|  |
| --- |
| Ruander Oktatási Kft. |
| DiceTravel |
| Szakdolgozat |

|  |
| --- |
| Molnár Patrik  2020.09.19. |

DiceTravel szakdolgozat

# 1. Témamegjelölés és indolkás

Szakdolgozatom témájául egy utazás naplózó alkalmazást választottam, mert szeretek túrázni, kirándulni, szeretem felfedezni a világot mely, körülvesz bennünket. Sokszor gondoltam arra, hogy milyen jó lenne, ha csak pár kattintás segítségével megörökíthetnék egy eseményt, sőt adott esetben még képet is mellékelhetnék hozzá.

Rengeteg applikáció található a piacon az utazások megszervezésére, megtervezésére legyen szó itt a repülőjegyvételről, szobafoglalásról, útvonaltervezésről. Én azonban egy olyan alkalmazást szeretnék készíteni, ami lehetőséget nyújt egyfajta útinaplóként arra, hogy megőrizzük az emlékeinket, valamint, hogy ezeket könnyen áttekintsük vagy megosszuk másokkal is.

Bonyolultnak tartottam, hogy mappákba szervezzem a képeket, hogy az utazás végeztével még esetleg nekiálljak rendszerezni ezeket. Ezzel az alkalmazással többet nem lenne erre szükség, mert az emlékek automatikusan logikailag könnyen értelmezhető rendszerben lesznek foglalva. Nincs más dolgunk csak felvenni egy új eseményt és az máris a helyére kerül.

Másik nagy vágyam az volt, hogy egy olyan alkalmazást készítsek, melynek segítségével véletlenszerű megállókat készíthetek, ezzel is izgalmasabbá, kalandosabbá téve az utazásaimat. Úgy gondoltam ez a szakdolgozat remek alapjául szolgálhat egy ilyen alkalmazásnak, melyet sajnos már csak később fogok tudni elkészíteni a mű terjedelmére való tekintettel. Az alkalmazás neve is erre a funkcióra utal.

Úgy gondoltam a tervezés alatt, hogy elsőnek egy asztali változatát készítem el az alkalmazásnak, mert így olyan környezetben mozoghatok, melyet a tanulmányaim alatt megismertem, ehhez hozzátartozik dotNet keretrendszer, a C# és MySQL nyelv egyaránt.

# 2. DiceTravel Home felhasználói kézikönyv

## Bevezető

Köszönjük, hogy utazásnaplózó alkalmazásunk asztali verziójának felhasználói között tudhatunk, és reméljük, hogy örömödet fogod lelni a használatában!

Mielőtt elkezdeném taglalni, hogy miképpen használhatod a DiceTravelt, szeretném pár szóban összegezni, hogy mire is való ez a kis program:

### Utazásnaplózás

Utazások indítása, célállomások felvétele, bejegyzések készítése, melyek akár valós időben is bővíthetőek. Utazásaink, túráink emlékeinek a rendszerezése és megőrzése. A DiceTravel segítségével könnyen újra átélheted a kalandokat.

### Közösségi oldal

Egy platform melyen megoszthatjuk barátainkkal és a nagyvilággal az utazásainkat és az azokhoz tartozó élményeinket. A DiceTravel lehetőséget ad barátok felvételére, valamint az élmények megosztására. Beállíthatjuk, hogy bejegyzéseink kik számára legyenek láthatóak.

### Utazási meghatározott vagy véletlenszerű állomásokra.

Új utazás indításakor, vagy célállomás hozzáadásakor akár magad is meghatározhatod, hogy hová szeretnél eljutni, viszont lehetőséget biztosítunk arra, hogy véletlenszerű településre utazhass Magyarországon belül.

## Mi a DiceTravel?

### Kulcsszavak

Szeretném definiálni a legfontosabb DiceTravelös kulcsszavakat, melyekkel gyakran lehet találkozni az alkalmazásban, valamint a kézikönyvben.

* **Utazás (Journey):** A DiceTravel utazással kapcsolatos legnagyobb logikai egysége, mely alatt a Szakaszok találhatóak. Egy kezdőponttal rendelkezik, valamint egy elnevezéssel.
* **Szakasz (Trip):** Az utazás elemei. Minden szakasz egy végponttal rendelkezik. A kezdőpont az előző szakasz végpontja határozza meg, vagy ha még nincs teljesített szakasz, akkor az Utazás kezdőpontja lesz.
* **Bejegyzés (Entry):** A DiceTravel utazással kapcsolatos legkisebb logikai egysége. Nevezhetnénk ezeket emlékeknek is, ezek adnak az egész alkalmazásnak életet. Minden Bejegyzés egy szöveges bejegyzésbő, ami maximum 1024 karakter hosszú lehet, és/vagy egy képből áll. Utóbbi maximum 16 MB méretű lehet.
* **Folyam (Flow):** A különböző utazással/barátokkal kapcsolatos szűrések nézetei. Egy felhasználó Utazás folyama például az összes általa készített Utazást listázza ki és jeleníti meg. A következő folyamok vannak például a DiceTravelben: Utazás, Történet, Aktuális, Barát.
* **Barát (Friend):** Olyan felhasználó, akivel láthatjuk egymás csak barátoknak szánt bejegyzésit.

A DiceTravel alapvetően egy naplózó alkalmazás. Lényege, hogy egy utazást szakaszokra lehet bontani, mely szakaszok alatt különböző események történhetnek a felhasználókkal. Persze itt nem kell feltétlen nagy dologra gondolni: Látunk egy szép tájat, éppen állunk a dugóban, a csemeténk kimondja első szavát a Balaton parton vagy egy nagyvad épp szépművészeti alkotássá alakította a motorháztetőnket.

Bármi is történjen, csak felveszünk egy új bejegyzést; ha akarunk, csatolunk egy képet és máris bekerült a rendszerbe az esemény! Ezt akár megtarthatjuk privát bejegyzésnek, de meg is oszthatjuk barátainkkal vagy akár a nagyvilággal.

Minden út megkezdése előtt indítanunk kell egy új utazást, ami a mi aktuális utazásunk lesz. Az utazás indításával rögtön meg is kell adnunk, hogy mi az első célja az utazásunknak, valamint hogy honnan indulunk. Ez a két pont fogja meghatározni a Szakaszunkat. Minden bejegyzés ez alá a Szakasz alá fog kerülni, amíg be nem fejezzük a Szakaszt, valamint minden Szakasz ugyan abba az Utazásba fog kerülni, amíg be nem fejezzük az Utazásunkat.

Ez a bekezdés lehet kicsit száraz lett, ezért megpróbálom egy példával elmagyarázni a működést:

*Nem tudom mit kezdjek magammal Augusztus 20-án, gondolkodom és arra jutok jó lenne kimozdulni. Hosszúhétvége. Nyár. Uborkaszezon. Ki is találom, hogy elmegyek Hollókőre, másnap Egerbe, harmadnap pedig visszajövök Budapestre.*

*Az indulás napján bejelentkezek a DiceTravelbe, és új utazást indítok. A rendszer megkérdezi, hogy honnan indulok és merre tartok. Meg is adom neki, előbbi Budapest, utóbbi pedig a terveim szerint Hollókő. A nagy öröm miatt rögtön készítek is egy bejegyzést, hogy eljött a nagy nap, mellécsapok egy boldog szelfit, beállítom, hogy a bejegyzést csak a barátaim láthatják, majd fel is töltöm a bejegyzést.*

Ezek után bármi is történjék velem útközben, arról készíthetek egy bejegyzést, ami az aktív Szakasz alá fog kerülni időrendi sorrendben.

*Pár óra elteltével megérkezem Hollókőre, szétnézek a várban, végig sétálok a faluban, ebédelek egyet, meglátogatom a különböző kézműves standokat.* Ezek mind lehetséges bejegyzések lehetnek, vagy bármi, amit mi fontosnak tartunk.

*Este elfogyasztom a vacsorámat, majd a szállásomon álomra hajtom a fejem. Másnap miután felébredek, megreggelizem, majd újra útra kelek. A DiceTravel alkalmazásban befejezem a Szakaszomat, hiszen véget értek a hollókői kalandjaim, majd egy újat indítok, aminek végállomása immár Eger városa. Ez lesz az utazásom második szakasza. (Hollókő-Eger)*

Innentől újabb bejegyzéseket készíthetünk, ám ezek már az új Szakasz alá lesznek helyezve. Azt, hogy a szakaszainkat milyen logika alapján bontjuk, teljesen ránk van bízva. Rendezhetjük őket naptári nap szerint, települések szerint, vagy bármilyen más elv alapján.

Fontos megjegyezni, hogy abban az esetben viszont, ha a DiceTravellel véletlenszerű utazást indítunk, akkor a szakaszok mindig települések között lesznek.

Mikor először használjuk az alkalmazást még nem lesznek bejegyzéseink. Nem lesznek Utazásaink, sem pedig Szakaszok, amik ezekhez tartoznak. Ahogy egyre több utazást teljesítünk úgy fog bővülni az emlékeink tárháza, viszont mi értelme lenne az egésznek, ha ezt nem oszthatnánk meg ismertőseinkkel.

Ha van olyan ismerősünk, aki szintén regisztrált a DiceTravelre, akkor fel tudjuk venni barátnak. Ha a keresőben sikerül megtalálnunk a felhasználó neve alapján, akkor barátnak jelölhetjük. Ilyenkor még nem lesz elfogadva a barátkérelem, előbb ezt neki is meg kell erősítenie. Onnantól kezdve, hogy ezt megteszi, a barátok számára látható tartalmakat is megtekinthetjük egymás folyamaiban.

A következő pár fejezeteben ismertetjük az alkalmazással kapcsolatos tudnivalókat.

