

# Követelményspecifikáció

Wikipedia-követő alkalmazás fejlesztése desktopra

Szoftverarchitektúrák házi feladat

## Feladatkiírás

A feladat egy olyan alkalmazás létrehozása, melynek segítségével a felhasználók feliratkozhatnak egy-egy wikipédia szócikkre és email értesítést kaphatnak, ha a cikk tartalma megváltozik. Állítható paraméterek: a frissítési gyakoriság és az "érzékenység" (mekkora módosítás számít relevánsnak). Egy felhasználó legfeljebb 5 szóra iratkozhat fel és legfeljebb napi egy frissítést kérhetnek, de a rendszergazda-jogosultságú felhasználók megnövelhetik ezeket a limiteket egy-egy felhasználó esetében.

## A fejlesztői csapat

Csapattag neve	Neptun-kód	Email-cím
Kinyik Bence	WXI48Y	kinyik94@gmail.com
Tóth Lajos Gábor	QVMZHU	tothlajosg@gmail.com

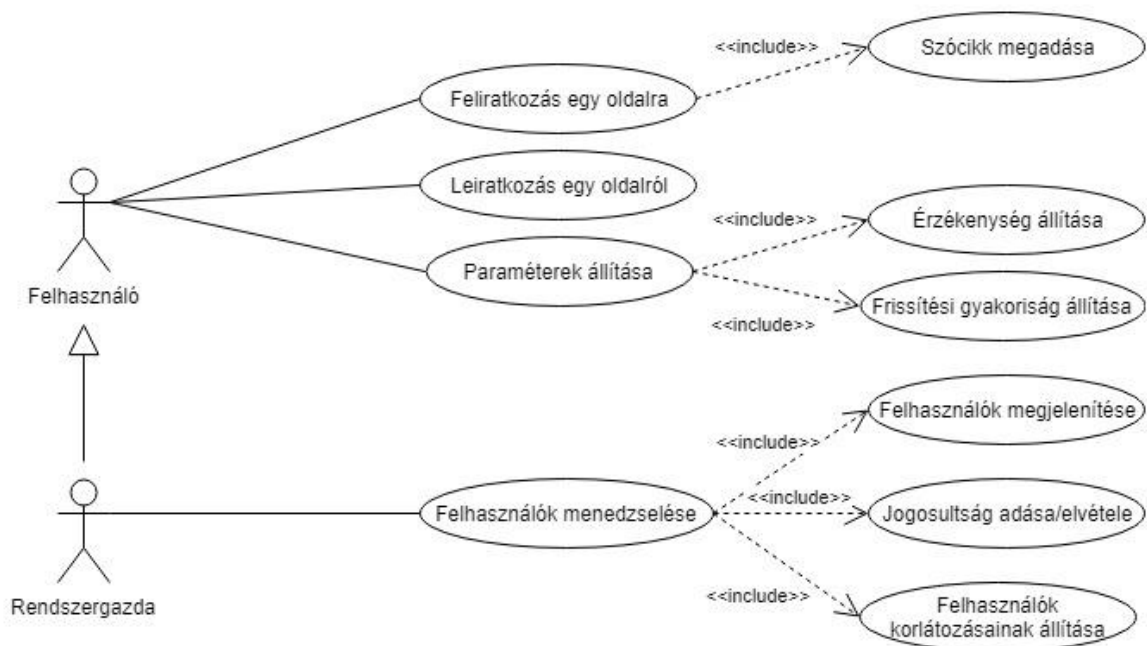
## Részletes feladatléírás

A projekt során célunk egy olyan alkalmazás készítése, amely képes egy felhasználót e-mail-ben értesíteni, amennyiben egy olyan wikipedia szócikk megváltozik, amelyre előzetesen feliratkozott. A felhasználók jogosultság szerint két csoportba oszthatók: egyszerű felhasználó és rendszergazda. Egy felhasználó bármely jogosultsági csoportba tartozik, képes a frissítési gyakoriságot és egy úgynevezett érzékenységi paramétert beállítani, azonban egy egyszerű felhasználó legfeljebb öt oldalra iratkozhat fel és legfeljebb napi egy frissítést kaphat, míg a rendszergazda-jogosultságú felhasználó módosíthatja is ezeket a limiteket egy már beregisztrált felhasználónál. A frissítési gyakoriság állításával a lekérdezés gyakoriságát, míg az érzékenységgel állításával az állítható be, hogy mekkora módosítás esetén küldjön értesítő e-mailt a program.

A program első lépésben azonosítja a felhasználót. Egyszerű felhasználó esetén lehetőséget ad egy adott oldalra való feliratkozásra egy Wikipedia honlap címének megadásával, a feliratkozott oldalak

megjelenítésére, és leiratkozásra is. A rendszergazda ezen felül képes a regisztrált felhasználók megjelenítésére, egy adott felhasználónál a frissítési gyakoriság és az érzékenység átállítására, valamint egy egyszerű felhasználónak rendszergazdai jogot is adhat. Amennyiben egy olyan Wikipedia szócikk -amire több, mint egy felhasználó feliratkozott- megváltozik, és megváltozása átlépi a „érzékenységi küszöböt”, a program értesítést küld a feliratkozott felhasználó(k)nak arra az e-mail címre, amely a felhasználóhoz tartozik. A program adatbázisa két rendszergazdát (admin1, admin2) és 3 egyszerű felhasználót tartalmaz (user1, user2, user3).

