# Kérdőívek kiértékelése diagramok segítségével

**Szerzők:**
**Demeter Imola-Krisztina**
**Kapás Krisztina**
**Kapás Orsolya**
**Olteán Péter Boróka**
**Szász Renáta**

**Vezető tanár:**
**Dr. Szántó Zoltán -egyetemi adjunktus**

**2018**

# Tartalomjegyzék

# Ábrajegyzék



.

# 1. Bevezető

Napjainkban minden kutatás eredményéről kimutatásokat készítenek. Számos esetben előfordulhat, hogy valamilyen cél érdekében szükségessé válik adatok bekérése, majd ezek értelmezése. Ilyen esetek lehetnek többek között:

- Elégedettségmérő kérdőívek
- Pedagógiai vagy az oktatott tárgyhoz kapcsolódó kutatások
- Információs adatok begyűjtése a tanulóktól
- Adott kérdéskörben vélemények begyűjtése

A projektünk célja az, hogy lehessen követni egy adott kutatás eredményét.

A pályaválasztás, azon belül a felsőoktatási továbbtanulás fontos döntés a fiatalok életében. A projektünkben kutatást végzünk arról, hogy a középiskolákban a diákok milyen szakon szeretnének tovább tanulni, illetve, hogy melyik egyetemen szeretnék folytatni tanulmányaikat.

Észrevételeink alapján az intézményválasztás során a térségválasztás is meghatározó szempont a diákok számára. Főként azokat a középiskolásokat célozzuk, akik magyarlakta megyékben tanulnak.

A kérdéseket egy online űrlapon közöljük az érdekeltekkel és egy másik űrlapon megjelenítjük ezek kiértékelését, grafikusan, diagramok használatával. Ezáltal például a kérdőíven szereplő egyetemek követhetik azt, hogy milyen szakra van igény.

# 2. Követelmény specifikáció

## 2.1. Felhasználói követelmények

- Anonymus felhasználó:
    - a felhasználó egy Webes felületen megjelenő kérdőívet tölt ki
    - a felület figyelmeztető üzenetet küld, ha a kötelező kérdésekre nem válaszolt a felhasználó
    - bizonyos kérdések csak akkor jelennek meg, ha szükségesek

## 2.2. Rendszer követelmények

- Funkcionális követelmények:
    - Webes felület a kérdésekkel
    - Adatok helyes tárolása
    - Adatok helyes kiértékelése
    - Grafikonok
    - Webes felület a kimutatásokkal

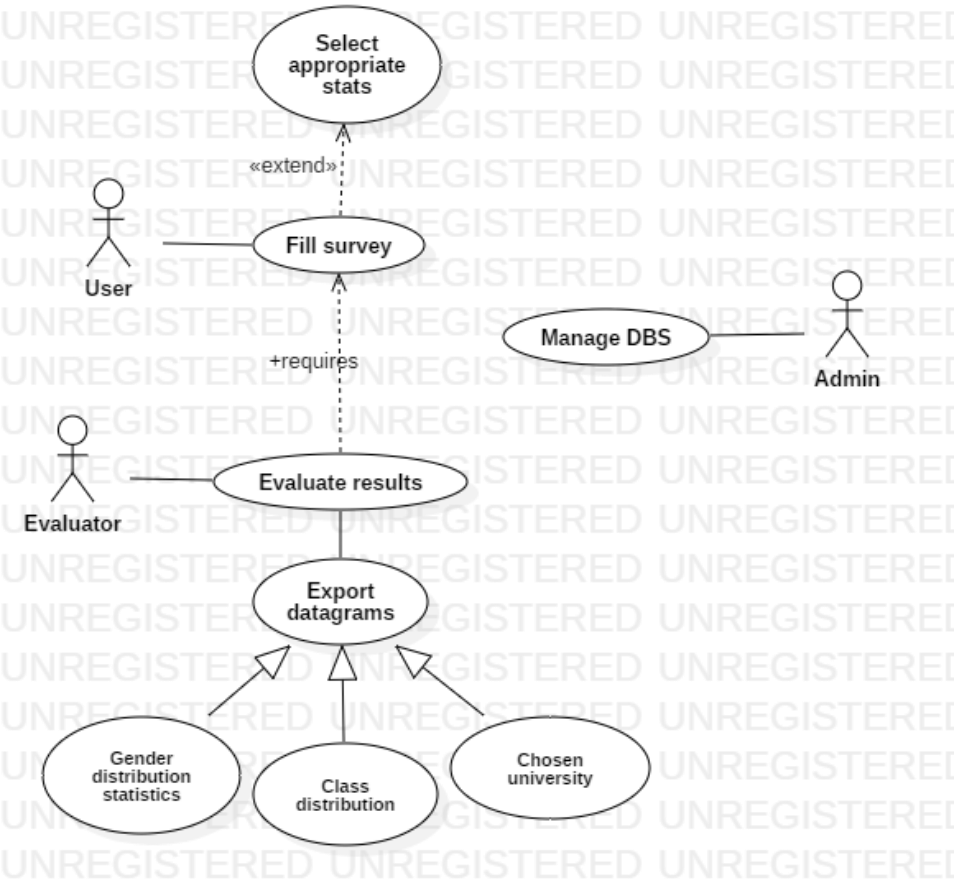
- Nem funkcionális követelmények:
    - Microsoft Visual Studio 2017
    - MongoDB
    - Microsoft SQL Server Management Studio
    - verziókövetés Git segítségével

