Verebély László Technikumi képzésének, RoboCop vizsgaprojektének dokumentációja.

Tartalom

Bevezetés

A projektmunka az iskola részéről került bele a választható projektek halmazába. Alapvetően a rendőrség által használt Robotzsaru csökkentett képességű változatának az elkészítése volt a cél.

Az eredeti Robotzsaru alapvető feladatai közé tartozott a:

- A rendőri tevékenység egyre nagyobb részének lefedése
- Adatok a parancsnoki döntésekhez
- Kimutatások
- Információs adatbázis

Tehát egy integrált ügyviteli, ügyfeldolgozó és elektronikus iratkezelő rendszer. Az eredeti programa teljes ügyvitel végrehajtását lehetővé teszi. Úgymint ügykarbantartás, iratrögzítés, Események, személyek, tárgyak összekapcsolása. Az eredeti program nem csak adatkezelésre hanem szolgálatvezénylésre, közigazgatásibírságok kiszabására, feladatok koordinálására és nem utolsó sorban szűrések, statisztikák, listák készítésére, egyszóval a a rendőrség munkájának teljes körű lefedésére készült.

Az általunk fejlesztett alkalmazás oktatási céllal készült, a diákok elméleti oktatásának a megtámogatására, segítésére, hogy az elméleti anyag elsajátítása után a gyakorlatban is kipróbálhassák azt.

A rendőrség által használt Robotzsaru.

Projekt Menedzsment

Előzmények

Első körben a jelenlegitől teljesen eltérő technológiát találtunk ki a projekthez. SQL Server adatbázis, Python Flask WebAPI backend és React.JS frontend lett volna a technológia. Amikor elkezdtük mindenféle projektmenedzsment terv nélkül indultunk neki. A fejlesztés database first szemlélettel indult, rá építve az API-t, amivel a frontend pedig GET és POST HTTP hívásokkal kommunikált. A funkcionalitás egy egyszerű email cím, plusz jelszavas regisztráció, valamint bejelentkezés. Azonosításhoz token generálás, ennek sütibe és adatbázisba való mentés volt elkészítve.

Felmerülő Problémák

A korábban kiválasztott technológia mentén egyrészről felmerült, hogy nem tudunk megfelelő segítséget kérni a megvalósítást illetően, másrészről csapaton belüli kommunikációs problémák merültek fel, mivel mindkettőnk számára a képzéshez képest teljesen új eszközök elsajátítására lett volna szükség.

Újratervezés

Miután az előbbi problémák meghatározásra kerültek, két lépést visszafele megtéve eldobtuk a korábbi front- és backend technológiát és megtartva az adatbázist, ASP.NET-re váltottunk. Érv emellett az volt, hogy az intézményben oktatott tudást felhasználva és rendszeres konzultáció mellett hatékonyabban fogunk tudni haladni a fejlesztéssel.

Agilis metodika

További csapatösszhang problémákkal küszködve, egy konzultációt követően megállapodtunk abban, hogy érdemes lenne Agilis és Scrum szemlélettel és elemekkel, kisebb részletekben haladni. Ehhez meghatározásra került az Ügyfél és a Product Owner szerep, valamint elkészült egy product backlog. A backlog 10 db, priorizált pontból került összeállításra, 2 hetes sprintre lett tervezve. Felosztásra kerültek a fejlesztési pontok. Az első iteráció kitűzött funkcionalitása a korábban lefejlesztett, de az új technológiára átírva.

Továbbfejleszthetőség

A kitűzött, 10 pontos backlogból végül több-kevesebb sikerrel, 9 db került teljesítésre. Ezen felüli lefejlesztett funkció: egy db input form. Az eredetileg kitűzött, teljes ügyviteli rendszer

megalkotása nem sikerült. Továbbfejlesztés esetén a második iterációban, az eddigi tapasztalatok alapján, a jelenleg meglévő input formon keresztül rögzített adatokon való keresési funkció lefejlesztése lenne a cél. Product Ownerünkkel való egyeztetés ennek sikeres teljesítése esetén esedékes.