## Use-case diagram



## Technikai paraméterek

Az alkalmazást C# nyelven a .NET keretrendszer segítségével valósítjuk meg Microsoft Visual Studio 2015-ös verziójú fejlesztőkörnyezetet használva, így a futtató számítógépnek előre telepített .NET keretrendszerrel kell rendelkeznie. Az előző ábrán látható use-case-ek Windows Formokon keresztül vihetők véghez és a fő Form lekicsinyítésekor a háttérben fut. Az implementálandó szoftver a futtatáshoz szükséges adatokat, egy helyi adatbázisban tárolja (felhasználónév, e-mail cím, jelszó, jogosultság, feliratkozott oldalak utolsó lekérdezés ideje, stb). A lekérdezésekhez és az oldalak megváltozásának a detektálásához a MediaWiki API-t használjuk, ami egy olyan webszolgáltatás,

amivel különböző lekérések intézhetők a Wikipedia felé (korábbi verziók lekérdezése, egy oldal utolsó módosításának dátuma, stb.).

## Szótár

**Wikipedia oldal:** Egy olyan szócikk, ami a wikipedia.org domain név alá tartozik (például: [https://en.wikipedia.org/wiki/User:Szarch\\_hf](https://en.wikipedia.org/wiki/User:Szarch_hf)).

**Feliratkozás egy oldalra:** Wikipedia szócikk hozzárendelése a feliratkozó felhasználóhoz. Az e-mail értesítések a szócikk megváltozásakor, de a feliratkozáskor meghatározott paraméterek függvényében kerülnek kiküldésre a felhasználóhoz tartozó e-mail címre.

**Leiratkozás egy oldalról:** Feliratkozás megszüntetése, azaz egy felhasználóhoz rendelt wikipedia szócikk megváltozásához rendelt értesítés küldésének eltörlése.

**Érzékenység:** Annak meghatározásán, hogy egy adott wikipedia szócikk hány betűjének megváltozása generál egy eseményt.

**Wikipedia követő**

Szoftverarchitektúrák házi feladat

Kinyik Bence

Tóth Lajos

## Tartalom

A rendszer célja, funkciói és környezete.....	6
Feladatkiírás.....	6
A rendszer által biztosítandó tipikus funkciók.....	6
A program környezete .....	6
Megvalósítás .....	7
Architektúra .....	7
Adatbázis réteg .....	7
Adathozzáférési réteg (Data Access Layer).....	8
Üzleti logika.....	9
Grafikus felhasználói felület.....	9
Osztálydiagram .....	12
Frissítési logika .....	13
Adat- és adatbázis terv.....	13
E-K diagram .....	14
AdatModell .....	14
User entitás .....	15
Page entitás.....	15
Telepítési leírás .....	16
A program készítése során használt eszközök.....	16
Összefoglalás.....	16
Továbbifejlesztési lehetőségek.....	17
Hivatkozások .....	18

## A rendszer célja, funkciói és környezete

### Feladatkiírás

A feladat egy olyan alkalmazás létrehozása, melynek segítségével a felhasználók feliratkozhatnak egy-egy wikipédia szócikkre és email értesítést kaphatnak, ha a cikk tartalma megváltozik. Állítható paraméterek: a frissítési gyakoriság és az "érzékenység" (mekkora módosítás számít relevánsnak). Egy felhasználó legfeljebb 5 szóra iratkozhat fel és legfeljebb napi egy frissítést kérhetnek, de a rendszergazda-jogosultságú felhasználók megnövelhetik ezeket a limiteket egy-egy felhasználó esetében.

### A rendszer által biztosítandó tipikus funkciók

Vázlatosan az alábbi funkciók biztosítását várjuk el a rendszertől. (A funkciók részletes definíciója szintén a követelményspecifikáció dokumentumban olvasható.)

#### **Egyszerű felhasználó esetében:**

- Feliratkozás egy Wikipedia oldalra
- Leiratkozás egy Wikipedia oldalról
- Paraméterek állítása (érzékenység, frissítési gyakoriság)
- A már feliratkozott oldalak és azok paramétereinek megtekintése

#### **Admin esetében (az egyszerű felhasználó use-case-eit kiegészítve):**

- Regisztrált felhasználók megtekintése
- Regisztrált felhasználók jogainak állítása
- Felhasználók paramétereinek állítása (érzékenység, frissítési gyakoriság)

### A program környezete

A programot teljes egészében C# nyelven .Net keretrendszer segítségével írtuk, így a futtató számítógépnek előre telepített .NET keretrendszerrel kell rendelkeznie, az adatbázisba való beszúráshoz, lekérésekhez pedig egy helyi SQL szervert hoztunk létre, aminek kezeléséhez az Entity Framework[[1]] keretrendszert választottuk.

