok sem feltétlenül egyeznek meg egy újonnan generálttal.)

## Felhasznált programkönyvtárak, függőségek

## Indulás, aktiváció

Az alkalmazás belépési pontja a környzet által generált App.g.cs fájlban található, ahol az App osztály példányosításával át is tértünk a fejlesztői kódra.

## Navigáció

A program logikailag elkülöníthető funkciói az alkalmazáson belül külön oldalon kerülnek megjelenítésre. A következő oldalak léteznek jelenleg:

* főmenü
* játék beállítások
* Játék
* eredmények
* kérdéssorok-kezelő
* játékos kezelő
* alkalmazás szintű beállítások
* statisztikák (lsd.: További fejlesztési lehetőségek c. fejezet)

Ezen oldalak megjelenítése, és az oldalak közötti váltás, egy Frame, és egy navigációs szolgáltatás segítségével történik.

### Shell

A megjelenítés alapját a ShellPage és a hozzá tartozó ShellViewModel képzi. A SehellPage a Microsoft.UI.Xaml.Controls névtérben megtalálható Page osztályból származik, és funkcióit xaml kódon keresztül nyeri el. Az oldal tartalmát egyetlen, szintén a fenti névtérben megtalálható NavigationView objektum teszi ki. Ennek a MenuItems gyűjteményébe vehetjük fel azokat az elemeket, amiket a menüben szeretnénk látni, a megjelenítendő oldalt pedig a NavigationView Content property-jének értékül adott Frame objektum fogja a képernyőre renderelni. A későbbiekben ha a program oldalt vált (akár a navigációs menüre történő kattintás hatására, akár más okból), akkor mindig a ShellPage belsejében található Frame által megjelenítendő oldalt változtatjuk.

A navigáció másik két meghatározó része a NavigationService és a NavigationViewService.

### PageService

Az oldalak egyszerű betölthetősége érdekében kulcs-érték párokként összekötjük az egyes oldalakhoz tartozó ViewModel-leket, és az oldalakat. A későbbiek során így már elég lesz csak a betöltendő oldal ViewModel-jére hivatkozni, és ebből tudható lesz, hogy melyik oldalt akarjuk betölteni. Fontos, hogy ezen logika miatt viszont egy az egyhez reláció él az oldalak, és a hozzájuk tartozó ViewModellek között. Az előbb leírt folyamat a PageService osztályban történik.

### NavigationViewService

Ez a szolgáltatás a NavigationView által tartalmazott MenuItem-ek felé húzott absztrakciós réteg, ami segíti azok használatát. Referenciaként megkapja a ShellPage-en található NavigationView objektumot, majd a metódusaival ezt manipulálja. Legfontosabb funkciói, az új elem felvétele a navigációs menübe, valamint az egyes menüpontokra történő kattintás kezelése.

Az új elem ellenőrzött hozzáadásához tartozik, hogy ne lehessen többször ugyanazt a navigációs elemet a menühüz adni, valamint, hogy ne legyen a menüben két olyan elem, ami ugyanarra az oldalra vezet. Ez utóbbi ellenőrzéséhez szükséges a privát mezőként tárolt PageService példány.

A NavHelper osztály segítséget nyújt, hogy be tudjuk állítani az egyes menüelemekhez, hogy melyik oldalra vezessenek. A NavigationViewService osztály feliratkozik a NavigationView ItemInvoked eseményére, mely az egyes menü elemekre történő kattintás hatására kerül elküldésre. Az esemény fogadásakor a privát referenciaként tárolt NavigationService példány NavigateTo metódusának meghívásávlakezdjük meg a tényleges navigációt.

### NavigationService

Az osztály paraméterként kapja meg a ShellPage-en szereplő Frame-et, majd ezen objektum metódusai köré alakít ki biztonságos metódusokat, amik hívása biztosan elvárt eredményt ad. Legfontosabb feladata, a Frame NavigateTo metódusának meghívása, mely a saját azonos nevű metódusában történik meg. Mivel a Frame a céloldal típusát kéri, ezen metódus pedig a céloldal ViewModel-jének nevét kapja meg, ezért szükség van a privát PageService objektumra, hogy lekérjük a ViewModel-hez tartozó típust.

## Az Adatelérési réteg

Adatelérési rétegről csak a Mindent vagy semmit programnál beszélhetünk, ebben az alfejezetben erről lesz szó.

# Összegzés

# További fejlesztési lehetőségek

# Hivatkozások

Vollai, G. (2019. május 15). *Retro TV #1 – Mindent vagy Semmit!* Forrás: ubulvilaga.com: https://ubulvilaga.com/2019/05/15/retro-tv-1-mindent-vagy-semmit/

1. Ez az általam elvégzett középiskolában található, és nem az én tulajdonomat képezi. [↑](#footnote-ref-1)
2. Ez lehet bármilyen tetszőleges szabály, akár véletlen szám generálással is történhet. [↑](#footnote-ref-2)
3. https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide [↑](#footnote-ref-3)
4. https://docs.microsoft.com/en-us/dotnet/maui/what-is-maui [↑](#footnote-ref-4)
5. https://docs.microsoft.com/en-us/windows/apps/winui/winui3/ [↑](#footnote-ref-5)
6. Korábban Project Reunion (bővebben: https://docs.microsoft.com/en-us/windows/apps/windows-app-sdk/) [↑](#footnote-ref-6)