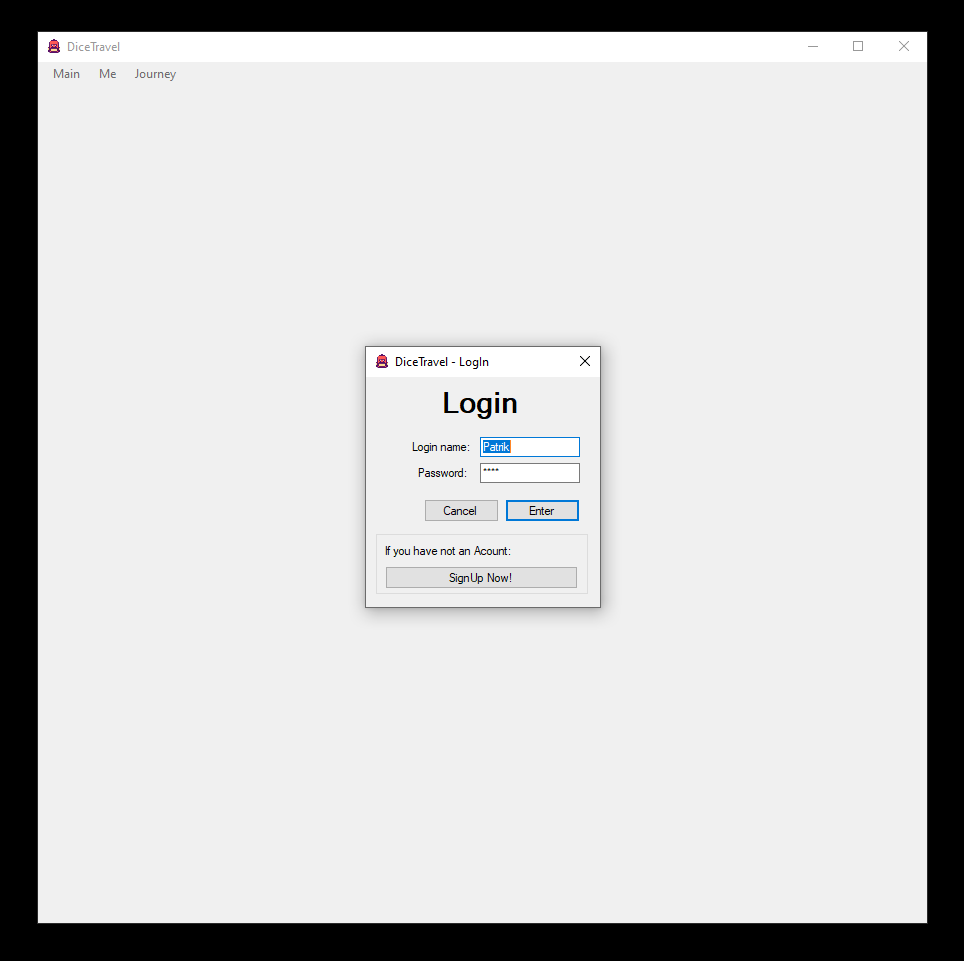
## A DiceTravel Home telepítése

Első lépésként el kell indítani az alkalmazás telepítőjét. A varázsló segítségével pedig végig kell vinni a telepítési folyamatot.

Az alkalmazás alapértelmezetten a „C:\Program Files (x86)\Pamasoft\DiceTravel\” mappába kerül telepítésre.

\*\*\*

## A DiceTravel Home indítása

Miután a program telepítése sikeresen megtörtént, az alkalmazás a telepítési mappában található DiceTravel.exe vagy az asztalra kihelyezett parancsinkon segítségével indítható. A program sikeres indítása után a következő felület jelenik meg:

## A DiceTravel Home felépítése

Áttekintő jelleggel mennyünk végig a program különböző felületein, ahol elmondjuk, hogy mely elem mire való.

### Bejelentkezés ablak

A felületen kettő beviteli mező és három gomb található. A bejelentkezéshez a megfelelő mezőkbe megfelelő értéket kell megadni, majd az Enter gombra kattintva léphetünk be a rendszerbe, ekkor megnyílik a számunkra a főablak. Ha nincs még fiókunk, akkor a *SignUp Now!* gombra kattintva megnyílik a *Regisztrációs* felület. Ha esetleg más menüpontot szeretnénk választani, ami bejelentkezés nélkül is elérhető, akkor kattintsunk a *Cancel* gombra. Ilyenkor elérhetővé válik a DiceTravel főablaka.

### Regisztrációs ablak

A regisztrációs felületen kell megadnunk a kívánt felhasználónevet (*Login name*), jelszót (*Password*), a valódi nevünket (*First* *name*, *Surname*) és születési dátumunkat (*Date* *of* *Birth*). Ezek közül csak a csillaggal megjelölt mezők kitöltése kötelező. Ha mindent sikeresen megadtunk, akkor a *SignUp* gombra való kattintással megtörténik a regisztráció. A rendszer ezután visszairányít bennünket a Bejelentkezés ablakra, ahol már a frissen regisztrált nevünk is be van írva.

*Fontos itt megjegyezni, hogy a felhasználónév nem lehet foglalt, ellenkező esetben a rendszer figyelmeztet bennünket és nem enged tovább a regisztrációs folyamatban.*

### Főablak

Három fő részre osztható a főablak:

|  |  |
| --- | --- |
| 1. **Menüsor** 2. **Adatáttekintő** **szekciók** 3. **Folyamablak** | **1**  **2**  **3** |

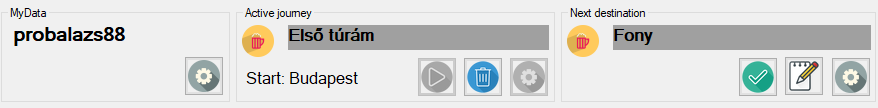
#### Menüsor

Itt találhatóak tematikusan elrendezve a program főbb funkciói. Ebben a fejezetben csak vázlatosan lesz megemlítve, hogy mely funkció mire való, ám a későbbiekben részeletesen kitérünk rájuk.

* **Main:** A ki-, bejelentkezéssel, regisztrációval és kilépessel kapcsolatos funkciók vannak ide helyezve.
  + **Login:** Erre kattintva bejelentkezhetünk a rendszerbe, már ha eddig nem tettük volna meg.
  + **SignUp:** Itt az előbbiekben már bemutatott regisztrációs ablak jelenik meg.
  + **LogOut:** Erre kattintva kijelentkezünk a programból.
  + **Exit:** Bezárja a programot, de előtte kijelentkeztet bennünket a rendszerből.
* **Me:** A saját magunkkal kapcsolatos információkat és folyam nézeteket találjuk ebben a menüben.
  + **Profile:** Megnyitja a személyes adatainkat tartalmazó ablakot, amiben megváltoztathatjuk a rólunk tárolt információkat.
  + **Story:** Erre kattintva a Folyamablak tartalma a saját bejegyzéseinket fogja megjeleníteni időrendi sorrendben.
  + **Journeys:** Ha ezt válasszuk ki akkor a Folyamablakban az összes Utazásunk fog szerepelni időrendi sorrendben.
* **Traveling:** Az utazással kapcsolatos főbb parancsok kerültek itt felsorolásra.
  + **Journey:** Indíthatunk új Journeyt, valamint módosíthatjuk vagy törölhetjük a jelenlegit.
  + **Destination:** Módosíthatjuk vagy befejezhetjük az aktuális szakaszt.
  + **New Entry:** Az aktív Utazás aktuális Szakaszához tudunk itt új bejegyzést írni.
* **Friends:** A barátokkal és más felhasználókkal kapcsolatos alapfunkciók vannak ide helyezve.
  + **My Friends:** Erre a menüpontra kattintva a Folyamablak a barátaink listáját fogja megjeleníteni.
  + **Search User:** Ha ezt válasszuk, akkor felhasználó név alapján kereshetünk a DiceTravel felhasználói között.
  + **Friend Entries:** Erre a menüpontra kattintva a Folyamablakban az elmúlt 24 óra eseményei lesznek kilistázva csökkenő időrendi sorrendben.
* **Help:** Ebben a menüpontban megtekinthetjük a felhasználói kézikönyvet, valamint az alkalmazás alapadatait.

#### Adatáttekintő szekciók

Három adatáttekintő szekció található az alkalmazás felső részén.



* **MyData:** Itt kaptak helyet a bejelentkezett felhasználó adatai. Valamint a fogaskerékre kattintva itt is meg lehet változtatni az adatainkat.
* **Active journey:** Itt látható az éppen aktuálisan *Utazás* (Journey) láthatósága, neve, valamint kiindulási pontja. A lejátszás gombbal lehet új *Utazást* indítani, viszont ha már van futó *Utazás*, akkor ez a gomb inaktív. Az *Utazás* nevére kattintva megnyílik a hozzá tartozó *Történetfolyam*.  
  *Fontos! Egyszerre csak egy Utazás* *lehet aktív!*
* **Next trip:** Minden *Utazás* *Szakaszokból* (Trip) áll, és az éppen soron következő *Szakasz* adatait jeleníti meg ez a szekció. Ezen felül a *Szakaszunkat* itt be tudjuk fejezni, tudunk hozzá új *Bejegyzést* rögzíteni, de akár módosíthatjuk is. A *Szakasz* nevére kattintva megnyílik az alá tartozó *Bejegyzésfolyam*.

#### FolyamablakC:\Users\molnarpatrik\AppData\Local\Microsoft\Windows\INetCache\Content.Word\folyamablak.png

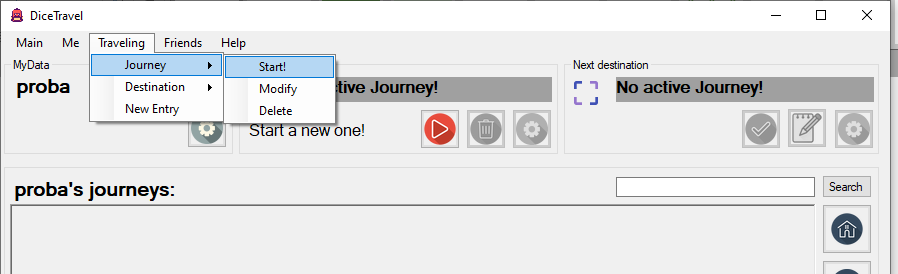
Itt jelennek meg a különböző *Utazások*, *Szakaszok*, *Bejegyzések* részletei, de itt jelennek meg a *Barátaink* vagy más *Felhasználók* adatai is. A jobb oldalán található gombokkal tudunk váltani a különböző folyamok között, felül pedig a *Felhasználókereső* található.

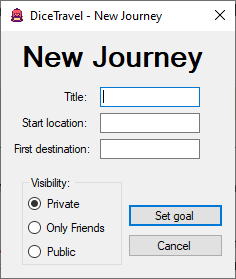
## A DiceTravel Home funkciói

### Új Utazás indítása

Két féle képpen tudunk új utazást indítani. Fontos megjegyezni, hogy ezt csak akkor fogjuk tudni megtenni, ha éppen nincs futó Utazásunk.

1. Rákattintunk a menüsorban a Traveling/Active Journey/Start! menüpontra.
2. Az Active journey adatáttekintő szekció alatt a piros lejátszás gombra kattintunk.



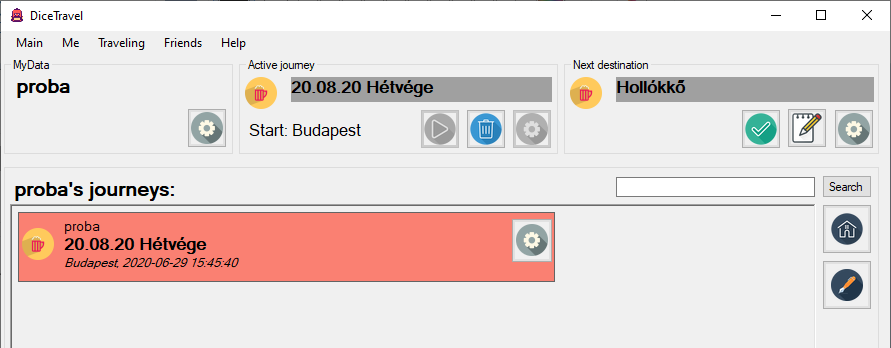
Ennek hatására meg fog jelenni a *New Journey* ablak a következő mezőkkel:

* **Title**: Ez az Utazásunk elnevezése. Bármit adhatunk neki, de érdemes olyat, ami a számunkra könnyen azonosítja.
* **Start location:** Az Utazás kezdőpontja, innen indulunk a kalandunkra.
* **First destination:** Az első Szakaszunk végpontja. Ennek segítségével meg is történik az első szakasz létrehozása.
* **Visibility:** Itt lehet kiválasztani a Journey láthatóságát. Private: csak a létrehozó felhasználó látja, Only Friends: csak a barátok látják, Public: Mindenki látja. Alapértelmezetten Only Friends az értéke, viszont ezt a későbbiekben még megváltoztathatjuk.