# 3. Kivitelezés

A kivitelezésünk első fázisában elkészítettük a Use Case Diagramot és a követelmény specifikációt, azért, hogy fogalmazzuk meg, hogy melyek azok a dolgok, amelyeket megszeretnénk valósítani.

## 1. Az alkalmazás modellezése

### a. Use Case Diagram

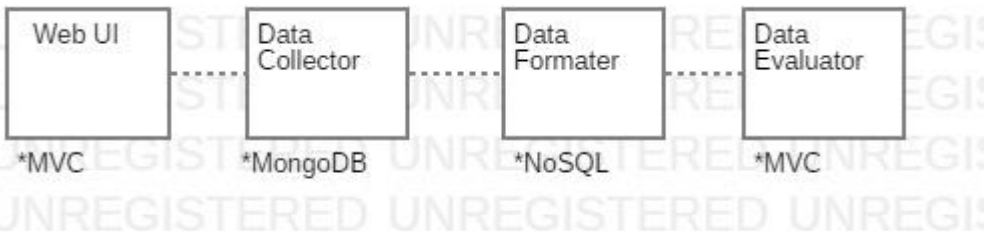



***1.ábra*** *A rendszer Use Case diagramja*

Az **1.ábrán** látható, hogy 3 típusú felhasználót különítettünk el, a “User”, aki kitölti a kérdőívet (több felmért személy), az “Admin”, aki karbantartja a rendszert, valamint az “Evaluator”, aki értelmezi, és felhasználja a kimutatásokat.

A User kitölti a rá vonatkozó kérdéseket, a válaszai bekerülnek egy adatbázisba, az ott szereplő összes válasz értékelődik és különböző diagramok készülnek belőlük, amelyet az “Evaluator” értelmez és felhasznál.

## b. Architekturai váz

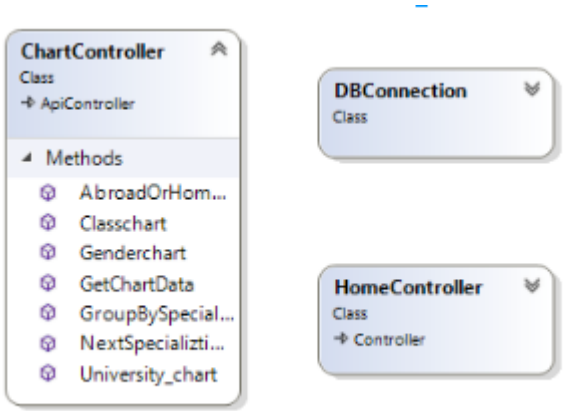



*2.ábra Architektúrai váz*

Az **2.ábrán** látható, hogy milyen részekre bontottuk le a projektet, és ezeket milyen technológiákkal valósítottuk meg. Először elkészítettük a kérdőív felületét, amelyet MVC alkalmazással valósítottunk meg. A kérdőív kitöltőinek válaszait MongoDB-be mentettük, majd átalakítottuk és elküldtük egy relációs adatbázis kezelő rendszerbe, MSSQL-be. A diagramok megjelenítéséhez szintén egy MVC alkalmazást hoztunk létre és a diagramokat JavaScript-ben írtuk meg. Az adatbázisból lekérdezések által kinyertük a diagramokhoz szükséges adatokat.