Visszatekintés

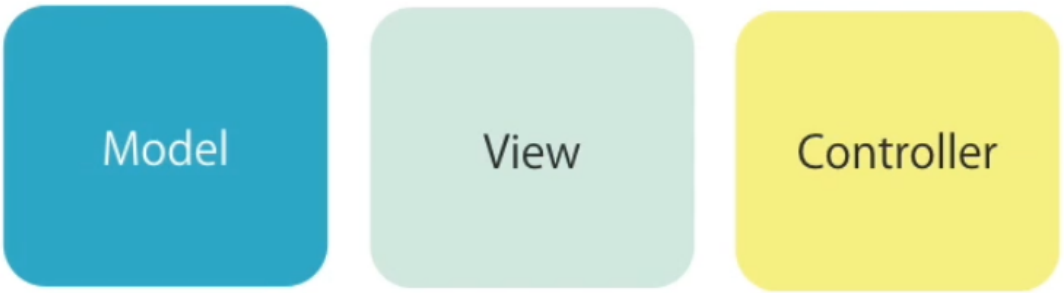
A projekt tapasztalata mentén meghatározható, legfontosabb fejlesztendő terület: kommunikáció. Az új technológia hátrányait mérlegelve, korábban kellett volna a meglévő, biztosabb úton járni. Emellett a szakmában is széles körben használt projekt menedzsment elemeket az elejétől alkalmazva, a kezdeti nehézségeken felülkerekedve, valószínűleg 2-3, sikeresebb iterációval tartana előrébb a termék.

Technológia

Az alkalmazást ASP.Net MVC applikácó létrehozásásával kívántuk megvalósítani, kihasználva annak minden rendelkezésre álló előnyét. Úgymint EntityFramework, Razor syntax és az implementálható eszközök sokaságát. Úgymint Bootstrap, JQuery. Ezen eszközök használatával. korszerű, gyors és könnyen karbantartható programot lehet létrehozni.

Az MVC Architektúra

Lehetővé teszi, hogy külön komponensekre bontva építsük fel az alkalmazást. Így a különböző működésű elemeket szétbontva, egységekben lehet kezelni. Az architektúra három főbb komponensekből áll: Modelből, a View-ból és a Controller-ből.



MVC részei

Model

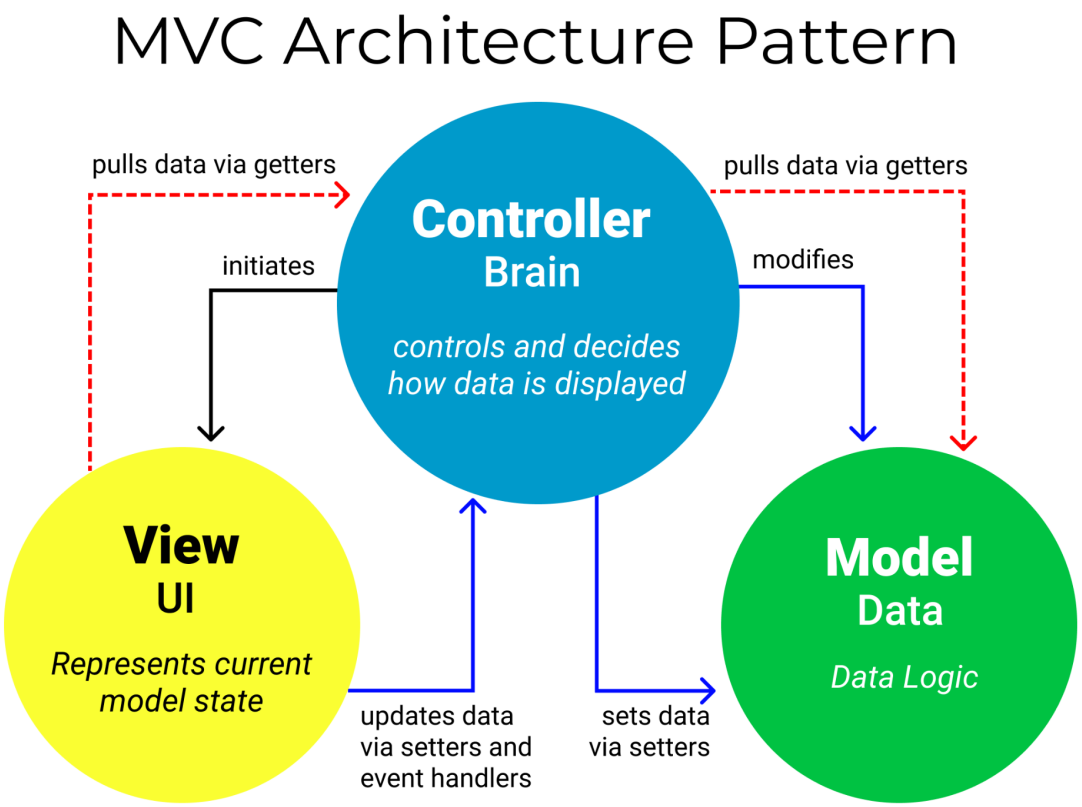
A Model tulajdonságokat és metódusokat tartalmazhat amik egyértelműen meghatározzák a modell állapotát és a rá vonatkozó szabályokat. Mivel nincsenek közvetlenül a User Interface-hez kapcsolva így más applikációkban is használhatóak.