**Minden mezőnek értéket kell adni!**

Ha végeztünk, akkor kattintsunk a *Set goal* gombra, hogy létrehozzuk az utazást, ha meggondoltuk magunkat, akkor kattintsunk a *Cancel* gombra. Ezek után a felületen, az *Active* *journey* adatáttekintő szekcióban megjelennek az Utazással kapcsolatos információk, valamint elérhetővé válnak a Törlés és Beállítás gombok.

Ezen felül a *Next* *destination* adatáttekintő szekcióban is megjelenik az aktív szakaszunk célpontja, hogy mennyi bejegyzés lett rögzítve az adott Szakasz alá, valamint aktívvá válnak a *Célpont* *elérve*, *Bejegyzés* *készítése* és *Szakasz* *módosítása* gombok.



Entries: 8

### Aktív Utazás törlése

Két módon is törölhetjük az aktív utazásunkat. Mindkét esetben fontos megjegyezni, hogy az Utazások törlésével **minden** az Utazás alá tartozó **Szakasz** és **Bejegyzés** is **törlésre** **kerül**!

1. Rákattintunk a menüsorban a Traveling/Active Journey/Delete menüpontra.
2. Az Active journey adatáttekintő szekció alatt a kék kuka gombra kattintunk.

Egy megerősítő ablak ugrik fel, mely figyelmeztet bennünket, hogy az Utazás törlésével minden alá tartozó adat is elveszik. Ezt elfogadva törlésre kerül az Utazás.

### Aktív Utazás módosítása

Két módon lehet az aktív utazás adatainak a módosítása:

1. Rákattintunk a menüsorban a Traveling/Active Journey/Modify menüpontra.
2. Az Active journey adatáttekintő szekció alatt a szürke fogaskerék gombra kattintunk.

Egy ’Edit Journey’ ablak ugrik fel, ahol meg tudjuk változtatni az Aktív utazásunk nevét, kezdőpontját valamint láthatóságát. Itt már nem változtatható meg az első Szakasz célpontja, hisz lehetséges, hogy azt már rég teljesítettük. Ha az első célpontot szeretnénk megváltoztatni, akkor azt magánál az első Szakasznál tehetjük meg.

Ha beírtuk a mezőbe az általunk megfelelőnek ítélt értékeket, akkor kattintsunk az Update gombra, és megtörténik az adatok frissítése.

Egy piros ’Delete’ gomb segítségével törölhetjük az utazásunkat.

### Szakasz befejezése

Ha szeretnénk befejezni a szakaszunkat, akkor ezt két féle képpen tehetjük meg:

1. Rákattintunk a menüsorban a Traveling/Destination/Done! menüpontra.
2. A Next destination adatáttekintő szekció alatt a zöld pipa gombra kattintunk.

Egy ’What’s Next?’ ablak jelenik meg, ahol a **Goal location**nel adhatjuk meg a következő cél nevét, valamint a Visibility segítségével állíthatjuk be, hogy az új Szakaszunkat kik láthatják.

**Fontos megjegyezni, hogy a Szakaszok láthatósága nem lehet megengedőbb, mint az Utazásé, amelybe tartoznak!**

Utazás lezárása

Ha lezárjuk az Utazásunkat, akkor többé nem lehet szerkeszteni sem magát az Utazást, sem pedig az alá tartozó Szakaszokat és Bejegyzéseket. Csak és kizárólag a láthatóságot állíthatjuk!

Fontos, hogy az Utazás lezárása előtt még egyszer nézzünk át mindent!

Ha a ’Go Here!’ gombra kattintunk, akkor létrejön a következő Szakaszunk a megadott Célponttal és láthatósággal.

Ha viszont az ’End Journey’-re, akkor lezárjuk az utazásunkat. Ilyenkor nem lesz több Szakasz, az Utazásunk befejezett lesz, valamint nem lesz aktív Utazásunk.

### Szakasz módosítása

Ha szeretnénk módosítani a szakaszunkat, akkor ezt két féle képpen tehetjük meg:

1. Rákattintunk a menüsorban a Traveling/Destination/Modify menüpontra.
2. A Next destination adatáttekintő szekció alatt a szürke fogaskerék gombra kattintunk.

Egy ’Edit Trip’ ablak ugrik fel, ahol meg tudjuk változtatni a Szakaszunk célpontját, valamint, hogy milyen legyen a láthatósága. Az Update gombra kattintva frissítésre kerülnek ezek az értékek.

Egy Szakaszt akkor érdemes megváltoztatni, ha útközben meggondoljuk magunkat, vagy bármilyen akadály jelentkezik, hogy azt a célt elérjük.

### Bejegyzés készítése

Két féle képpen tudunk Bejegyzéseket hozzáadni az aktuális Szakaszunkhoz:

1. Rákattintunk a menüsorban a Traveling/New Entry menüpontra.
2. A Next destination adatáttekintő szekció alatt a jegyzettömb és ceruza gombra kattintunk.

Egy ’New Entry’ című ablak jelenik meg, aminek a segítségével új Bejegyzést rögzíthetünk. A következő beviteli mezők jelennek meg:

* **Title**: Ezzel adjuk meg a bejegyzésünk címét.
* **Story**: Itt írhatjuk le, hogy mi történt velünk, mit láttunk, milyen élményben volt részünk.

A beviteli mezők alatt egy képfeltöltés gomb is található, ahol kiválaszthatunk a számítógépünkről egy tetszőleges képet és csatolhatjuk a bejegyzésünkhöz. Fontos, hogy a kép maximum 14 MB lehet. Mikor kiválasszuk a feltöltendő képet, akkor egy előnézeti kép jelenik meg az ablak jobb oldalán, hogy biztosak legyünk, hogy jót választottunk ki.

A Bejegyzésünknek is be kell állítani egy láthatóságot, ami nem lehet megengedőbb, mint a Szakaszé, amihez tartozik.

A ’Save’ gombra való kattintáskor létrejön az új Bejegyzésünk.

### Keresés

A Folyamablak felett egy kis beviteli mező található, mellette pedig egy ’Search’ gomb. Ha beírunk egy szöveget, majd rákattintunk a gombra, akkor azok a felhasználók fognak kilisítázódni, akik felhasználó nevében szerepel a megadott szöveg. Ez alapján fog elkészülni a Felhasználó-folyam

### Ismerős jelölése/jelölés elfogadása

Ha szeretnénk valakit felvenni a barátlistánkra, ezzel megengedve a számukra, hogy a csak barátok számára elérhető tartalmainkat is láthassák, akkor elsőnek rá kell keresnünk a felhasználó nevére az adott felhasználónak.

Ha sikerült megtalálni, akkor kattintsunk az őt tartalmaző folyamelem melletti pipára. Ezzel el is lett küldve a barátkérelem a számára.

Ha ismerősnek jelöltek bennünket, akkor menjünk a Friends/Invitations menüpontra.

Ilyenkor a Folyamablakban a függő bartájelölések kerülnek megjelenítésre, abban az esetben, ha szeretnénk elfogadni egy kérelmet, egyszerűen kattintsunk a megfelelő Felhasználó-folyamelem melletti pipa gombra.

## A Folyamablak

A következőkben a Folyamablak használatáról és a bennük található elemekről lesz szó. A Folyam különböző elemekből épül fel. Ezek lehetnek: Utazások, Szakaszok, Bejegyzések, Felhasználók, Barátok.

Attól függően, hogy mire vagyunk kíváncsiak, meg tudjuk változtatni, hogy pontosan mi is jelenjen meg a Folyamablakban. Ha például rákeresünk egy felhasználóra, akkor a Folyamban felhasználók fognak megjelenni, de ha a saját Utazásainkra vagyunk kíváncsiak, akkor Utazások.

A következőkben részletezésre kerülnek a különböző Folyam elemek.

### Folyam elemek

Összesen 4 darab elemet különböztetünk meg.

* Utazás
* Szakasz
* Bejegyzés
* Felhasználó

#### Utazás folyamelem

Az utazás folyamelem bal oldalán információkat találunk magáról az utazásról. Itt olvasható az Utazás neve, láthatósága, a létrehozó felhasználó neve, indulási helye és ideje. Az aktív Utazás lazacszínű háttérrel rendelkezik.

A jobb oldalon egy szürke fogaskerék található, mely az adott Utazás módosítására ad lehetőséget, mely az előzőekben ismertetett ’Edit Journey’ ablakra irányít bennünket.

Ha rákattintunk magára az elemre, akkor az adott Utazásfolyam fog megjelenni a Folyamon.

#### Szakasz folyamelem

A szakasz folyamelem bal oldalán információkat találunk magáról az utazásról. Itt olvasható az Utazás neve, melyhez a Szakasz tartozik, maga a Szakasz neve, a létrehozó felhasználó neve, a láthatóság, valamint, hogy mikor lett befejezve az adott Szakasz. Ha a szakasz még folyamatban van, akkor egy ’In progress…’ felirat jelenik meg, valamint lazacszínű háttérszínt kap az elem.

Az elem bal oldalán két gomb található: egy kék visszafelé mutató nyíl, erre kattintva az adott szakaszhoz tartozó Utazásfolyamot nyitjuk meg, valamint egy szürke fogaskerék, mellyel az adott szakaszt tudjuk szerkeszteni. Utóbbira kattintva a már említett ’Edit Trip’ ablak nyílik meg.

Ha magára a Szakasz folyamelemre kattintunk, akkor a Szakaszhoz tartozó Bejegyzés-folyam fog megjelenni a Folyamablakba.

#### bejegyzés folyamelem

A Bejegyzés folyamelem bal oldalán található a bejegyzés címe, láthatósága. Ez alatt magát a bejegyzéshez tartozó történetet olvashatjuk el, valamint ­­­– ha csatoltunk - itt láthatjuk a hozzá tartozó képet is. Legalul a Bejegyzés időpontja található.

Jobb oldalon a két gombot találunk: egy kék visszafelé mutató nyíl, erre kattintva az adott Bejegyzéshez tartozó Szakaszfolyamot nyitjuk meg, valamint egy szürke fogaskerék, mellyel az adott Bejegyzést tudjuk szerkeszteni.

Utóbbira kattintva egy ’Edit Entry’ ablak nyílik meg, ahol tudjuk módosítani a bejegyzéshez tartozó címet, történetet és láthatóságot. Ha még nincs lezárva az Utazás, akkor a ’Delete’ gomb segítségével törölni is tudjuk a Bejegyzésünket, ha pedig az ’Update’ gombra kattintunk, akkor mentésre kerülnek a változtatások.

#### Felhasználó folyamelem

A Felhasználó folyamelem bal oldalán található a felhasználó felhasználóneve, közvetlenül alatt pedig az eredeti neve, valamint zárójelben az egyedi azonosítója.

A Felhasználó folyamelemnek három különböző háttérszíne lehet, melyek azt mutatják, hogy milyen kapcsolatban vagyunk vele:

* **Szürke**: Nem barát,
* **Kék**: Folyamatban lévő barátjelölés,
* **Zöld**: Barát.