## c. Osztály diagram

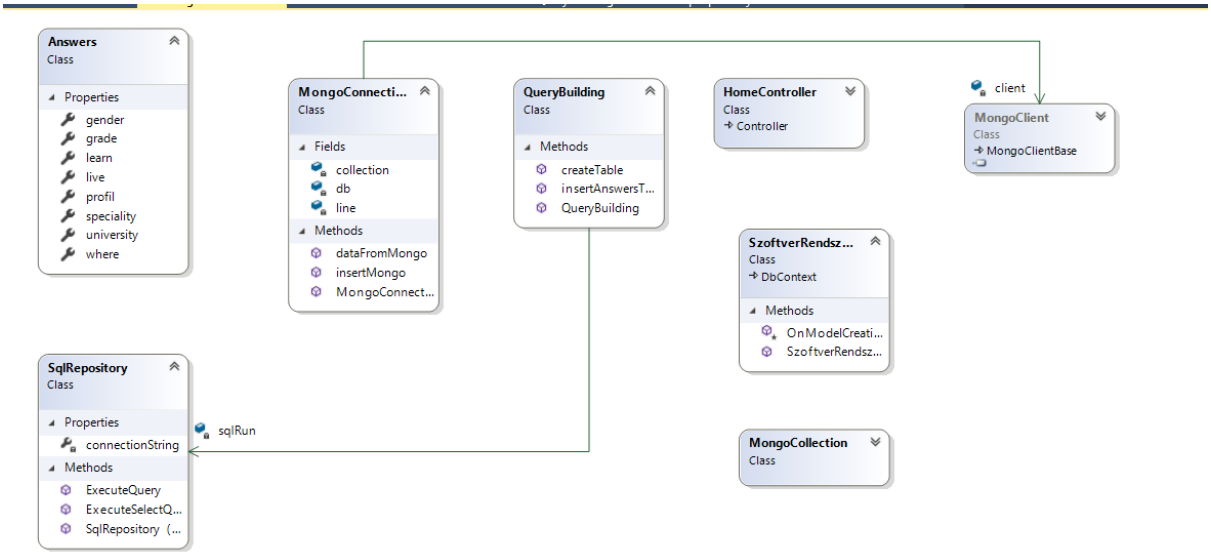
### i. Kimutatások késztése




*3.ábra Statisztika rész osztálydiagramja*

A **3.ábrán** látható osztálydiagramon megtekinthetjük a kimutatásokhoz szükséges komponenseket. A “DBConnection” osztály segítségével létrejön az adatbázis kapcsolat, ez az osztály tartalmazza a kapcsolódáshoz szükséges karakterláncot.

Mivel egy MVC projektben dolgozunk ezért szükségünk van egy Controller-re is amelynek szerepe az, hogy válaszoljon a felhasználói inputra. Az MVC vezérlői az alkalmazás folyamatával, illetve az üzleti logikával törődnek, feldolgozzák a bejövő adatot és kimenő adatot szolgáltatnak a megfelelő View számára. Tehát a “HomeController” metódusa dönti el, hogy melyik View-t adja vissza. A Controller egy-egy request kezelésében a második vonal szerepét látja el (az első a routing mechanizmus). Emelett pedig szükségünk van egy Web Api Controller-re is amelyben a diagramokhoz szükséges adatokat fogjuk elérni. A “ChartController” minden diagram számára tartalmaz egy metódust.




*4.ábra Kérdőív és válaszai osztálydiagram*

A **4.ábrán** megtekinthetjük az adatok lementéséhez és feldolgozásához szükséges osztályokat. A MongoConnection osztály felépíti a kapcsolatot a MongoDB-vel, beszúrja az adatokat a táblába és kinyeri az adatokat. A QueryBuilding osztály összefűz olyan query-ket, amikkel létrehozhatunk SQL táblákat és beszúrhatunk adatokat a táblába. Ezeket a lekérdezéseket az SQLRepository segítségével tudjuk lefuttatni. Van egy metódusunk olyan query-k számára, amelyek nem térítenek vissza

értékeket, például a beszúrás, és van egy olyan metódusunk, amely segítségével adatokat tudunk visszatéríteni, például a select-es lekérdezések.