View

A View-k felelősek az alkalmazás adatmegjelenítésért és segítségével kezelhető, feldolgozható minden felhasználói interakció. A View vagy nézet lényegében HTML kód mely egy Razor jelöléssel rendelkezik aminek segítségével .Net alapú kód ágyazható be a weboldalakba.

Controller

A Controller felelős a HTTP requestek kezeléséért. Ha kérés jön a weboldal felől vagy kérést intézünk a weboldal felé. Ilyenkor a megfelelő Controller kezelei a kérést és a megírt kódnak megfelelően műveletek hajt végre az adatbázisban vagy adatokat szolgáltat a View-k számára. A megfelelő Controller kiválasztásáért a Router felelős.



Teljes MVC Architektúra

Entity Framework

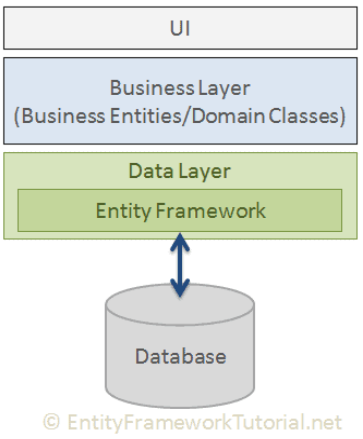
Az Entity Framework tulajdonképpen egy Object-Relational Mapper (ORM) aminek segítségével hozzáférünk az adatbázishoz anélkül, hogy az SQL query-ket kellene írnunk. A DbContext lényegében az adatbázisunk leképzése. A DbContext-ből adatokat LINQ segítségével tudunk kinyerni úgy, hogy az Entity Framework lefordítja a LINQ utasításokat SQL nyelvre.

Az Entity Framework használatának két módja létezik:

- Database First
- Code First

A Database First megközelítés használatával a már meglévő adatbázist alapján az EF legenerálja az applikációnk számára a modelleket, domain osztályokat. Az adatbázishoz tartozó view és tárolt eljárások is elérhetővé válnak.

A Code First megközelítés alkalmazásánál a domain osztályok és a migrációk létrehozása után az EF le generálja az adatbázis tábláit.



Entity Framework felépítése

Bootstrap

Az alkalmazás reszponzívvá tételét és egységes kinézet megalkotását a Twitter által fejlesztett CSS keretrendszerrel oldottuk meg. A Nuget Package Manageren keresztül feltelepíthető a legújabb,teljes Bootstrap 5.0 csomag.