Jobb oldalon a két gombot találunk: egy zöld pipa gombot, erre kattintva ismerősnek jelölhetjük az adott felhasználót, ha pedig minket jelöltek be, akkor elfogadhatjuk a barátfelkérést. E mellett található egy kék szemetes gomb, mellyel el tudjuk távolítani az adott felhasználót a barátlistánkról – persze abban ez esetben, ha rajta volt.

Ha magára a folyamatelemre kattintunk, akkor az adott felhasználóhoz tartozó Utazásfolyam kerül megjelenítésre a Folyamablakban.

### Folyamok

A Folyam nem más, mint egy adott elemtípusokból valamilyen logikai szempont alapján összeállított folyamelemlista.

* **Főfolyam**: A Barátok bejegyzései kerülnek a Főfolyamba, időrendben csökkenő sorrendben, de csak az utolsó 3 nap bejegyzései jelennek meg.
* **Utazás-folyam**: Egy adott felhasználóhoz tartozó összes Utazás kerül a Folyamba. Időben csökkenő sorrendben.
* **Szakasz-folyam**: Egy adott Utazás összes Szakasza kerül a folyamba. Időben csökkenő sorrendben.
* **Bejegyzés-folyam**: Egy adott Szakasz összes Bejegyzése kerül a folyamba. Időben csökkenő sorrendben.
* **Történet-folyam**: Egy adott Utazás összes Bejegyzése kerül a folyamba, időben növekvő sorrendben.
* **Felhasználó-folyam**: Egy keresési érték alapján kilistázott felhasználók jelennek meg a Folyamban.
* **Barát**-**folyam**: Egy adott felhasználó Barátainak a listája kerül a folyamba.
* **Jelölés-folyam**: Egy adott felhasználó barátjelöléseinek a listája kerül a folyamba.

#### Főfolyam

Bejelentkezéskor, ez fogad bennünket a Folyamunkon. Az összes ismerősünk bejegyzései kerülnek itt felsorolásra időrendi sorrendben, de csak a 25 legfrissebb fog megjelenni. Ha szeretnénk, hogy újra a Főfolyam jelenjen meg a folyamablakban, akkor kattintsunk a Folyam ablak jobb oldalán lévő házikó gombra.

Más felhasználó Főfolyama nem megtekinthető.

#### Utazás-folyam

Kilistázásra kerül egy adott felhasználóhoz tartozó összes Utazás, az aktív utazás háttere lazacszínű. Ha rákattintunk valamelyik Utazásra, akkor az adott Utazáshoz tartozó Szakasz-folyam kerül megjelenítésre a Folyamablakban.

* A saját Utazás-folyamunkat kétféle képpen tudjuk megnyitni:

1. Ha a folyamablak melletti zászlós gombra kattintunk,
2. vagy ha rákattintunk a menüsorban a *Traveling/New Entry* menüpontra.

* Ha más felhasználónak szeretnénk megtekinteni az Utazás-folyamát, akkor rá kell keresni az adott felhasználóra (lásd felhasználó vagy barát keresése), majd rá kell kattintani a kiválasztott Felhasználó folyamelemre. Ha a felhasználó a barátunk, akkor a publikus és barátoknak engedélyezett elemek fognak megjelenni, ha nem, akkor csak a publikusak lesznek számukra láthatóak.

#### Szakasz-folyam

Kilistázásra kerül egy adott Utazáshoz tartozó összes szakasz folyamelem, persze csak akkor, ha van jogosultságunk a megtekintésükre. Az aktív Szakasz háttere lazacszínű. Ha rákattintunk valamelyik Szakaszra, akkor a hozzá tartozó Bejegyzés-folyam kerül megjelenítésre a Folyamablakban.

Bármely Szakasz-folyam úgy érhető el, hogy rákattintunk valamelyik Utazás folyamelemre.

#### Felhasználó-folyam

Kilistázásra kerül a felhasználónév keresési feltétel alapján az első 10 Felhasználó folyamelem. Kék háttérrel rendelkeznek azok, akik vagy be lettek barátnak jelölve, vagy pedig bejelöltek bennünket barátnak. Zöld háttérrel rendelkeznek azok, akik a barátaink.

A Felhasználó folyam a keresés funkcióval érhető el. (Lásd a keresésnél)

#### Barát-folyam

Valójában ez is egy Felhasználó folyam, itt viszont minden barátunk Felhasználó folyameleme megjelenítésre kerül. Meg tudjuk nézni más felhasználók barátlistáját is.

A Barát-folyam két módon érhető el:

* A sajátunkat *Friends/My Friends* menüpontra kattintva kapjuk meg,
* vagy ha Folyamablak jobb oldalán szereplő Barátok gombra kattintunk, akkor az aktuálisan megnyitott Folyamablakhoz tartozó Felhasználó barátlistája fog megjelenni. Abban az esetben, ha a Folyam nem köthető egy konkrét személyhez, akkor a gomb inaktívvá válik.

#### Jelölés-folyam

Ez szintén Felhasználókból épül fel, itt az aktuális barátjelölések vannak kilistázva.

Az Jelölés-folyam két módon érhető el:

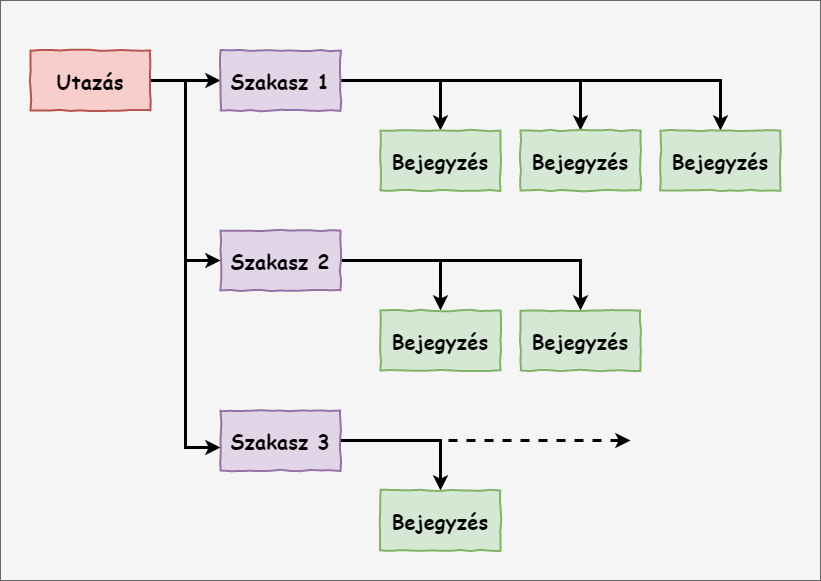
* A *Friends/Invitations* pontra kattintva érhető el.
* Vagy pedig a Folyamablak jobb oldalán található Jelölések gombra kattintva.

# 3. Fejlesztői kézikönyv

## A DiceTravel koncepciója

### Mi is a DiceTravel?

A DiceTravel alapvetően egy naplózó alkalmazás, mely utazásokkal kapcsolatos eseményeket tárol, rendszerez és jelenít meg a felhasználó számára. A DiceTravel továbbá egy platformot nyújt, hogy a felhasználók megoszthassák egymással az élményeiket.

Minden utazást kisebb részekre lehet osztani, ezeket szakaszoknak nevezzük. Egy esemény, melyről bejegyzés készül, mindig egy adott szakasz alatt történik. Ezért az adott bejegyzés az adott szakasz alatt kerül rögzítésre.

Egy szakasz befejezése egy új szakasz indítását vagy az utazás végét indukálja. Mivel ez egy logoló, és nem pedig tervező alkalmazás, nem lehetséges egyszerre több utazáson, szakaszban részt venni. Ettől függetlenül egy felhasználónak tetszőleges számú lezárt utazása lehet, egy utazásnak tetszőleges számú szakasza.

A bejegyzéseink, szakaszaink és utazásaink láthatóságának három szintje van: Privát, Barátok, Publikus.

Baráti kapcsolatokat jelölés-visszaigazolás módszerrel lehet létesíteni, ami azt jelenti, hogy tetszőleges felhasználónak küldhetünk barátjelölést, viszont a barátság csak abban az esetben kerül aktiválásra, ha ezt a célfelhasználó elfogadja.

A felület egyik legfontosabb része, amit szeretnék már itt kiemelni: a Folyam.  
A Folyamban fognak megjelenni a különböző keresésekkel kapcsolatos találatok, valamint összességében elmondható, hogy egy kontrol, az nem más mint egy adatbázis entitás. Például a felhasználó utazásai, más felhasználók utazásai, vagy egy felhasználó barátai. Ezek mind valamely Control osztály példányai lesznek.

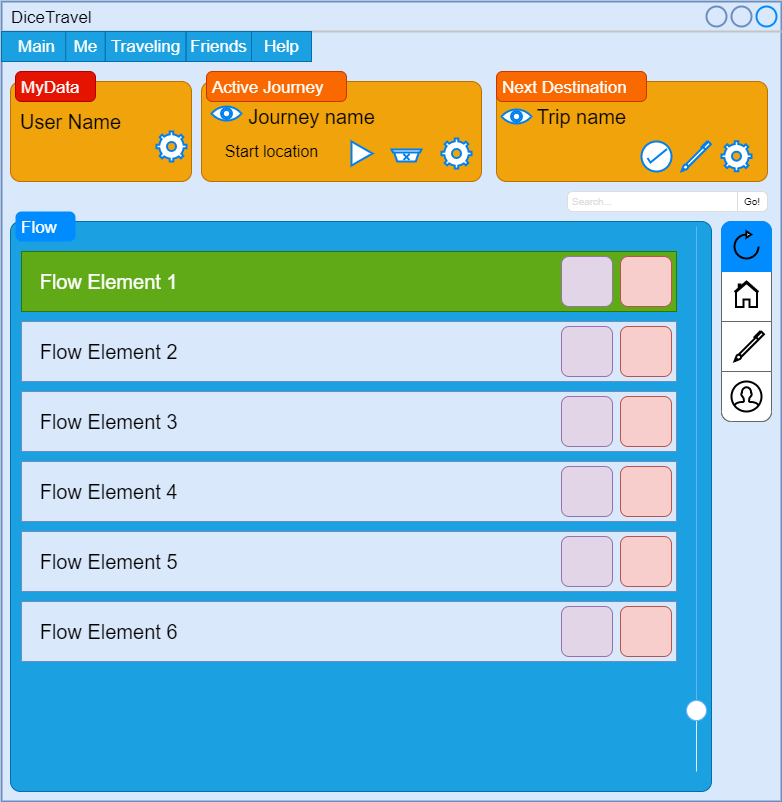
### Kulcsszavak

* **Utazás (Journey):** A DiceTravel utazással kapcsolatos legnagyobb logikai egysége, mely alatt a Szakaszok találhatóak. Egy kezdőponttal rendelkezik, valamint egy elnevezéssel.
* **Szakasz (Trip):** Az utazás elemei. Minden szakasz egy végponttal rendelkezik. A kezdőpont az előző szakasz végpontja határozza meg, vagy ha még nincs teljesített szakasz, akkor az Utazás kezdőpontja lesz.
* **Bejegyzés (Entry):** A DiceTravel utazással kapcsolatos legkisebb logikai egysége. Minden Bejegyzés egy szöveges bejegyzésbő, ami maximum 1024 karakter hosszú lehet, és/vagy egy képből áll. Utóbbi maximum 16 MB méretű lehet.
* **Folyam (Flow):** A különböző utazással/barátokkal kapcsolatos szűrések nézetei. Egy felhasználó Utazás folyama például az összes általa készített Utazást listázza ki és jeleníti meg. A következő folyamok vannak például a DiceTravelben: Utazás, Történet, Aktuális, Barát.
* **Barát (Friend):** Olyan felhasználó, akivel láthatjuk egymás csak barátoknak szánt bejegyzésit is.