# 4. Felhasznált Technológiák

A kimutatások és a kérdőív elkészítéséhez .NET technológiát használtunk, a kódot C# programozási nyelvben írtuk meg. A projektünk két részből állt, az MVC projekt és a Web API.

Az MVC egy jól használható, kipróbált módszer arra, hogy hogyan válasszuk szét a felhasználói felületet és az alkalmazás logikát. Az elsődleges cél az, hogy a felhasználói felület megjelenítéséért felelős kódot teljesen elkülönítsük. Ezáltal annak módosítása, kiegészítése nem vonja maga után az alkalmazás logikát megtestesítő kód módosítását vagy megismétlését.

MVC-ben építettük fel a felhasználói felületet és a Web API segítségével kaptuk meg az adatokat az adatbázisból. Ugyanakkor CSS, JavaScript, Razor, HTML használata nagy szerepet játszott a design kialakításában. Ahhoz, hogy a csapaton belül, valamint a csapatok között zökkenőmentesen haladjon a munka GitHub-ot, valamint Trello-t használtunk. Ezek arra szolgáltak, hogy folyamatosan elérhető legyen az addigi kész projekt a csapattársaknak, valamint “to do” -kat írtunk a feladatok kiosztása és elkülönítése érdekében.

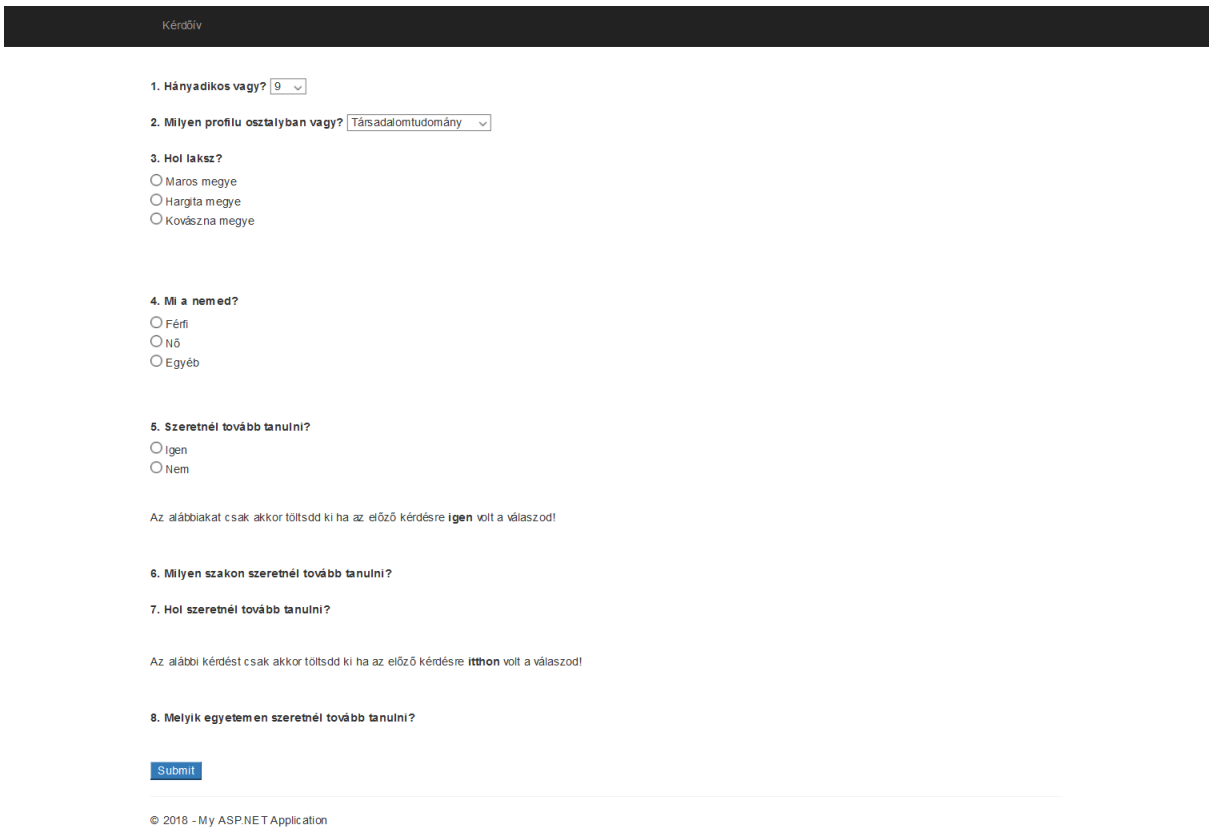
Az adatok tárolását adatbázisban oldottuk meg. Ehhez két típusú adatbázis kezelő rendszert használtunk, relációs és nem relációs adatbázist, vagyis SQL és NoSQL adatbázisokat. A kitöltött kérdőívek adatait NoSQL-ben mentettük, hiszen ezek az adatok nagy mennyiségben érkeznek a szerverhez, nincsenek validálva, vagyis a célunk az volt, hogy egy olyan technológiát találjunk, ahol gyorsan tudunk menteni, a lehető legkevesebb költséggel. A MongoDB-re esett a választásunk az előző ismereteink alapján. Miután lementettük és validáltuk az adatainkat, szükségünk volt egy olyan adatbázis kezelő rendszerre, ahol viszonylag kevés költséggel tudunk lekérdezéseket és procedúrákat írni. Ezért a már MongoDB-ben meglévő adatokat átalakítottuk és lementettük MSSQL-ben. Ennek a folyamatnak a pontos procedúráját a későbbi fejezetben fogjuk tárgyalni.