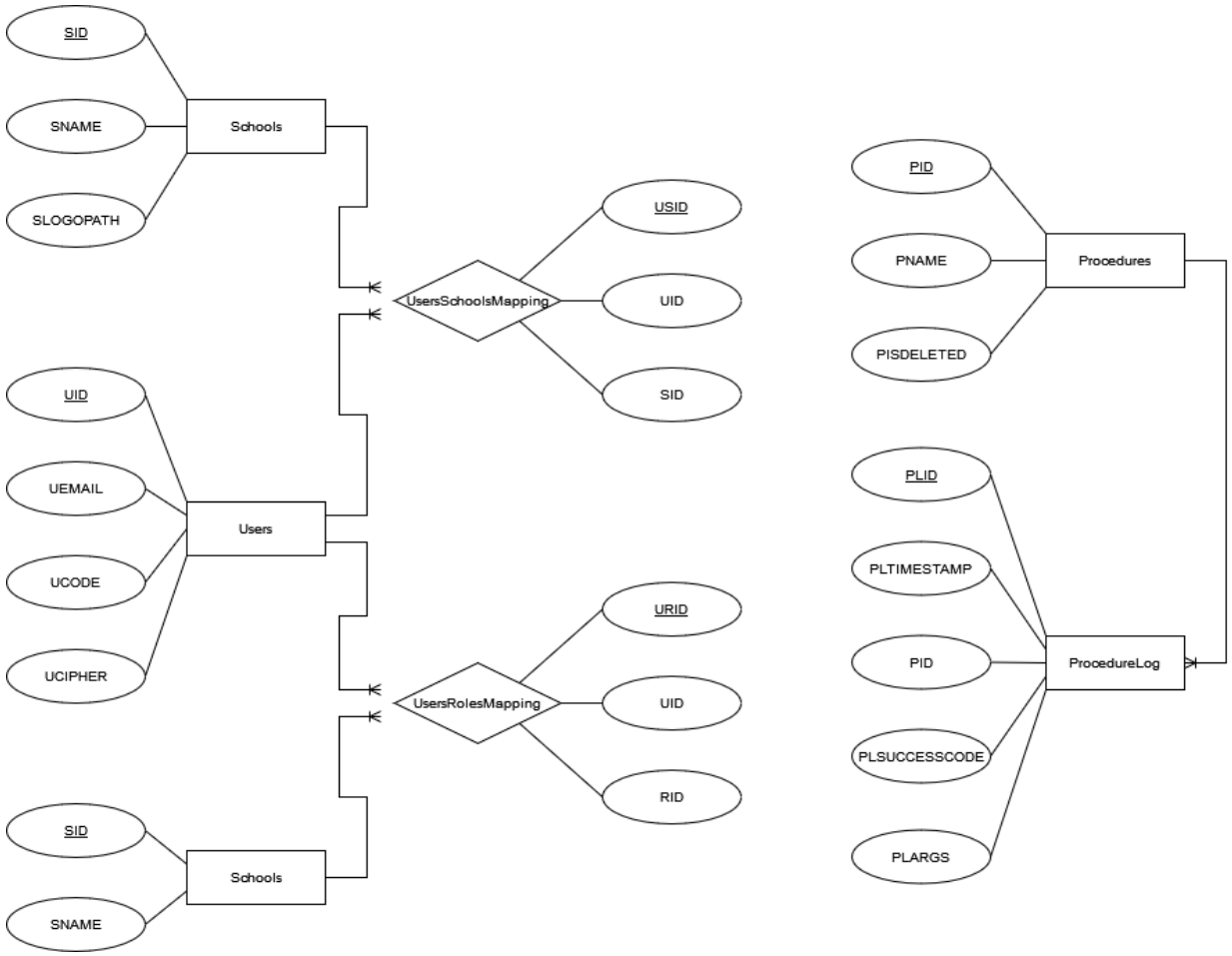
Adatbázis

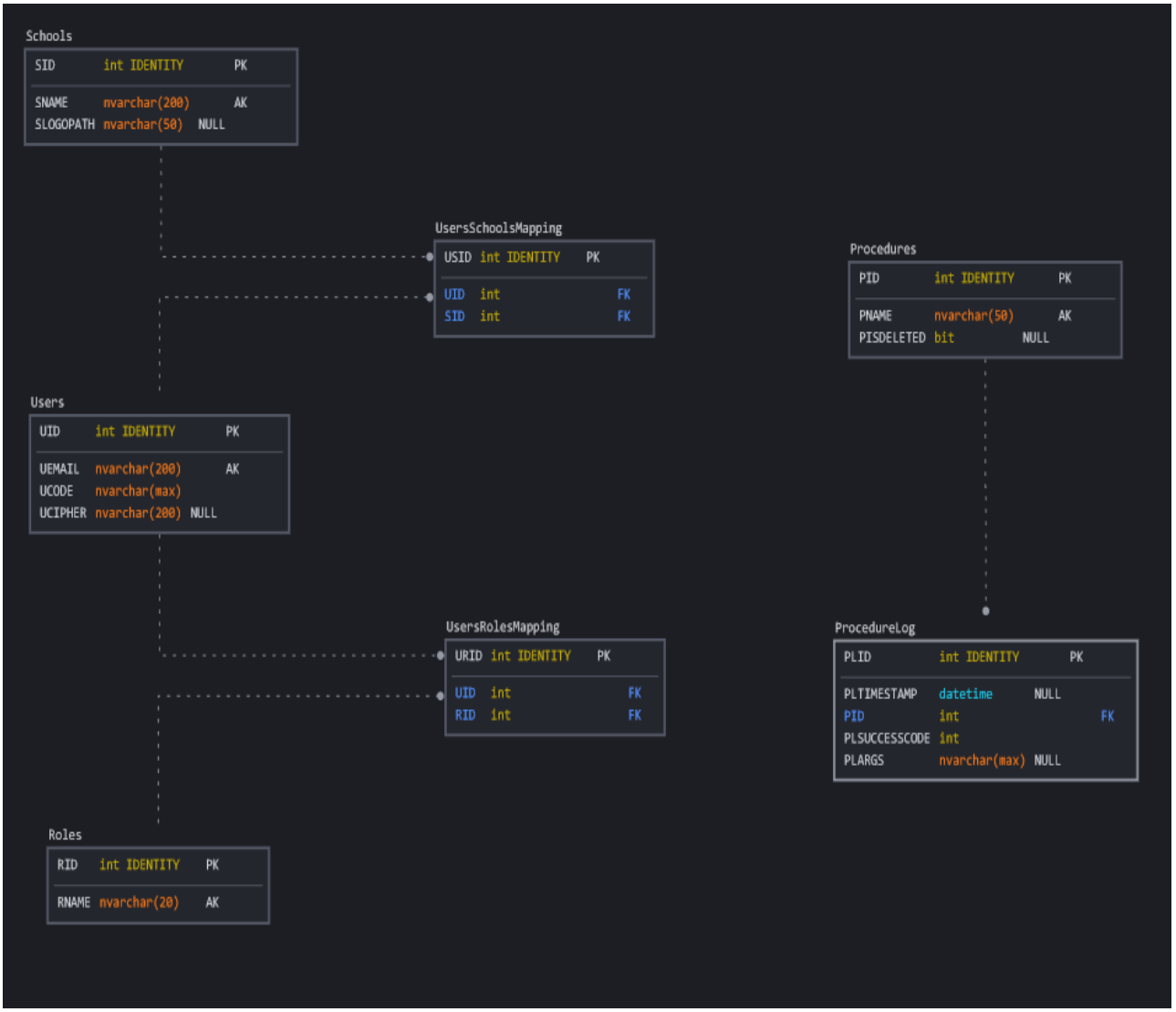
A megtervezett adatbázis, RoboCop v0.0.0.0.1, szakdolgozatunk első iterációjához készült. A projekt maga, a 'RobotZsaru' elnevezésű, rendőrségi iktatóprogram, rendészeti iskolák által is használható implementációjáról szól. Az első iterációban kitűzött funkcionalitási cél: regisztráció, belépés. Technológiának webes (React.js) frontendet, REST API (Python) backendet, és Microsoft SQL Server adatbázist választottunk.