## A DiceTravel felépítése

### a GUI terve

Az alkalmazás főablaka 3 részre osztható:

* A menükbe kerülnek a fontosabb parancsok, de ezek többsége magán az ablakon is implementálásra kerülnek.
* Az adatáttekintő szekcióba Felhasználóval, Aktív utazássa, és a Következő Szakasszal kapcsolatos információk és legfontosabb funkciók kerülnek.
* A Folyamban jelennek meg a felhasználó által lekérdezett adatok. A jobb oldalán pedig az alapvető Folyam nézetek gombjai találhatóak.

### Osztálytípusok

Szeretném általánosságban ismertetni, hogy a Dice Travel hogyan épül fel, milyen osztálytípusokkal doglozik.

#### Entitások

Az entitások az adatbázissal való kommunikációra lesznek kifejlesztve, valamint az adott tábla egy adott sorának az programbeli leképzései lesznek. A hozzá tartozó metódusokkal könnyen képesek leszünk az alapvető CRUD műveletek végrehajtására, valamint a fontos lekérdezések indítására.

A koncepció alapján az alkalmazás az alábbi entitásokkal rendelkezik:

**Utazás, Szakasz, Bejegyzés, Felhasználó, Barátság.**

Ezek az entitások lesznek az alkalmazás adatbázisában tárolva, valamint ezek lesznek a programunk főbb osztályai, amik a megfelelő adatok manipulálására lesznek használva. Ezek mind az *Entity* absztrakt osztályból fognak származni.

Ezek egy közös *EntityClasses* mappába lesznek szervezve.

#### Formok

A felhasználó különböző űrlapokon keresztül tud majd létrehozni/módosítani/törölni bizonyos entitásokat az adatbázisból. Ezek szintén külön a *Forms* mappába lesznek szervezve, azon belül is entitásonkénti almappába. Valójában ezek az űrlapok fognak lehetőséget adni az adott entitás metódusainak felügyelt hívásaira.

Az alkalmazásnak a *MainForm* űrlap osztálya fog keretet adni, ez lesz a fő ablaka az alkalmazásnak. Ez szintén a Forms mappában fog szerepelni, közvetlenül az entitás almappák mellett egy szinten.

#### Kontrollok

A kontroll egy entitásnak a folyamban való megjelenése. Segítségével a felhasználó az adott entitással kapcsolatos műveleteket végezhet. Például törölheti az entitást, vagy kontextustól függően megváltoztathatja a folyamot. Az űrlapokhoz hasonlóan itt is a megfelelő entitások ellenőrzött hívásai fognak történni.

A kontrollok a Controls mappába lesznek szervezve.

#### Kiegészítő osztályok

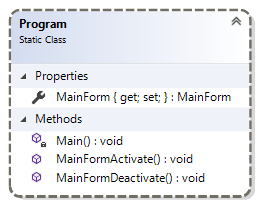
A Util mappán belül a különböző kiegészítő osztályok szerepelnek. Ebből három van:

* *ActiveUserStore*: Ebben a statikus osztályban van letárolva az éppen bejelentkezett felhasználó, valamint ez az osztály felel a ki és bejelentkeztetésért.
* *Encryptor*: Ez a statikus osztály tud egy string típusú adatot MD5 kódolási algoritmus segítségével titkosítani. Az *ActiveUserStore* igényli, hogy meg tudja csinálni a regisztrációt és bejelentkeztetést.
* *FlowElementProvider*: Ez az osztály felel a folyam összeállításáért/tárolásáért/frissítéséért.
* *ImageHandler*: Ez a statikus osztály képes egy képet bináris formára hozni, hogy azt fel lehessen tölteni az adatbázisba.

## Az osztályok részletezése

Terjedelemi okok miatt csak az alapvető osztályokat és metódusokat szeretném részletezni, valamint egy osztálytípusnál csak egy konkrét osztályt fogok bemutatni. A dokumentumban a Visual Studio 2019 által generált osztály-diagrammokat használom.

### Program osztály

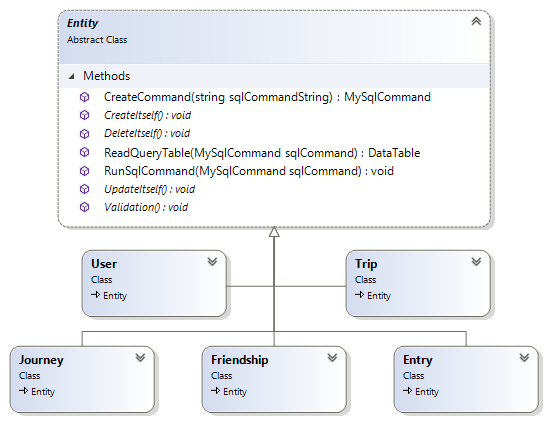
A statikus Program osztályunk tartalmazza a *Main* metódust. Ezen felül tartalmaz egy MainForm nevű statikus property-t, melyben a MainForm egy példánya lesz tárolva. Ennek csak a get ága lesz publikus, hogy a későbbiekben ne lehessen megváltoztatni. Rögtön a Program indulásakor megtörténik a MainForm példányosítása.

A Program osztály fog gondoskodni arról, hogy a MainFormunk aktiválásáról és deaktiválásáról. A Deaktiválásra akkor van szükség, ha egy párbeszédablak jelenik meg, természetesen aktiválni pedig akkor kell, ha bezáródik a párbeszédablak.

### MainForm osztály

A MainForm osztály egy példánya lesz az alkalmazásunk főablaka. Egyetlen MainForm lesz az alkalmazás indulásakor, ami nem lesz módosítható a kód többi részében a private hatáskör miatt. \*\*\* ezt még jobban ki kell fejteni…

### Entitások

Minden az adatbázisban szereplő sor egy-egy entitásnak felel meg. A különböző entitások hasonlóan viselkednek és ezért mind az Entity absztrakt osztályból származnak.

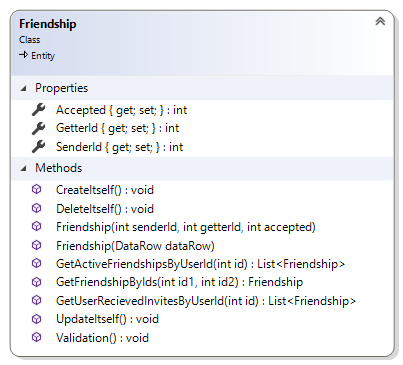
Az Entity osztály elvárja minden gyermek osztályától, hogy implementálja a *CreateItself, DeleteItself, UpdateItself, Validation* metódusokat. Ezek fogják megvalósítani az adott entitás alap adatbázis műveleteit. Ha manuálisan létrehozunk egy entitást, akkor a *CreateItself* segítségével tudjuk felvinni új sorként az adatbázisunkba. A *DeleteItself* metódus az adott entitást törli az adatbázisból. Ha egy entitáson módosításokat hajtunk végre, akkor az *UpdateItself* metódussal tudjuk az adatbázisban is módosítani az adott sort. A *Validation* metódussal meg tudjuk vizsgálni, hogy egy entitás az adatbázisnak megfelelő adatokat tartalmaz, ha bármi probléma adódik, akkor egy *ValidationException* fog dobásra kerülni.

Az Entity absztrakt osztály tartalmaz továbbá három statikus metódust is mely az adatbázishasználathoz szükséges gyakori lépéseket rendszerezi, hogy csökkentve legyen a kódban a duplikáció. A *CreateCommand* egy új *MySqlCommandot* hoz létre a megadott MySql parancs alapján, amit egy String paraméterként kap. A *RunSqlCommand* egy MySqlCommand futtatásáról gondoskodik. A *ReadQueryTable* pedig egy DataTable-t állít össze egy MySqlCommand alapján. Jellemzően akkor van használva, ha egy lekérdezést szeretnénk indítani.

#### Új entitás osztályok létrehozása

Minden entitásnak az Entity ősosztályból kell származnia, implementálnia kell annak absztrakt metódusati, valamint a következő adattagokkal, konstruktorokkal és metódusokkal kell őket kiegészíteni.

* Minden adatbázisban szereplő oszlopnak létre kell hozni egy megfelelő típusú és elnevezésű publikus property-t get és set ággal egyaránt.
* Készíteni kell egy konstruktort, ami **adatbázis** **lekérdezés** segítségével fogja előállítani egy entitás példányát. A paraméter egy DataRow lesz, és ennek a megfelelő értéke kerül beállításra az entitás megfelelő propery-jébe.
* Készíteni kell egy másik konstruktort, ami a **felhasználó** **által az adatbázisban** **létrehozandó új** entitások előállítására lesz fenntartva. Paraméterként meg kell kapnia az összes adattagot, kivéve az Id azonosítókat, mert ezeket az értékeket közvetlenül az adatbázis fogja beállítani.
* Dinamikus Get metódusokat, amik az adott entitáshoz szorosan köthető más entitásokkal térnek vissza az adatbázisból. Például egy utazás alá egy vagy több szakasz tartozhat, szükséges létrehozni egy olyan metódust, ami az adott utazáshoz tartozó szakasz entitások listájával tér vissza. Ezt a lista lesz felhasználva egy utazáshoz tartozó szakasz-folyam kialakításakor.
* Statikus Get metódusokat kell létrehozni a különböző adatbázis olvasással kapcsolatos feladatokhoz. Ezek vagy adott entitással térnek vissza, vagy adott entitás típusú listákkal. Az applikáció továbbfejlesztése esetében csak a megfelelő adatbázis lekérdezésekkel kell az osztályt bővíteni.

**Friendship osztály bemutatása:**

A Friendship osztály reprezentálja az adatbázisban, hogy a különböző felhasználók milyen baráti kapcsolatban vannak. Három mező található erre az adatabázis ’friends’ kapcsolótáblájában: a *sender\_id*, a *getter\_id*, valamint az *accepted*. A *sender\_id* azonosítja a barátkérelem elküldőjét, a *getter\_id* a barátkérelem célpontját, az *accepted* pedig azt, hogy a barátkérelem elfogadásra került-e. (0: még nem került elfogadásra, 1: elfogadásra került) Abban az esetben, ha elutasításra került egy barátkérelem, akkor a megfelelő sor törlésre kerül a táblából.

A Friendship osztálynak ennek megfelelően rendelkeznie kell típushelyes SenderId, GetterId, és Accepted property-kkel. Kell lennie egy írásra és egy olvasásra alkalmas konstruktornak. Valamint szükséges lesz a következő három metódusra.