## 4.1. Felhasználói felület létrehozása

Mint ahogy azt az előbbiekben is olvashattuk a felhasználói felülethez .NET technológiát használtunk, a kódot pedig C# -ban írtuk. A felületen 8 kérdést jelenítettünk meg. Minden kérdésre adottak különböző válaszlehetőségek.

Az ellenőrzéseket JavaScriptben írtuk meg. Ellenőrizni kellett, hogy minden kötelező kérdésre válaszolt a felhasználó, ha nem akkor figyelmeztető üzenet jelenik meg. Emellett voltak olyan kérdések amire, ha nem jó választ adtak, akkor nem jelentek meg a következő kérdések.

A kitöltött kérdőív válaszait először visszaküldtük a Controller-hez, később pedig elmentettük NoSQL adatbázisba, erről a következő fejezetben adunk bővebb leírást.




*5.ábra Kérdőív felhasználói felülete*

Az első kérdésre egy legördülő lista segítségével választ adhat a felhasználó arra, hogy hányadik osztályos. A második kérdésre szintén egy legördülő listából kiválaszthatja, hogy milyen szakon tanul. A következő kérdéseknél megtudhatjuk, hogy hol lakik, illetve, hogy milyen nemű az illetőnek. Az ötödik kérdés, hogy szeretne tovább tanulni, erre, ha a felhasználó igennel válaszol csak akkor jelennek meg a többi kérdések különben csak elküldheti a válaszait. Ha az előző kérdésre igennel válaszolt a felhasználó, akkor arra is választ kell adjon, hogy milyen szakon és hol szeretne tovább tanulni. Arra vagyunk kíváncsiak, hogy hányan szeretnének itthon tanulni nem külföldön, tehát az utolsó kérdésre, hogy melyik egyetemen szeretne tovább tanulni csak akkor tud választ adni, ha az itthon opciót adta meg. Majd a végén rákell kattintson a felhasználó a küldés gombra, ha minden kérdésre válaszolt, akkor elküldődnek az adatok, különben megjelenik egy értesítés, ami tudatja a felhasználóval, hogy minden kérdésre választ kell adni.

