Autószervíz adminisztrációs szoftver

Trello board elérhetősége: <https://trello.com/b/g7Ugd2Hm/project-retek-carmechanic>

GitHub projekt elérhetősége: <https://github.com/krovka93/ReTek-CarMechanic>

# Projektben résztvevő fejlesztők:

* Varga Eszter – BusinessLayer réteg (adatbáizis <-> szoftver kapcsolata), Adatbázistáblák megtervezése, View és ViewModel
* Pozsonyi Bence - BusinessLayer réteg (adatbáizis <-> szoftver kapcsolata), Adatbázis triggerek, sequencerek fejlesztése
* Bertók Bence – GUI és ViewModel fejlesztés
* Kovács Levente – Technical Lead feladatok, Magas szintű specifikáció, View, ViewModel, BusinessLayer fejlesztés

# Alkalmazás felépítése

-> Taskok szétbontása rétegenként:

* View
* ViewModel
* Modellek +BusinessLayer
* DataBase

## Adatbázis

Táblák (tartalomhoz nézd a szoftver és a „Továbbfejlesztéshez feladatok (Haladó DBMS) beadandó” részt):

* Ügyfelek
* Gépjárművek
* Szolgáltatások
* Alkatrészek
* Munkalapok

## Szoftver

* View
  + Főablak, 6 gombbal, amik újabb 5 különböző ablakra navigálnak
    - 1 gomb – Ügyfelek – külön ablak
      * névvel, születési hellyel és idővel, TAJ számmal, adóazonosító jellel, a rögzítés idejével (ezt majd csak az SQL insertbe beleírjuk)
    - 2 gomb – Gépjárművek – külön ablak
      * Legördülő menüből kiválaszthatjuk a tulajdonos nevét (ID alapján Primary KEY lesz a táblában), rendszámát, gyártóját, típusát, gyártás évét és alvázszámát
    - 3 gomb – Szolgáltatások – külön ablak
      * Szolgáltatás neve, ára
    - 4 gomb – Alkatrészek
      * Alkatrész neve, márkája, ára, készleten lévő mennyiségek
    - 5 gomb – Munkalap generálás
      * Itt egy legördülőből kiválaszthatjuk a tulajdonost (ez alapján ugye ellenőrizzük, hogy van kocsija, ami be van regisztrálva hozzá), hozzáadhatjuk az igényelt szolgáltatást meg az igényelt alkatrészt.
    - 6 gomb – Adattáblák frissítése manuálisan
* ViewModel
  + ÜgyfelekViewModel
    - Bindinggal összekötni a textboxokat (név, születési hely, dátum, tajszám, adóazonosító jel)
    - UploadData() függvény előbb megvizsgálja, hogy az adott név és születési hellyel rendelkező ügyfél már az adatbázisban van-e. Ha igen akkor feldob egy ablakot, hogy már létezik és a friss adatokkal szeretnénk-e módosítani a jelenlegit. Ha igenre kattint akkor BusinessLayer-ben meghívjuk az UpdateExistingUsert()-t. Ha nem létezik még akkor UploadNewUser() metódus meghívása.
  + GépjárművekViewModel
    - Ugyan az, mint az ügyfelek kivéve, hogy itt lesz egy List<Ügyfél> lista ami tartalmazni fogja az adott ügyfeleket, ez alapján tudjuk majd kilistázni a legördülő menübe az embereket. Ehhez szintén kell egy public Ügyfél SelectedÜgyfél{get;set;} property ahol a kiválasztott ügyfelet tároljuk le.
  + SzolgáltatásokViewModel
    - * 3 property Szolgáltatás neve, ára
      * UploadSzolgáltatás() metódus. Ha van akkor hibaüzenet (név alapján keresés), ha nincs akkor update (mivel az ára frissülhet).
  + AlkatrészekViewModel
    - Ugyan úgy, mint a szolgáltatásoknál, csak itt még a mennyiséget is lehet módosítani.
  + MunkalapgenerálásViewModel
    - 3 dátum property – Kezdet, várható vég, tényleges vég
    - 1 szám property – KM óra állása
    - 1 azonosító property (getterben elmegyünk a BusinessLayerhez -> GetAzonosítóForMunkalap()

2 legördülő -> Szolgáltatás kiválasztása, Alkatrész kiválasztása (itt vizsgálni kell, hogy az igényelt alkatrész raktáron van-e. Ha igen akkor feltöltésnél triggerrel le kell vonni a mennyiségből)

* Model – BusinessLayer
  + Modellek a következőkhöz -> Ügyfelek, Gépjárművek, Szolgáltatások, Alkatrészek,Munkalap (a fenti propertyk alapján)
  + Interface megírása mindegyikhez külön (pl. IÜgyfelek ami tartalmazza a függvényeg headerjét -> UpdateExistingUsert, UploadNewUser stb)
  + Az interfaceket implementálni kell a BusinessLayerbe és a már meglévő adatbázis kapcsolat segítségével megírni az adatbázis műveleteket

# Továbbfejlesztéshez feladatok (Haladó DBMS)

Szervíz: Készítsen ügyviteli szoftvert egy autók javításával foglalkozó vállalat részére. A feladat megoldása során, ahol lehetséges PL/SQL tárolt alprogramokat használjon! A megvalósított szoftvernek az alábbi feltételeknek kell eleget tennie:

* A szervízben az ügyfelek kerülnek előzetesen rögzítésre névvel, születési hellyel és idővel, TAJ számmal, adóazonosító jellel, a rögzítés idejével. A rögzítés/módosítás során az alábbi feladatok kell ellátni:
  + alapvető ellenőrzések (nem lehet null, üres szöveg, az aktuális dátumhoz képest a születési idő)
  + TAJ szám ellenőrzése (ellenőrző algoritmus segítségével)
  + adóazonosító jel ellenőrzése (ellenőrző algoritmus segítségével)
  + a rögzítés során minden ügyfélhez generálunk egy egyedi azonosítót az alábbiak szerint: U betű, majd aktuális dátum ÉÉÉÉMMDD formátumban, majd az adott napon rögzített ügyfél sorszáma (a rögzítés ideje mezőből következik). Pl.: U-20191212-10 a 2019. 12. 12-én rögzített 10. ügyfél azonosítója.
* A rögzített ügyfelekhez tudjuk beregisztrálni a járműveket. Minden járműről eltároljuk a rendszámát, gyártóját, típusát, gyártás évét és alvázszámát. A rögzítés során az alábbi feladatokat kell ellátni:
  + alapvető ellenőrzések
  + alvázszám ellenőrzés (ellenőrző algoritmus)
  + rendszám (formai ellenőrzés)
* A rögzített járművekhez munkalapokat tud készíteni a szerviz, egy-egy szerviztevékenység kapcsán. A munkalap tartalmazza a jármű javításának kezdetét, várható végét, tényleges végét és a javítás kezdetén a kilométeróra állást. Minden egyes munkalaphoz a fenti elv szerint hasonló azonosítót generálunk azzal a különbséggel, hogy ez az azonosító M betűvel fog kezdődni. A munkalapra ezt követően a javítás során felmerülő költségeket tételesen felsorolják (alkatrészek és szolgáltatások) egységárral, mennyiséggel és végösszeggel. A szolgáltatások és alkatrészek külön forrástáblából kerülnek betöltésre és trigger segítségével biztosítsa, hogy az alkatrészek készletfogyása megtörténjen.

Készítsen C#/Java programozási nyelven kliensalkalmazást az adatbázis kezeléséhez. Ügyeljen arra, hogy az adatbázis csak előzetes bejelentkezést követően lehessen kezelhető.