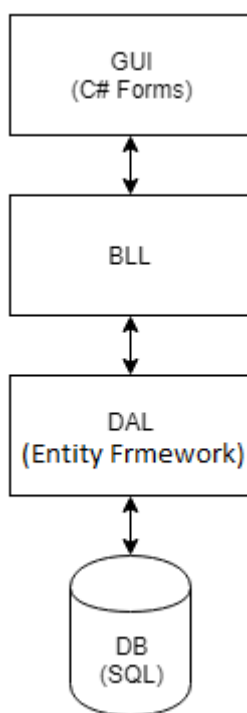
A program egy feltelepített 4.5-ös verziójú .Net keretrendszeren kívül, a manapság használatos számítógépek által nem teljesített követelményt nem támaszt.

## Megvalósítás

Az alkalmazást a feladatkiírásnak megfelelően egy többrétegű alkalmazásként készítettük el.

A fejezetben áttekintést adunk a program architektúrájáról, bemutatjuk az egyes komponensek feladatait és felelősségeit, továbbá részletesen ismertetjük a használt adatmodellt és a grafikus felhasználói felület felépítését.

### Architektúra



1. ábra Rétegek

### Adatbázis réteg

**Célja:** Az adatbázis réteg felel az adatok perzisztálásáért.

Erre mi a Microsoft SQL Server-t választottuk, hiszen egyszerűen létrehozható, módosítható és könnyen mozgatható a fejlesztés alatt (Visual Studio-ban is elérhető), ami a tesztelést nagyban megkönnyíti. Az adatbázisban található adatokat egy lokális mdf kiterjesztésű adatbázis fájlban tároljuk. Emiatt az adatbázis könnyen mozgatható a programmal együtt, így nem szükséges külön alkalmazás telepítése.

Az adatbázis réteg definiálja az egyes sémákat, amelyek leírják az entitások felépítését. Két tábla található az adatbázisban:

- Users

Ez a tábla tartalmazza a felhasználóhoz tartozó információkat: felhasználói név, jelszó, email cím, értesítési szám naponta, feliratkozások maximális száma és azt, hogy admin-e a felhasználó. Ebben a táblában a felhasználói név a kulcs, ezzel egyértelműen lehet azonosítani egy felhasználót.

- Pages

Ez a tábla tartalmazza az egyes feliratkozásokról tárolt információkat. A feliratkozáshoz tartozó felhasználó neve (UserName), wikipédia oldal, érzékenységi, frissítési gyakoriság, hányszor lett értesítve a felhasználó az adott napon, utolsó tárolt revision id, utolsó frissítés ideje és utolsó értesítés ideje. Az egyes feliratkozásokhoz tartozik egy külön autoinkrementális id. A UserName mező egy külső kulcs a Users táblához.

### Adathozzáférési réteg (Data Access Layer)

**Célja:** a külső, objektumrelációs leképezést (ORM) biztosító eszközzel együttműködve adathozzáférés biztosítása a felsőbb rétegek számára.

Ennek megfelelően a réteg feladatai:

- új entitások létrehozása az adatbázisban (az ORM eszköz felhasználásával),
- igény szerinti adathozzáférés biztosítása felsőbb rétegnek az adatbázishoz (az ORM eszköz felhasználásával).

**Megjelenés a kódban:** az adathozzáférési réteg megvalósítása a Model könyvtárban található fájlokban van implementálva.

Az itt található .edmx fájl tartalmazza az információkat az adatbázisról és az adatbázishoz tartozó ORM mappingről. Az egyes táblákhoz tartozó leíró osztályok Page.cs és User.cs fájlokban lettek definiálva. Ezek a fájlok az Entity Framework segítségével lettek létrehozva az adatbázisban található sémák alapján.

Az adatbázis lekérdezések és az adatok manipulálása az Entity Framework-en keresztül, linq szintaxis használatával történtek.

- Új entitás létrehozása és perzisztálása
  - Az alábbi entitások létrehozására van lehetőség a programban:



- feliratkozás egy oldalra (Page)
  - Feliratkozás létrehozásakor a felhasználói név és az oldal nevének megadása kötelező.
- Entitás törlése
  - Az alábbi entitások törlésére van lehetőség:
    - feliratkozás egy oldalra (Page)
  - A törlés a felhasználói év és oldal nevének magadásával történik, mivel ezek egyediek minden feliratkozás esetén
- Entitás tulajdonságainak módosítása
  - Az alábbi entitások törlésére van lehetőség:
    - felhasználó(User)
    - feliratkozás egy oldalra (Page)
  - Felhasználó esetén egy admin felhasználó tudja állítani azt, hogy admin e egy másik felhasználó
  - Feliratkozás esetén állítható az érzékenység és a frissítési gyakoriság.
- Adatok lekérése
  - Az adatelérési réteg ad lehetőséget adatok lekérésére. A lekérések linq segítségével történnek a programban.
  - Az alábbi lekérdezéseket használja a program:
    - Minden felhasználó adatainak lekérdezése a jelszó kivételével (csak admin felhasználó)
    - Adott felhasználóhoz tartozó feliratkozások lekérése
    - A frissítéshez szükséges információk lekérése

## Üzleti logika

**Célja:** az adatelérési rétegtől kapott adatok alapján kiszolgálni a grafikus felhasználói felületet, illetve értesítés küldése az egyes felhasználóknak. Lényegében egy wrapper facade-ként viselkedik.