## 4.2 Adatok mentése és feldolgozása

A következő fejezetben az adatok pontos mentésének és feldolgozásának a folyamatát fogjuk szemléltetni. A 4-es fejezetben tárgyaltaknak megfelelően a beérkező adatok két adatbázis kezelő rendszeren mentődnek le, amíg elérik a diagramok számára a megfelelő adatstruktúrát. Ahhoz, hogy megértsük, hogy pontosan miért van szükségünk erre meg kell ismernünk a relációs és nem relációs adatbázis kezelő rendszerek előnyét és hátrányát.
A nem relációs adatbázis kezelő rendszerek, ú.n. NoSQL rendszerek, kis költséggel mentenek adatokat. Ezeknek az adatoknak nem szükséges egy előre meghatározott struktúrát megadjunk. Ez a mi esetünkben nagyon előnyös, hiszen az adatok direktben a felhasználók válaszaiból érkeznek, mindenféle szűrés nélkül. Ezen felül a kérdőívek felépítése is folyamatosan változhat, amit a NoSQL megenged. Azonban fontos megemlíteni, hogy ennek a rendszernek a hátránya, hogy a lekérdezések, keresések nagyon költségesek, illetve programozói szempontból nehezen megvalósíthatóak. Összefoglalva, míg mentés szempontjából előnyös választás volt számunkra a NoSQL használata keresnünk kellett egy olyan technológiát is, ahol könnyen, kevés költséggel tudunk lekérdezéseket írni.
Ezért esett a választásunk arra, hogy a NoSQL-ben lementett adatokat kiszűrjük és a számunkra lényeges információkat lementjük SQL-ben. Itt már figyelnünk kellett az adatbázisunk pontos szerkezetére, hiszen később nem, vagy csak nagyon nehezen módosíthatjuk. Az adatbázis szerkezete a harmadik csapat határozta meg, hogy a lekérdezések szempontjából a lehető legmegfelelőbb legyen. A következő képen látható a szerkezet.

| | User_id | Gender_4 | Class_1 | Group_by_specialtization_3 | Group_by_specialtization_2_1 | Group_by_specialtization_2_2 | Group_by_specialtization_2 |
|---|---|---|---|---|---|---|---|
| 1 | 5c0e1a860041b6e4d976fab6 | ferfi | 9 | maros | tarsadalom_tudomany | NULL | NULL |
| 2 | 5c0e1a9c0041b6e4d976fac1 | ferfi | 9 | hargita | NULL | tarsadalom_tudomany | NULL |

***6.ábra:*** *Relációs adatbázis szerkezete*

Látható, hogy az oszlop neve a diagram nevéből és a hozzá tartozó kérdésből áll. Például, a Gender nevű diagram a 4-es számú kérdést használja fel. A Group by specialization nevű diagram a 3-as, a 2.1, a 2.2. stb. kérdéseket használja fel a diagram készítéséhez. Részletesebben a következő fejezetben lesz tárgyalva.
A tábla dinamikusan kódból van létrehozva és adatokkal feltöltve. Ez azt jelenti, hogy a MongoDB-ből lekérjük az adatokat és például annak függvényében, hogy a 3-as kérdésre “maros” választ kaptuk, kitöltjük a Group_by_specialization_2_1 oszlopot, ha “hargita” választ kaptuk a Group_by_specialization_2_2 oszlopot.
Ezzel szemben, MongoDB-ben az adatokat úgy mentettük, hogy az első kérdésre adott válasz, a második kérdésre adott válasz stb. A következő képen látható egy collection pontos szerkezete.

```
_id: ObjectId("5c0e1a860041b6e4d976fab6")
grade: "9"
profil: "tarsadalom_tudomany"
live: "maros"
gender: "ferfi"
learn: "igen"
speciality: "humantudomany"
where: "itthon"
university: "sapientia"
```