* **GetFriendshipByIds**(int id1, int id2):Friendship:ezzel lehet keresni egy barátságra, ahol tudjuk a két felhasználó azonosítóját.
* **GetRecievedFriendshipInvitesByUserId** (int id): List<Friendship>: Ez a metódus egy adott felhasználóhoz tartozó olyan barátságokkal tér vissza, melyeket még nem fogadott el a felhasználó. Paraméterként a felhasználó azonosítóját kell megadni.
* **GetActiveFriendshipsByUserId** (int id): List<Friendship>: Ez a metódus egy felhasználó aktív barátságlistájával tér vissza. Paraméterként a felhasználó azonosítóját kell megadni.

Ha szeretnénk adott esetben egy új funkciót bevezetni, például hogy legyen kilistázva a flowban egy felhasználónak az összes általa elküldött barátkérelme, akkor csak létre kell hozni egy új statikus metódust, aminek a neve lehet **GetSentFriendshipInvitesByUserId,** ami szintén egy List< Friendship >-val tér vissza.

### Formok

Az entitásokat két féle képpen lehet kezelni ­- gondolok itt azok létrehozására, törlésére, módosítására, - formok, vagy pedig kontrollok segítségével. Most a Form osztályokról szeretnék pár szót szólni.

Minden form hozzárendelhető egy entitáshoz, és rendelkezik olyan mezőkkel, vagy más felületi elemmel, melyek a felhasználó által megadandó információk bekérését teszik lehetővé. Minden entitáshoz két fajta form tartozik, az egyik az entitás létrehozásáért a másik az adott entitás frissítéséért felel.

A projekten belül a Form osztályok a Forms mappába vannak szervezve, valamint a az egy Entitáshoz tartozó osztályok az EntitásNeveForms mappában találhatóak.

Minden Formnak validálnia kell, hogy a bevitt adat megfelelő, erre a Validation metódust kell használni, ami meghívja az adott entitás validációját, mert minden entitás tudja, hogy mi a jó neki. Csak ezután történhet adatabázis utasítás végrehajtás.

Fontos, hogy már a beviteli és más mezők esetében is biztosítani kell, hogy lehetőleg csak a megfelelő formátumú és típusú adatok kerülhessenek bevitelre.

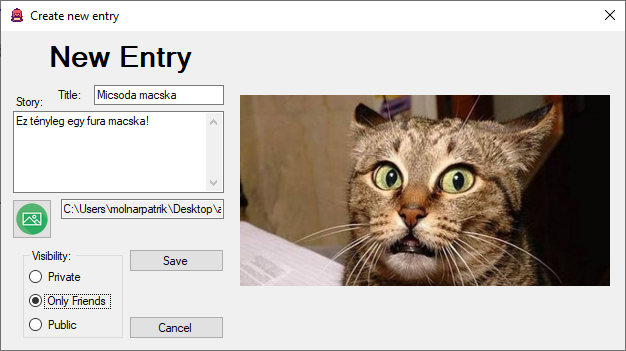
*Megjegyzem, hogy a MainForm, ami az alkalmazás fő ablaka, nem tartozik ehhez az osztálytípushoz.*

Természetesen minden Form a System.Windows.Forms.Form osztályt implementálja.

**Az Entry entitás Form osztályainak bemutatása:**

Az Entry entitásnak két Formja van az előzőekben tárgyaltak miatt.

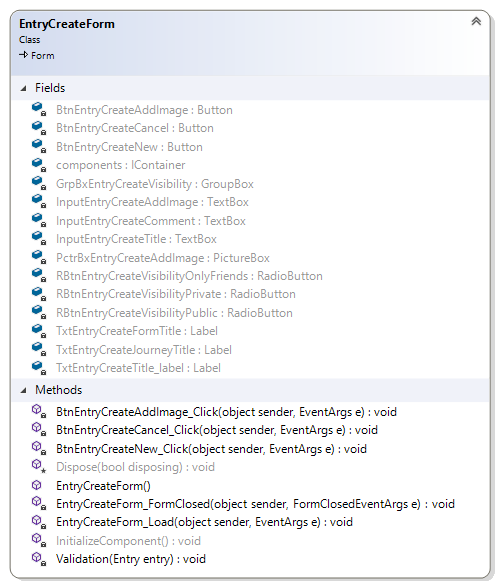
Az EntryCreateForm, valamint az EntryUpdateForm.

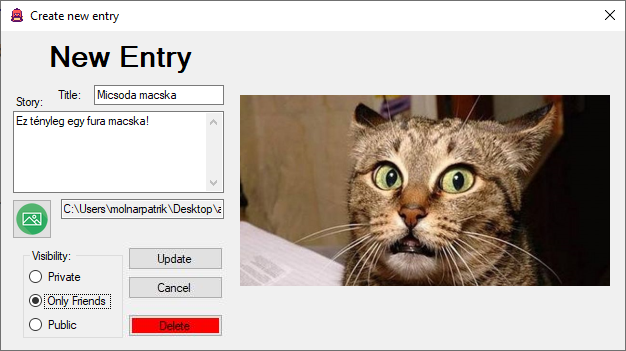
**Az EntryCreateForm:**

* **Title**: Ez az adott Entry címe, az adatbázisban az entries tábla title oszlopa fogja tárolni ezeket az értékeket.
* **Story**: Ez az adott Entry leírása, ide kerül az adott bejegyzés írásos része, az entries tábla comment oszlopa fogja tárolni.
* **Kép kiválasztása gomb:** Ennek a segítségével lehet egy megnyíló ablakban kiválasztani a feltölteni kívánt képet. Ez a kép fog az entries tábla megfelelő rekordjának picture oszlopában letárolódni.
* **Visibility:** Itt lehet beállítani az Entry láthatóságát, ez az entries tábla visibility oszlopába kerül.
* **Save gomb:** Ennek a segítségével validáció után történik meg az adatbázisba írás. Form bezárása.
* **Cancel gomb:** Az új bejegyzés megszakítása. Form bezárása.

**Az EntryCreateForm metódusai:**

Az EntryCreateFormnak csak paraméter nélküli konstruktora van. Ő maga fogja előállítani az Entry-t és a DB-be írni.

* **BtnEntryCreateCancel\_Click(object sender, EventArgs e):** Ez történik a *Cancel* gombra kattintás esemény hatására. Az új bejegyzés megszakítása kerül a Form pedig bezáródik.
* **BtnEntryCreateNew\_Click(object sender, EventArgs e):** Ez történik a *Save* gombra kattintás esemény hatására. Az új Entry validálásra kerül, majd új rekordként hozzáadásra kerül az adatbázis entries táblájához.
* **Validation(Entry entry):** Meghívja a paraméterként átadott entry validációs metódusát, ami bármilyen kritikus hiba esetében *ValidationException*-t dob. Ez meggátolja, hogy az adatbázisba fel legyen véve az új rekord. Ez jelen esetben konkrétan azt jelenti, hogy csak akkor valid egy entry, ha a title és a comment nem üres, valamint kötve van hozzá ID alapján egy trip.
* **BtnEntryCreateAddImage\_Click(object sender, EventArgs e):** Ez a metódus a Kép hozzáadása gombra kattintáskor fut le, ez felel egy OpenFileDialog megnyitásáért, valamint az elérési útvonal beállításáért, ami majd az adatbázisba íráskor lesz fontos.
* **EntryCreateForm\_Load(object sender, EventArgs e):** Ez a metódus a form betöltésekor végrehajtandó fontos lépésekre való, mint például a MainForm deaktiválása, hogy amíg nyitva van a Form ne lehessen a főablakban műveleteket végrehajtani.
* **EntryCreateForm\_FormClosed(object sender, FormClosedEventArgs e):** Ez a metódus a form bezárásakor végrehajtandó fontos lépésekre való, mint például a MainForm aktiválása, hogy újra lehessen műveleteket végrehajtani a főablakban.

**Az EntryUpdateForm:**

Szinte teljesen megegyezik az *EntryCreateForm* felépítésével, azonban itt nem *Save* gomb található, hanem *Update*, valamint van még egy további gomb is a felületen, ami a *Delete*. A további elemek megegyeznek.

Az osztálynak egy paraméteres konstruktora van, ami egy *Entry*-t vár. Ez az *Entry* már egy, a DB-ben lévő entitás, azt szeretnénk majd a form segítségével UPDATE-elni. Ez egy *Entry* nevű property-ben lesz eltárolva. A form mezőiben már ennek az entitásnak az értékei szerepelnek, ha ezeket módosítjuk, majd az Update gombra kattintunk, akkor lefut az *Entry* validációja, és ha nincs probléma megtörténik a DB UPDATE.

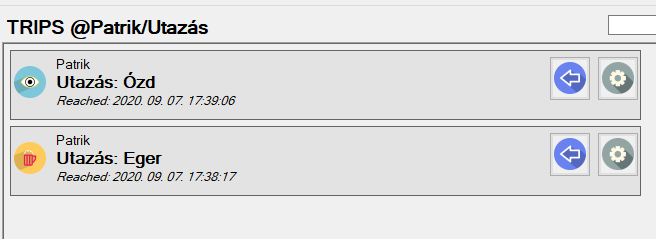
A konstruktornak ezen kívül biztosítania kell, hogy ha az *Entry* egy lezárt *Journey*-hez tartozik, akkor csak a *Visibility* értéke módosítható, a többi érték nem lehet változtatható. Ezt az adott mező lezárásával kell elérni.

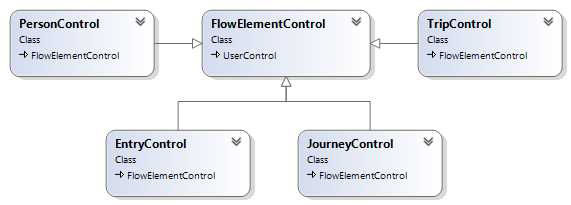
Az Update gombra kattintáskor a BtnEntryUpdateUpdate\_Click(object sender, EventArgs e) metódus fut le. Ez egyébként nagyon hasonlít a *Save* gombhoz, annyiban különbözik, hogy nem egy CREATE sql parancs fut le ha megtörtént a validáció, hanem egy UPDATE.

A Delete gombra kattintva lefut a BtnEntryUpdateDelete\_Click(object sender, EventArgs e) szignatúrájú metódus, ami az adott *Entry* entitás *DeleteItself()* metódusával törli az adott rekordot az adatbáziaból. Majd egy MainForm frissítés után bezáródik.

A további metódusok hasonló feladatokat valósítanak meg, mint az EntryCreateFormé**.**

### Kontrollok

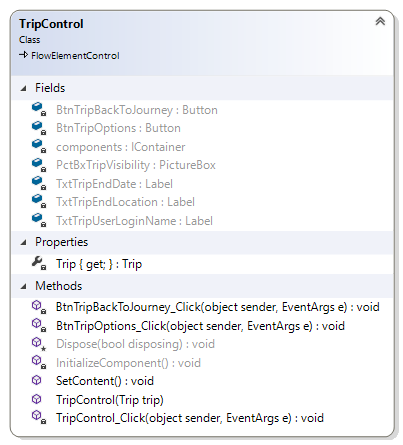
A DiceTravelben található kontrollok, azok a Flow elemeit jelentik. Hasonlóan a Formokhoz a Kontrollok szintén interaktív felületek, melyek megjelenítik az adott entitások legfontosabb információit, valamint lehetővé teszik bizonyos entitásokhoz köthető funkciók végrehajtását. *Ilyen például egy adott felhasználó barátnak jelölése, vagy, hogy megtekintsük, hogy egy adott Journey alatt milyen Tripek találhatóak.*

Minden kontroll a *FlowElementControl* osztályból származik, mely leginkább vizuális effekteket valósít meg.