Az adatbázisban jelen állapotban a következő táblák találhatók:

- 'Users' – ebben tároljuk a felhasználókat
- 'Schools' – ebben tároljuk az iskolákat
- 'Roles' – a jogosultságokat hivatott tárolni
- 'UsersRolesMapping' – tárolja az adott felhasználók jogosultságait
- 'UsersSchoolsMapping' – tárolja az adott felhasználók iskoláit
- 'Procedures' – a definiált tárolt eljárások neveit tároljuk itt
- 'ProcedureLog' – itt kerülnek naplózásra a tárolt eljárások hívásai

A táblák mezőit, kulcsait, megszorításait, egymáshoz való kapcsolatait az alábbi ER diagram és relációs adatmodell tartalmazza:





A backend-en futó API-nak, az adatbázissal való kommunikációjához a következő adatbázis objektumokat hoztuk létre (SP: tárolt eljárás; FUNC: függvény):

- ’RegisterUser’ (SP) – az input email és jelszó paraméterekre rögzíti a felhasználót az adatbázisba, illetve output paraméterként visszaadja, hogy sikeresen lefutott-e.
- ’SetUserCipher’ (SP) – az input email és token paraméterekre rögzíti az emailhez tartozó tokent, illetve output paraméterként visszaadja, hogy sikeresen lefutott-e.
- ’LoggedIn’ (FUNC) – az input email és jelszó paraméterekre visszaadja, hogy létezik-e az adatbázisban a megadott páros

- ’Authorized’ (FUNC) – az input tokenre visszaadja, hogy létezik-e az adatbázisban.
- Egyéb funkcionalitást biztosító objektumok:

- ’SchoolsAndRoles’ (VIEW) – a projekt első iterációjában ez lesz a teszt visszatérési érték sikeres azonosítás esetén
- ’LogProcedureCall’ (SP) – ez felel az eljárás hívások naplózásáért; input paramétereknek az eljárás nevét, a futás végén visszaadott success kódját, valamint az eljárásnak átadott paramétereket ’;’-vel összefűzve kapja meg
- ’GetUserIdByCipher’ (FUNC) – az input tokenre visszaadja a felhasználó id-ját
  ’GetUserRoleIdByCipher’ (FUNC) – az input tokenre visszaadja a felhasználó jogosultságát
- ’GetUserSchoolIdByCipher’ (FUNC) – az input tokenre visszaadja a felhasználó iskolájának id-ját

Az adatbázis generálásához a következő, előre megírt script az irányadó. Futtatni az SQL Server szintaktikájának megfelelően részletekben kell.

DB_alap_create_scri pts.sql

A teljes funkcionalitás teszteléséhez előre feltölteni szükséges a ’Schools’, a ’Roles’, a ’UsersSchoolsMapping’, és a ’UsersRolesMapping’ a táblákat.

Komponensek technikai leírása

Model-ek

Mivel a választásunk az Database First megközelítésre esett, így az Entity Framework ADO.Net Entity Data Model struktúrát azadatbázis alapján maga hozza létre a RoboCop.edmx filet leképezve a teljes adatbázist. A EF generált modellekhez A CaseRiport Modelt és a Navbar Modelt hoztuk létre.

Robocop.edmx

Definiálja a RoboCop adatbázis sémáit.

CaseRiport Model

A CaseRiport Model felelős a riportok elkészítéséhez szükséges adatok szolgáltatásáért (választható, előre meghatározott adatok, pl. hajszín, város).

Navbar Model

A Navbar Model szolgáltatja az menük listáját és azok elérési útvonalát melyek segítségével lehetővé válik a menük megjelenítése és rootolása.

Controller-ek

HomeController

- Index Ation van beállítva alapértelmezett kezdő actionnek a Route Configban. Az oldal kezdő felületének megjelenítéséért felelős.

AccountController

Ez a Controller szolgáltatja a belépéshez szükséges Action-okat.

- Login Action felelős a felhasználók beléptetésért. A beléptetés során ellenőrzi a kódot, az email címet és a süti érvényességét. Ha létezik a felhasználó akkor a CipherMaker függvény előállít egy új érvényes sütit. A süti ellenőrzését a ValidateToken függvény végzi. Az Action a sütit ezután átadja a böngészőnek.
- Register Action végzi az új felhasználó regisztrálását. A megadott belépési kódot a CreateMD5Hash függvény titkosítja és a titkosított kód kerül bele az adatbázisba.

CaseRiportController

- Riport Action szolgáltatja a szűréshez szükséges alapadatokat, lekérdezi és átadja a CaseRiport View-nak megjelenítésre.
- SaveRiport Action A CaseRiport View által begyűjtött és validált adatokat rögzíti és elküldi az adatbázisnak. Később ezek az adatok könnyen szűrhetőek.