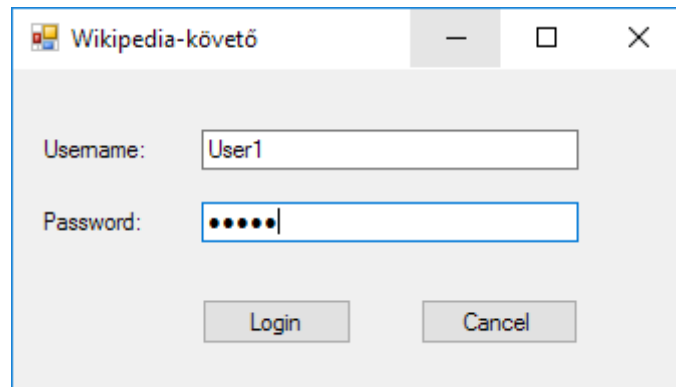
Esetünkben ez a réteg végzi a különböző verziójú wikipedia oldalakon a változások ellenőrzését és adott esetben email értesítést küld a feliratkozott felhasználónak, ezenkívül ebben a rétegben valósítjuk meg a MediaWiki API által szolgáltatott adatok feldolgozását is.

**Megjelenés a kódban:** WikipediaPoller, WikiAPI, EmailSender osztályokban.

## Grafikus felhasználói felület

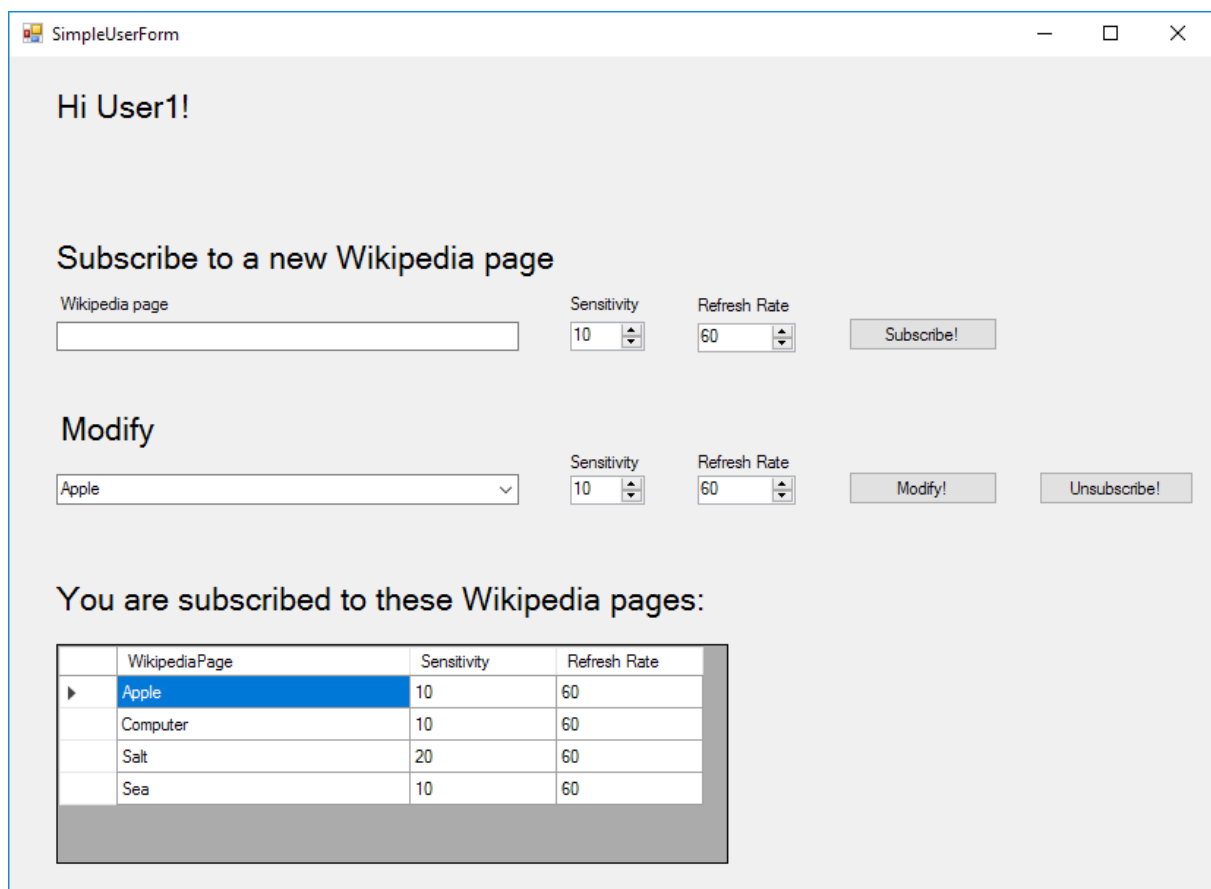
**Célja:** A felhasználók számára egyszerű, könnyen átlátható felületet nyújtani, az összes funkciót elérhetővé tenni.