Nincs kontroll valamilyen entitás nélkül. Például a Trip kontroll minden eseteben tartalmaz egy Tripet.

A mezők többnyire szöveges információt jelenítenek meg, rendelkeznek egy vagy több funkciógombbal, valamint legtöbbjükre rá is lehet kattintani. Nincsenek beviteli mezőik.

**A TripControl osztály bemutatása:**

****A **TripControl** osztály egy a flowban megjelenő kontrollnak az implementációja, melynek példányai különböző tripeket fognak reprezentálni, valamint azok funkcióit elérhetővé tenni a felhasználó számára.

A trip kontrollon szerepel a trip birtokosának felhasználóneve, a Journey neve, a trip következő állomása, valamint egy jel, ami a láthatóságot szimbolizálja. Ezeken kívül egy vissza gomb, valamint egy beállítások gomb.

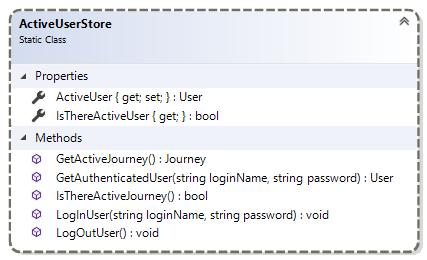
A Trip kontroll rendelkezik egy csak olvasható *Trip* típusú property-vel, melyet az egyetlen konstruktor fog beállítani a kapott *Trip* típusú paraméter alapján. A konstruktor ezen kívül beállítja még a kontrollon megjelenő adatokat a *SetContent()* metódus segítségével, valamint a kontroll stílusát is meghatározza a *Trip* alapján.

A kontrollra kattintva megnyílik az adott triphez tartozó *bejegyzés-folyam*. A vissza gombra kattintva visszaléphetünk az adott felhasználó *utazás-folyamjára*, a beállításokkal pedig az adott triphez tartozó *TripUpdateForm*-ja jelenik meg.

## Kiegészítő osztályok

### ActiveUserStore

Ez a statikus osztály gondoskodik a felhasználók be- és kijelentkeztetéséről, tárolásáról.

Az osztály az ActiveUser property-ben tárolja a bejelentkezett felhasználót. Ha ez az érték null, akkor nincs senki sem bejelentkezve a rendszerbe.

Annak eldöntésére, hogy valaki be van-e jelentkezve az IsThereActiveUser csak olvasható property adja vissza.

A LogOutUser() metódus segítségével tudjuk kijelentkeztetni a felhasználót, ilyenkor az ActiveUser property értéke null lesz, valamint lefut a FlowElementProvider ResetFlow() metódusa, ami alaphelyzetbe állítha a Flow-t.

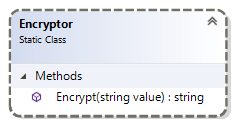
A LogInUser(string loginName, string password) metódus fogja a felhasználót beléptetni a rendszerbe, paraméterként a belépési nevet és jelszót kapja meg. Itt jelszó értéke már sózva és titkosítva van az Encryptor osztály segítségével.

A GetAuthenticatedUser(string loginName, string password) metódus visszatér egy User entitással abban az esetben, ha a felhasználónév, valamint a hozzá tartozó jelszó megfelelő. Ez egy private metódus, és egyedül a LogInUser metódus hívja.

A GetActiveJourney() metódus az épp bejelentkezett felhasználó aktív utazásával tér vissza.

Az IsThereActiveJourney() metódus igaz/hamis értékkel tér vissza, attól függően, hogy a bejelentkezett felhasználónak van-e aktív utazása. Azért készítettem ezt a metódust, hogy próbáljam növelni a kódom olvashatóságát.

### Encryptor

Az Encryptor osztály felel a jelszavak titkosításáért. Egy metódust tartalmaz csak, ez az Encrypt(string value). Ennek a visszatérési értéke lesz a titkosított String típusú adat. Három helyen lesz használva, a regisztrációs felületen, a bejelentkezésnél, valamint a jelszó változtatásnál.

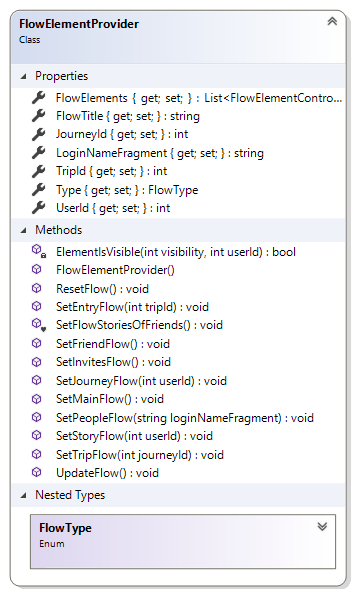
### FlowElementProvider

A FlowElementProvider felelős a folyamban megjelenő elemek létrehozásáért, a flow összeállításáért és megjelenítéséért.

A MainForm rendelkezik egy példánnyal belőle, ami rögtön a MainForm példányosításakor jön létre.

A FlowElementProvider rendelkezik egy Enummal, ami az összes folyamtípust azonosítja, ez a FlowType.

A következő propertiket kell megvalósítani az osztálynak:

* Type: A Flow típusát tárolja, hogy a flow frissítése esetén emlékezzen, hogy mit kelll előállítania.
* UserId: A Flowban megjelenítendő elemekhez tartozó *UserId*-t tárolja. Abban az esetben, ha olyan flow kerül előállításra, ahol ez nem értelmezhető -1 értéket kap.
* JourneyId: A Flowban megjelenítendő elemekhez tartozó *JourneyId*-t tárolja. Abban az esetben, ha olyan flow kerül előállításra, ahol ez nem értelmezhető -1 értéket kap.
* TripId: A Flowban megjelenítendő elemekhez tartozó *TripId*-t tárolja. Abban az esetben, ha olyan flow kerül előállításra, ahol ez nem értelmezhető -1 értéket kap.
* FlowTitle: A folyam megjelenítendő címe kerül tárolásra ebben a property-ben.
* LoginNameFragment: A felhasználó keresésekor felhasznált szótöredék kerül eltárolásra ebben a property-ben.
* FlowElements: A Flowban megjelenítendő kontrolok kerülnek itt eltárolásra. Ennek beállítására csak a FlowElementProvidernek van jogosultsága.

A konstruktor egyedül a ResetFlow() metódust tartalmazza, ami a flow alaphelyzetbe állításáért felel.

Az UpdateFlow() metódus felel a folyam frissítéséért. Ha felhasználó a frissítés gombra kattint, vagy bármikor, amikor frissíteni kell a folyamot, ez a metódus fut le. Az eltárolt értékek alapján újra generálja a flowt.

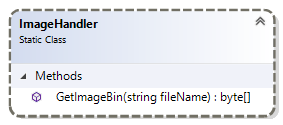
Egy nagyobb metódus fajta a folyambeállító metódusok.

A következő Flow beállító metódusokra van szükség:

* SetMainFlow(): A folyamnak a főfolyamot állítja be, ahol a bejelentekezett felhasználó ismerőseinek a 3 napnál korábbi bejegyzései jelennek meg.
* SetPeopleFlow(string loginNameFragment): A folyamnak Felhasználó-folyamot állítja be, ahol a kereső mezőben megadott feltételnek megfelelő felhasználók jelennek meg.
* SetFriendFlow(): Az bejelentkezett felhasználó Barát-folyama jelenik meg.
* SetJourneyFlow(int userId): Az adott felhasználó Utazás-folyama jelenik meg.
* SetTripFlow(int journeyId): Az adott utazás Szakasz-folyama jelenik meg.
* SetEntryFlow(int tripId): Az adott Szakasz Bejegyzés-folyama jelenik meg.
* SetStoryFlow(int userId): Az adott felhasználó Történet-folyama jelenik meg.
* SetInvitesFlow(): A bejelentkezett felhasználó Jelölés-folyama jelenik meg.

A kiegészítő ElementIsVisible(int visibility, int userId) metódus egy igaz/hamis értékkel tér vissza attól függően, hogy az adott felhasználó láthatja-e az adott láthatósági szintet. Ennek a segítségével lehet kiszűrni azokat a kontrollokat, amiket nem láthat a felhasználó.

### ImageHandler

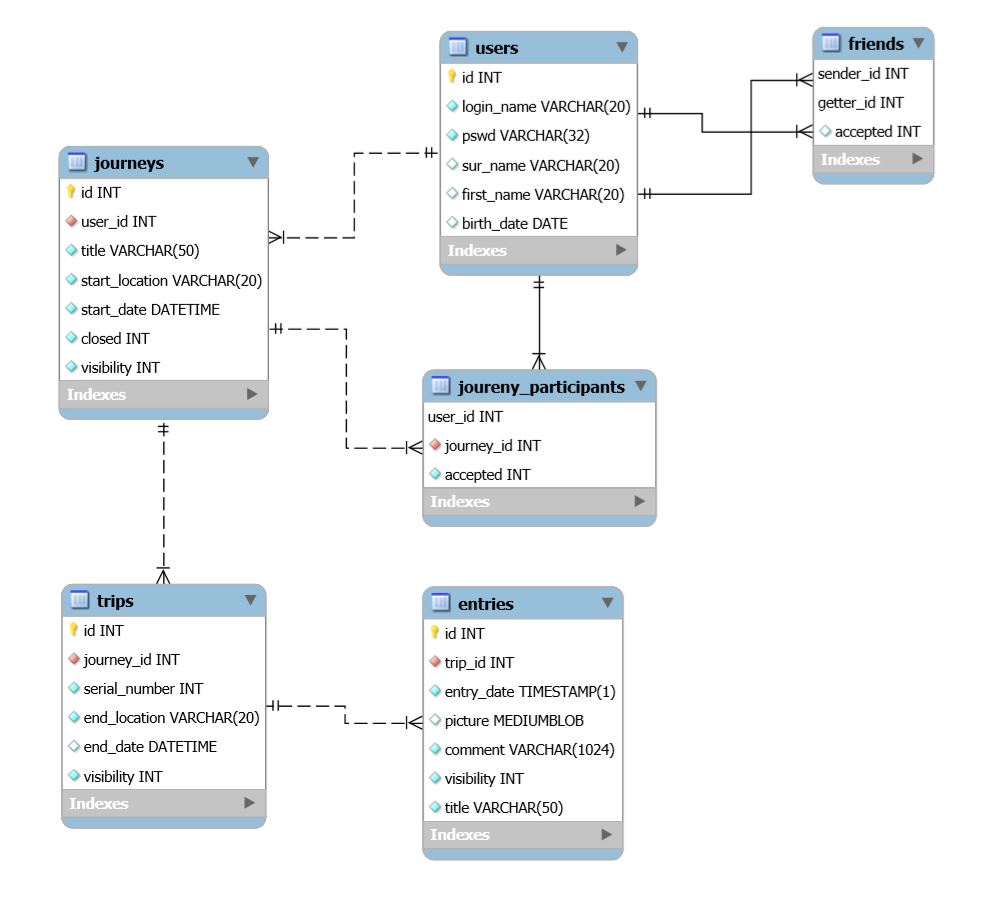
Ez a statikus osztály képes egy képet bináris formára hozni, hogy azt fel lehessen tölteni az adatbázisba.