NavController

- Navbar Action adja vissza a Navbar View-nak a menük neveit és útvonalát a UserMenus függvény segítségével.

View-k

Felelnek a vizuális megjelenítésért és létesítenek kapcsolatot a felhasználó és az applikáció és az adatbázis között.

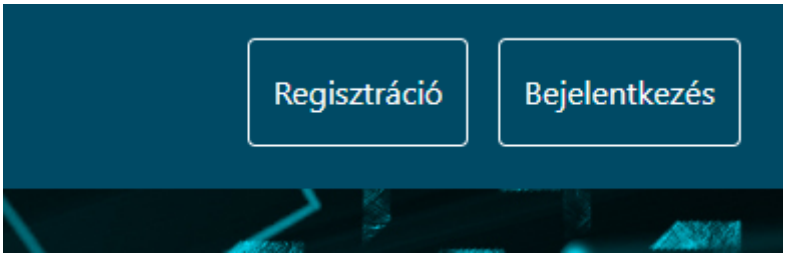
- Home View a kezdőlap informálja a felhasználót a lehetőségeiről.
- _Layout View tartalmazza az oldal gerincét, a meta adatokat, a head-et és a body-t. Itt találhatóak a főbb komponensek is, úgy mint a Navbar, a RenderBody függvény

által renderelt aktuális tartalom és a footer valamint az oldal futtatásához szükséges scriptek például a bootstrap és a jquery eléréséhez.

- Login View a felhasználó beléptetéséért valamint a hibás belépésnél a hibák kiírásáért felelős.
- Register View a felhasználó regisztrációját és hibás regisztráció esetén a hibák megjelenítését végzi.
- Navbar Partial View az oldal tetején fixen elhelyezett navigációs terület. A user belépése után a különböző engedélyezési szintek szerint jeleníti meg a müket.
- Riport View elemeinek kitöltésével lehet megtenni a bejelentéseket.

Komponensek használata

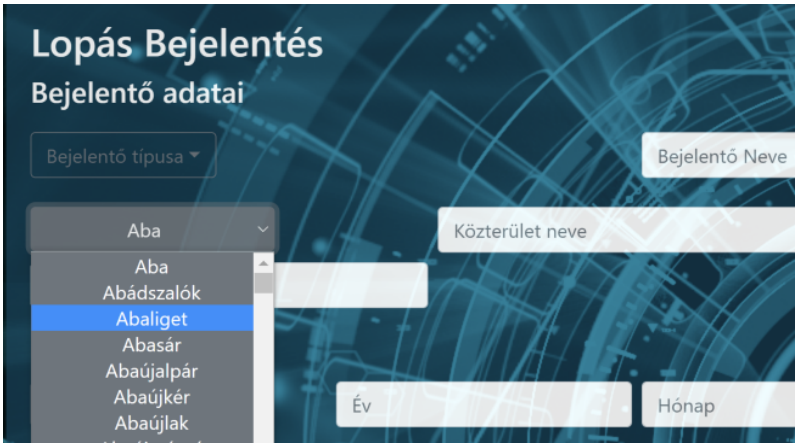
A főoldalon lehetősége van a felhasználónak belépni az alkalmazásba vagy regisztrációt létrehozni.



A regisztrációs oldalra navigálva a “szokásos” regisztrációs folyamatot kell végrehajtani. Meg kell adni egy érvényes email címet és egy saját magunk által választott belépési kódot.

A regisztráció után automatikusan a Belépés oldalra kerül a felhasználó, ahol a korábban megadott email cím és belépési kód kettőssel tud belépni.

A belépés után az új szűrés gombra kattintva megjelenik a lopás bejelentése oldal ahol a megfelelő comboboxok tartalmának kiválasztásával és a textboxok kitöltésével mentésre kerül a lopás bejelentése.


Lopás Bejelentés
Bejelentő adatai
Bejelentő típusa
Aba
Aba
Abádszalók
Abaliget
Abasár
Abaújalpár
Abaújkér
Abaújlak
Bejelentő Neve
Közterület neve
Év
Hónap