A Wikipedia-követő felületét C# Formokkal valósítottuk meg. A grafikus felület három ablakot tartalmaz: egy bejelentkező ablakot, az egyszerű felhasználó ablakát és az admin ablakát.



2. ábra Bejelentkező form

A bejelentkező form teszi lehetővé, hogy a regisztrált felhasználók bejelentkezhessenek és jogosultságuktól függően eljussanak a később bemutatott formokra, ahol a követelményekben definiált use-case-eket hajthatják végre. Helyes felhasználónév/jelszó beírásakor a Login gomb megnyomásával térhetnek át a felhasználók a jogosultságuknak megfelelő formokra.



	WikipediaPage	Sensitivity	Refresh Rate
▶	Apple	10	60
	Computer	10	60
	Salt	20	60
	Sea	10	60

3. ábra Egyszerű felhasználó form

Amennyiben egy egyszerű felhasználó jelentkezett be a 3. ábrán látható ablak fogadja. A felső sorban új oldalra tud feliratkozni a megfelelő érzékenység és a frissítési gyakoriság beállításával (amennyiben a feliratkozási limitje nem haladja meg a feliratkozások számát).

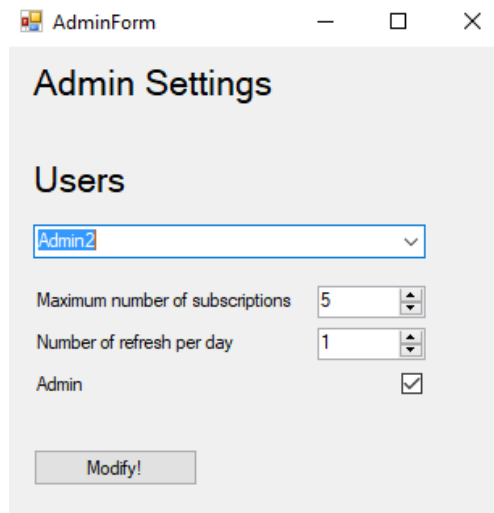
A középső sorban a bejelentkezett felhasználó a kiválasztott oldalhoz tartozó paramétereit tudja módosítani, illetve leiratkozni, amennyiben nem kíván több e-mail értesítést kapni az adott oldal megváltozásáról.

The screenshot shows a window titled "SimpleUserForm" with standard Windows window controls. The main content area has a light gray background. At the top left, it says "Hi Admin1!". At the top right, there is a button labeled "Admin Settings". Below this, the section "Subscribe to a new Wikipedia page" contains a text input field for "Wikipedia page", two spinners for "Sensitivity" (set to 10) and "Refresh Rate" (set to 60), and a "Subscribe!" button. The next section, "Modify", features a dropdown menu, a "Sensitivity" spinner (set to 0), a "Refresh Rate" spinner (set to 10), a "Modify!" button, and an "Unsubscribe!" button. The final section, "You are subscribed to these Wikipedia pages:", contains a table with the following structure:

WikipediaPage	Sensitivity	Refresh Rate

4. ábra User form admin gombbal

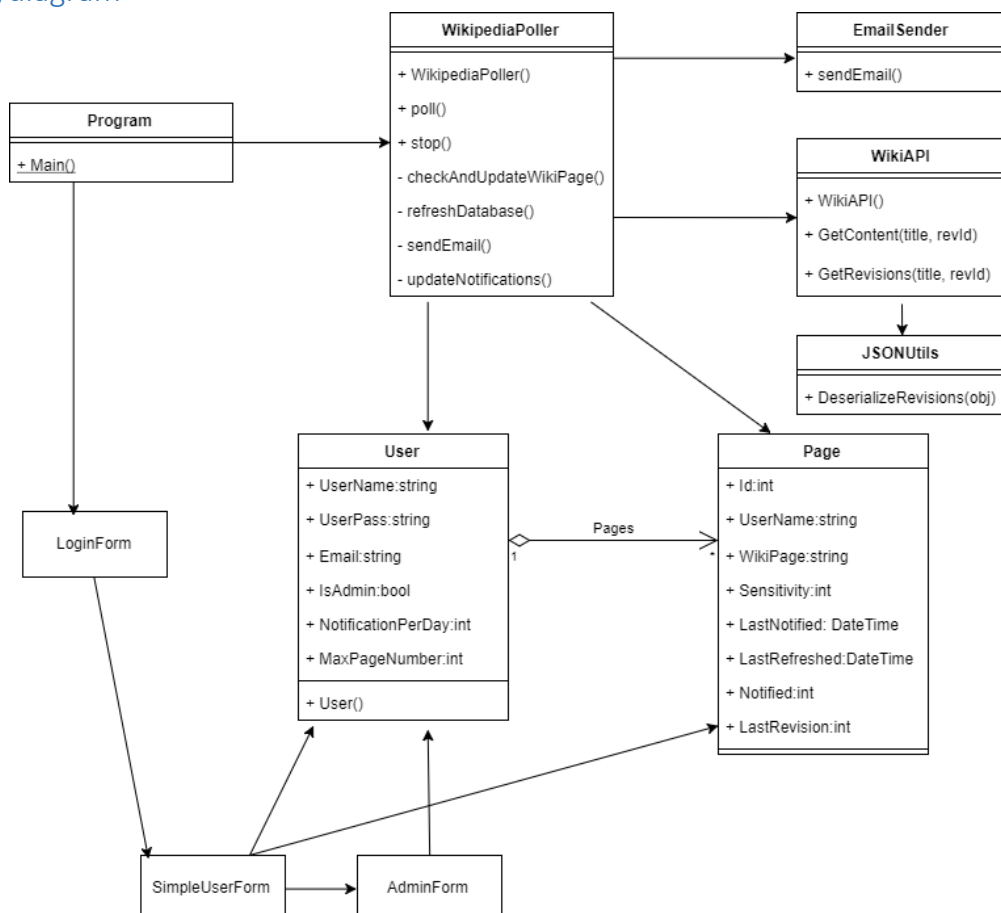
Amennyiben valaki admin jogosultsággal lép be, akkor szintén a felhasználói form fogadja, viszont lesz egy extra gomb, amivel eléri az admin beállításokat(5. ábra).



5. ábra Admin form

Az admin formon egy admin felhasználónak lehetősége van beállítani az egyes felhasználóknál, hogy mennyi oldalra iratkozhatnak fel, mennyi értesítést kaphatnak oldalanként egy nap, illetve be tudja állítani az egyes felhasználóknál, hogy azoknak van-e admin joga.

## Osztálydiagram



6. ábra Osztálydiagram

## Frissítési logika

A frissítési logika akkor kezd el futni, amikor elindítjuk az alkalmazást, és az alkalmazással együtt leáll, tehát, ha nem fut az alkalmazás, akkor nem kap értesítést a felhasználó.

A program indulásakor indul egy poller szál, aminek az a felelőssége, hogy figyelje a változtatásokat és időzítse a frissítéseket. Ez a program futása alatt végig teljesen külön működik a UI szálról, egyetlen közös pont közöttük az adatbázis. Amikor bezárjuk az alkalmazást, akkor erről értesíti a poller szál is, amit ezután bevár, és utána lép ki.

A poller adott időközönként (1 perc) végig nézi az összes Wikipedia oldalt, amire a felhasználók feliratkoztak, és megnézi, hogy szükséges-e frissíteni (lehet-e még értesíteni a felhasználót, letelt-e a frissítési gyakoriságnak megfelelő idő). Amennyiben frissíteni kell, indít egy új szálát, ami először megnézi, hogy van-e új állapota az oldalnak, ha van, akkor pedig összehasonlítja az új állapotot az előző állapottal és adott esetben értesítést küld a felhasználónak az EmailSender osztály segítségével. A ciklus végén bevár minden külön szálát a poller szál, és csak ezután indít új szálát, ha letelt az egy perc.