A GetImageBin(string fileName) metódus egy elérési út alapján egy képet bináris többé alakít. Ez a bináris formátum lesz majd az adatbázisban letárolva.

# 4. Adatbázisterv

Az alábbi fejezetben felvázolom, hogy miképpen valósítom meg az alkalmazás adatbázisát.

## Az adatbázis grafikus ábrája



## Az adatbázis tábláinak részletes leírása

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Táblanév** |  |  |  |  | **Komment** |
| users | | |  |  | A userekkel kapcsolatos információkat tartalmazó tábla. |
| **Mezőnév** | **Típus** | **Kulcsok** | **Null-e** | **Alapérték** | **Komment** |
| id | number | primary | not null | 0 | User azonoító. |
| login\_name | varchar(20) | unique | not null | üres string | A User egyedi neve a rendszerben. |
| pswd | varchar(32) |  | not null | üres string | A User jelszavának hash kódja. |
| sur\_name | varchar(20) |  |  | NULL | A User keresztneve. |
| last\_name | varchar(20) |  |  | NULL | A User vezetékneve. |
| birth\_date | date |  |  | NULL | A User születési dátuma |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Táblanév** |  |  |  |  | **Komment** |
| journeys | | |  |  | Journey tábla, ez tárolja a Journey indulási adatait |
| **Mezőnév** | **Típus** | **Kulcsok** | **Null-e** | **Alapérték** | **Komment** |
| id | number | primary | not null | ai | Journey azonosító |
| user\_id | nubmer | foreign | not null | 0 | Users azonosítója, aki indította a Journeyt. |
| title | varchar(50) |  | not null | „N.A.” | Rövid leírás a Journeyről. Ez a cím. |
| start\_location | varchar(20) |  | not null | NULL | A Journey kiindulópontja. |
| start\_date | date |  | not null | NULL | A Journey indításának a dátuma. |
| closed | number |  | not null | NULL | Azt mutatja, hogy le van-e zárva a Journey. |
| visibility | number |  | not null | 1 | A Journey láthatósága. (2: public, 1: friendOnly, 0: private ) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Táblanév** |  |  |  |  | **Komment** |
| trips | | |  |  | Trips tábla, ez tárolja a Tripek adatait. |
| **Mezőnév** | **Típus** | **Kulcsok** | **Null-e** | **Alapérték** | **Komment** |
| id | number | primary | not null | ai | A Trip azonosítója. |
| journey\_id | number | foreign | not null | 0 | A Journey azonosítója, amibe a Trip tartozik. |
| serial\_number | number |  | not null | 0 | Ez Határozza meg a Tripek sorrendjét. |
| end\_location | varchar(20) |  | not null | „N.A.” | A Trip célállomása. |
| end\_date | date |  |  | NULL | A Trip befejezésének dátuma. Ez mutatja meg, ha egy Journey aktív, mert akkor az utolsó end\_date = NULL. |
| visibility | number |  | not null | 1 | A Trip láthatósága. (2: public, 1: friendOnly, 0: private ) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Táblanév** |  |  |  |  | **Komment** |
| entries | | |  |  | Entry tábla, itt vannak az entry-k eltárolva. |
| **Mezőnév** | **Típus** | **Kulcsok** | **Null-e** | **Alapérték** | **Komment** |
| id | number | primary | not null | ai | Az Entry azonosítója. |
| trip\_id | number | foreign | not null | 0 | A Trip azonosítója, amibe az Entry tartozik. |
| entry\_date | date |  |  | sysdate | Az Entry létrehozásának a dátuma. |
| picture | blob |  |  | NULL | Az Entry-hez csatolt kép. |
| comment | varchar(1024) |  | not null | „N.A” | Az Entryhez tartozó szöveges memo. |
| visibility | number |  | not null | 1 | Az Entry láthatósága. (2: public, 1: friendOnly, 0: private ) |
| title | varchar(50) |  | not null | „N.A” | Az Entry címe. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Táblanév** |  |  |  |  | **Komment** |
| friends | | |  |  | A barátságok kapcsolatos adatok találhatóak itt. |
| **Mezőnév** | **Típus** | **Kulcsok** | **Null-e** | **Alapérték** | **Komment** |
| sender\_id | number | primary, foreign1 | not null |  | A barátkérelem küldőjének azonosítója |
| getter\_id | number | primary, foreign2 | not null |  | A barátkérelem célpontjának azonosítója. |
| accepted | number |  | not null | 0 | Ez mutatja, hogy a barátság elfogadásra került-e. (0: pending, 1: accepted) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Táblanév** |  |  |  |  | **Komment** |
| journey\_participants | | |  |  | Itt történik egy adott Journeyhez meghívott társak eltárolása. |
| **Mezőnév** | **Típus** | **Kulcsok** | **Null-e** | **Alapérték** | **Komment** |
| user\_id | number | primary, foreign1 | not null |  | A Journey meghívás célpontjának az azonosítója |
| jouney\_id | number | primary, foreign2 | not null |  | A Journey azonosítója. |
| accepted | number |  | not null | 0 | Ez mutatja, hogy a meghívás elfogadásra került-e. (0: pending, 1: accepted) Journey csak akkor indul el, ha nem marad nullás meghívás. |

# 5. Teszjegyzőkönyv

\*\*\*\*ide jön a tesztjegyzőkönyv

# 6. Gondolatok

## Saját véleményem

Sajnálom, hogy asztali verziót készítettem, viszont nem állt rendelkezésemre elegendő idő, hogy egy mobilapplikációt készítsek. Jelenleg ott tartok, hogy szeretném átültetni androidra ezt az egész alkalmazást, viszont ahhoz még rengeteget kell képeznem magamat.

Mivel szerettem volna a legtöbb dolgot saját kútfőből megcsinálni, gyakorlatilag a teljes adatbázis kapcsolódáshoz egy saját módszert kellett kidolgoznom. Tudom tele van valószínűleg elvi hibákkal, ettől függetlenül ennek a résznek a készítését élveztem a legjobban. Jó volt látni, ahogy az adatbázisból az általam elvárt adatok jönnek vissza. A másik nagy kedvenc részem a Flow controlok megteremtése volt. Amikor elkezdtem velük dolgozni nem voltam benne biztos, hogy ez az egész működőképes lesz, de végül valahogy összeállt az egész.

Azt sajnálom még nagyon, hogy a MainForm valamint FlowElementProvider osztályom túl bonyolult maradt. Gondolkodtam rajta, hogy több osztályra szét kellene osztani a funkciókat, azonban erre nem maradt elég időm.

Végül többször is jó döntésnek találtam, hogy az egész projektemhez GIT-et használtam (GitHub). Voltak olyan helyzetek, amikor megmentett pár agyvérzéstől és vissza tudtam állítani a projektet egy stabil verzióra.

Úgy vagyok a DT-vel, hogy tudom iszonyatosan lassú, túl bonyolult, de minden hibája ellenére azért szeretem.

## Továbbfejlesztési ötletek

* Regisztráció törlése
* Magyarország városainak feltöltése és kiválasztási lehetősége célpontnak
* Véletlenszerű utazások készítése
* Kommenttelési és lájkolási lehetőség az Entry-knél.
* Valamely térképes applikáció beépítése az alkalmazásba, hogy lehessen útvonalat nézni.

# 7. Felhasznált irodalom

\*\*\*\*\* ide jön az irodalom

*Titkosítás*:

* <https://www.youtube.com/watch?v=da5GSeJV9qE>
* http://curlybrackets.com/posts/43017/how-to-encrypt-and-decrypt-a-string-in-c-sharp

*Chrome megnyitása*:

* <https://stackoverflow.com/questions/58024/open-a-url-from-windows-forms>

Tartalom

[1. Témamegjelölés és indolkás 1](#_Toc51170314)

[2. DiceTravel Home felhasználói kézikönyv 1](#_Toc51170315)

[Bevezető 1](#_Toc51170316)

[Utazásnaplózás 2](#_Toc51170317)

[Közösségi oldal 2](#_Toc51170318)

[Utazási meghatározott vagy véletlenszerű állomásokra. 2](#_Toc51170319)

[Mi a DiceTravel? 2](#_Toc51170320)

[Kulcsszavak 2](#_Toc51170321)

[A DiceTravel Home telepítése 4](#_Toc51170322)

[A DiceTravel Home indítása 5](#_Toc51170323)

[A DiceTravel Home felépítése 5](#_Toc51170324)

[Bejelentkezés ablak 5](#_Toc51170325)

[Regisztrációs ablak 5](#_Toc51170326)

[Főablak 6](#_Toc51170327)

[A DiceTravel Home funkciói 8](#_Toc51170328)

[Új Utazás indítása 8](#_Toc51170329)

[Aktív Utazás törlése 10](#_Toc51170330)

[Aktív Utazás módosítása 10](#_Toc51170331)

[Szakasz befejezése 10](#_Toc51170332)

[Szakasz módosítása 11](#_Toc51170333)

[Bejegyzés készítése 11](#_Toc51170334)

[Keresés 12](#_Toc51170335)

[Ismerős jelölése/jelölés elfogadása 12](#_Toc51170336)

[A Folyamablak 12](#_Toc51170337)

[Folyam elemek 13](#_Toc51170338)

[Folyamok 15](#_Toc51170339)

[3. Fejlesztői kézikönyv 17](#_Toc51170340)

[A DiceTravel koncepciója 17](#_Toc51170341)

[Mi is a DiceTravel? 17](#_Toc51170342)

[Kulcsszavak 18](#_Toc51170343)

[A DiceTravel felépítése 18](#_Toc51170344)

[a GUI terve 18](#_Toc51170345)

[Osztálytípusok 19](#_Toc51170346)

[Az osztályok részletezése 20](#_Toc51170347)

[Program osztály 20](#_Toc51170348)

[MainForm osztály 21](#_Toc51170349)

[Entitások 21](#_Toc51170350)

[Formok 23](#_Toc51170351)

[Kontrollok 26](#_Toc51170352)

[Kiegészítő osztályok 28](#_Toc51170353)

[ActiveUserStore 28](#_Toc51170354)

[Encryptor 28](#_Toc51170355)

[FlowElementProvider 29](#_Toc51170356)

[ImageHandler 30](#_Toc51170357)

[4. Adatbázisterv 31](#_Toc51170358)

[Az adatbázis grafikus ábrája 31](#_Toc51170359)

[Az adatbázis tábláinak részletes leírása 31](#_Toc51170360)

[5. Teszjegyzőkönyv 33](#_Toc51170361)

[6. Gondolatok 33](#_Toc51170362)

[Saját véleményem 33](#_Toc51170363)

[Továbbfejlesztési ötletek: 33](#_Toc51170364)

[7. Felhasznált irodalom 34](#_Toc51170365)