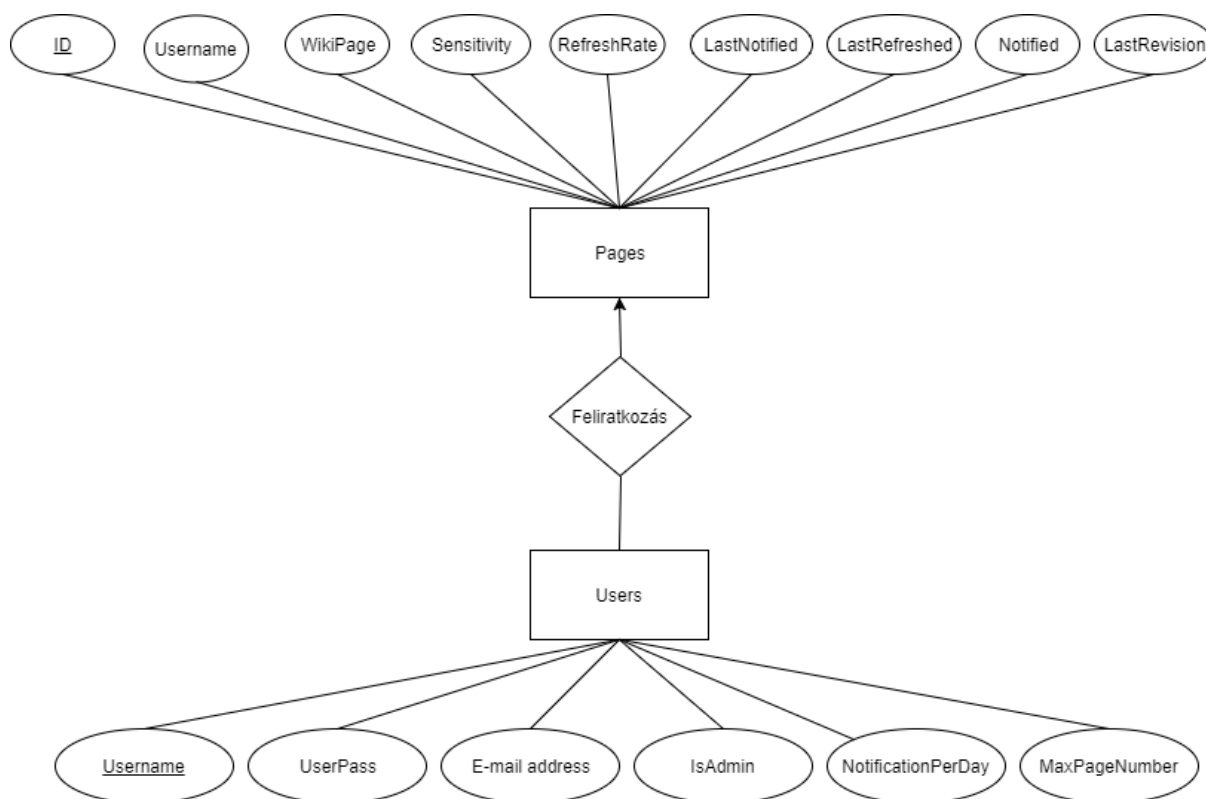
## Adat- és adatbázis terv

Az adatbázis és a hozzá tartozó osztályok elkészítésénél az úgy nevezett „Database First development” [2] stratégiát követtük, azaz először elkészítettük az adatbázist, majd ebből készítettük az Entity Framework segítségével az osztályainkat.

Az Entity Framework minden táblához készít egy osztályt, amelynek a példányai megegyeznek az adatbázis táblájában található entitásokkal. A két tábla között 1:N kapcsolat van, mivel 1 felhasználó több wikipédia oldalra is feliratkozhat. Ezt az 1:N kapcsolatot az Entity Framework úgy oldja fel, hogy az N oldalon tárol egy referenciát a túloldalon lévő entitásra, és az 1 oldalon tárol egy referencia listát a túloldalon lévő elemekről.

## E-K diagram

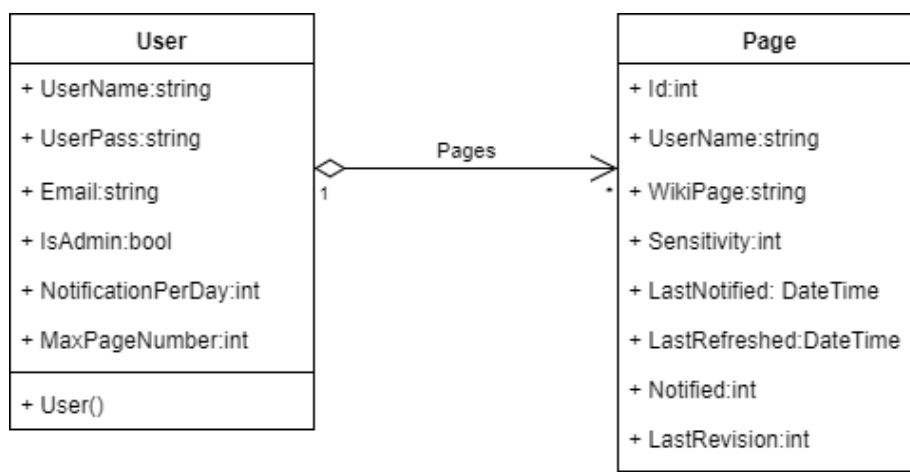
Az alábbi ábrán az egyed-kapcsolat diagram látható:



7. ábra ER diagram

Ennek megfelelően két táblát hoztunk létre az SQL adatbázisunkban: Users és Pages. A Users táblában a felhasználókhöz tartozó adatokat tároljuk a Pages-ben, pedig a feliratkozásokhoz tartozó adatokat. Megjegyzendő, hogy a Users táblához csak a fejlesztők tudnak új sort felvenni.

## AdatModell



8. ábra Adatmodell

### User entitás

**Cél:** Egy felhasználó reprezentálása

**Programban:** User osztály

**Adatbázisban:** Users tábla elemei

**Tulajdonságai:**

Név	C# típus	Adatbázis típus
UserName	string	NVARCHAR (50) PRIMARY KEY
UserPass	string	NVARCHAR (50) NOT NULL
Email	string	NVARCHAR (50) NOT NULL
IsAdmin	bool	BIT DEFAULT ((0)) NOT NULL
NotificationPerDay	int	INT NOT NULL
MaxPageNumber	int	INT DEFAULT ((5)) NOT NULL
Pages	ICollection<Page>	-

- **UserName:** A felhasználónév, egyedi minden felhasználónak
- **UserPass:** Jelszó SH1 hashként tárolva
- **Email:** Felhasználó email címe
- **IsAdmin:** Admin jogosultságú e a felhasználó
- **NotificationPerDay:** Hány értesítést kaphat oldalanként a felhasználó
- **MaxPageNumber:** Hány oldalra iratkozhat fel a felhasználó
- **Pages:** A felhasználóhoz tartozó feliratkozások kollekciója

### Page entitás

**Cél:** Egy feliratkozás adatainak tárolása

**Programban:** Page osztály

**Adatbázisban:** Pages tábla elemei

**Tulajdonságai:**

Név	C# típus	Adatbázis típus
Id	int	INT PRIMARY KEY
UserName	string	NVARCHAR (50) NOT NULL FOREIGN KEY (Users)
WikiPage	string	NVARCHAR (50) NOT NULL
Sensitivity	int	INT DEFAULT ((10)) NOT NULL
RefreshRate	int	INT DEFAULT ((60)) NOT NULL
LastNotified	DateTime	DATETIME
LastRefreshed	DateTime	DATETIME
Notified	int	INT DEFAULT ((0)) NOT NULL
LastRevision	int	INT DEFAULT ((-1)) NOT NULL
User	User	-

- **Id:** Egyedi azonosítója a feliratkozásnak
- **UserName:** Annak a felhasználónak a neve, aki feliratkozott
- **WikiPage:** Az szócikk neve, amire feliratkozott a felhasználó
- **Sensitivity:** Hány szó változása felett értesíti a program a felhasználót
- **RefreshRate:** Hány percenként frissítse az adatokat az adott feliratkozáshoz, értéke legalább 10
- **Notified:** Hányszor volt az adott nap értesítve a felhasználó egy oldal megváltozásáról
- **LastRevision:** Utoljára értesített állapot azonosítója
- **User:** A feliratkozáshoz tartozó felhasználó

## Telepítési leírás

A program telepítése annyiból áll, hogy a közétett install könyvtárban található .zip fájlt kicsomagoljuk tetszőleges helyre. Ez tartalmaz minden szükséges elemet a program futásához, beleértve a futtatható wikipedia\_koveto.exe fájlt és a példa adatokkal feltöltött .mdf adatbázis fájlt. Ezután bármely .Net keretrendszerrel(legalább 4.5) rendelkező Windows operációs rendszeren futtatható.

## A program készítése során használt eszközök

- Microsoft Visual Studio 2015
  - Fejlesztőkörnyezet
- Microsoft SQL Server
  - Adatbázis-kezelő rendszer
- Microsoft Word
  - Dokumentáció írása
- GitHub[3]
  - Verziókezelés, csapatmunka támogatására
- draw.io weboldal[4]
  - Ábrák készítése

## Összefoglalás

Munkánk során megterveztük, implementáltuk illetve dokumentáltuk a Wikipedia-követő rendszert. Az elkészített alkalmazással Wikipedia oldalakra tudunk feliratkozni és a megfelelő mértékű változás esetén e-mail értesítést küldeni a felhasználóknak.



A megvalósított alkalmazás négyrétegű architektúrát használ: adatbázis réteg, adatelérési réteg, üzleti logikai réteg és felhasználói felület. Az alkalmazás az adatokat egy helyi sql fájlban tárolja, ezt éri el az adatelérési réteg. A legnagyobb kihívást az üzleti logikai réteg implementálása volt, mivel erre a feladatra nem voltak előre implementált megoldások. A GUI-t C# formokkal oldottuk meg.

## Továbbfejlesztési lehetőségek

A wikipedia követő jelen állapotában egy jól működő, a specifikációt teljesítő program, azonban számos továbbfejlesztési lehetőség rejlik benne:

- Új felhasználók regisztrálása
- A program (adatbázis) távoli elérése és használata, mint szerveralkalmazás
- Frissítési logika optimalizációja

## Hivatkozások

- [1] Entity Framework: [https://msdn.microsoft.com/en-us/library/gg696172\(v=vs.103\).aspx](https://msdn.microsoft.com/en-us/library/gg696172(v=vs.103).aspx)
- [2] Database First development: [https://msdn.microsoft.com/en-us/library/jj206878\(v=vs.113\).aspx](https://msdn.microsoft.com/en-us/library/jj206878(v=vs.113).aspx)
- [3] GitHub: <https://github.com/>
- [4] draw.io: <https://www.draw.io/>