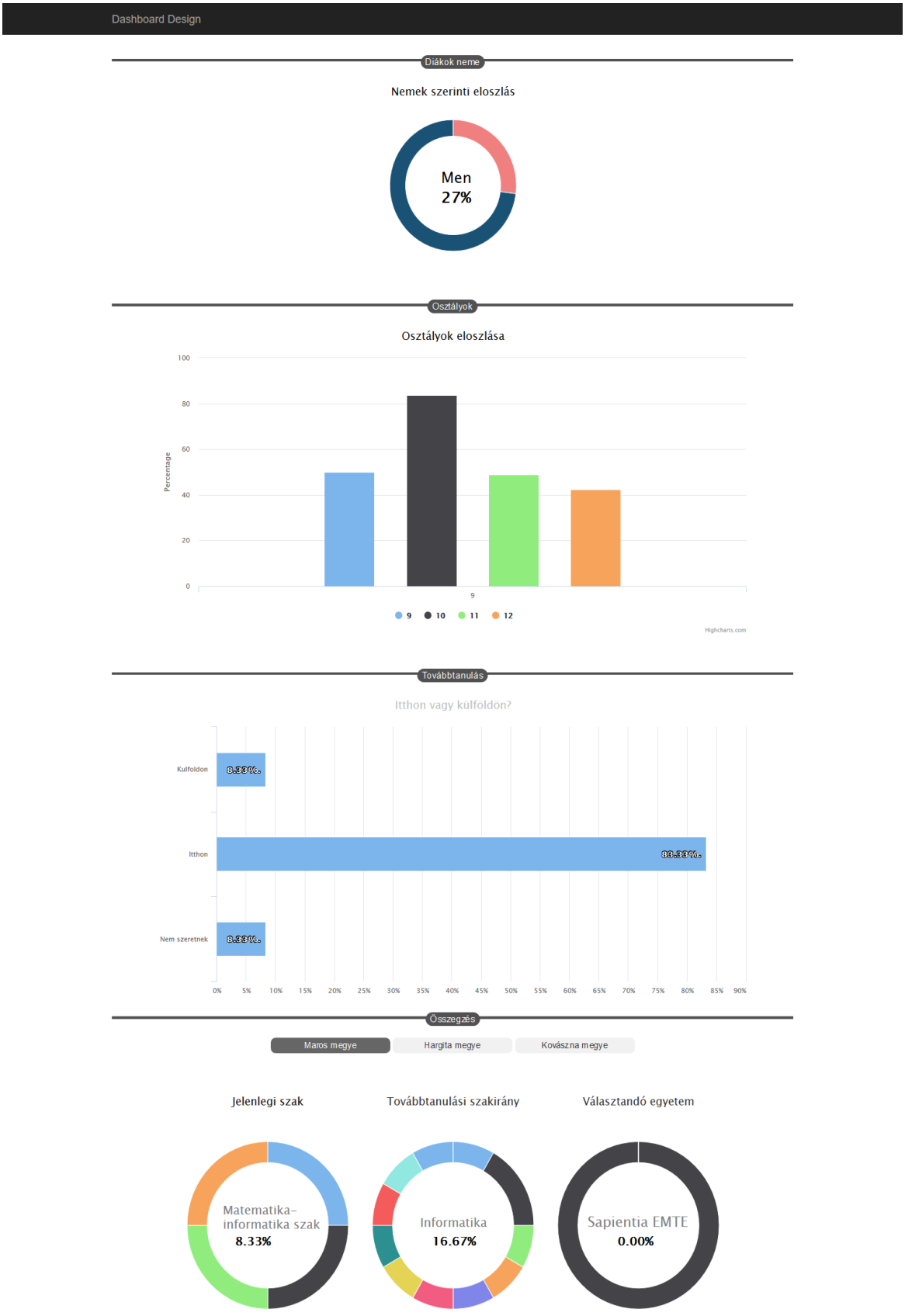

***7.ábra:*** *NOSQL adatbázis szerkezete*

## 4.3 Kimutatások készítése

A következő fejezetben az adatok feldolgozásának a folyamatát, illetve a diagramok és kimutatások készítésének folyamatát fogjuk szemléltetni. A 4-es fejezetben tárgyaltaknak megfelelően a beérkező adatok egy előre definiált tábla szerkezet alapján elkészített adatbázis táblába vannak eltárolva, szerkezetét lásd a **6.ábrán**.

A diagramok megjelenítése hasonlóan a felhasználói felülethez MVC mintára épült, részletekért lásd a [**4**]. fejezetetet. A diagramok felépítéséhez szükségünk volt a kérdőív kitöltőinek válaszaira. Tárolt eljárások segítségével sikerült minden diagram számára felépíteni egy olyan adatszerkezetet amely megfelel a diagram számára.

Egy MVC (Model-View-Controller) projektet hoztunk létre a felhasználói felület elkészítéséhez, emellett még használtuk a Web API keretrendszert amely segítségével kinyertük az adatokat az adatbázisból.
A View-hoz hozzáadtunk *JavaScript* fájlokat, amelyben fel vannak építve a diagramok. Tárolt eljárásokat írtunk ahhoz, hogy megkapjuk azt az adatot, amelyet meg szeretnék jeleníteni a diagramokon. Egyes tárolt eljárásokban dinamikus SQL-t is használtunk mivel volt olyan kimutatás, amely változó volt futásidőben dőlt el, hogy melyikre van az adott pillanatban szükségünk, a tárolt eljárások eredményét átadtuk a diagramoknak. A Controller-ben minden diagramhoz írtunk egy metódust, amely által létrejön a kapcsolat a frontend és a backend között.




***8.ábra:*** *Dashboard design felhasználói felülete*

# 5. Következtetések

A projekt segít egy átfogó képet adni arról, hogy Erdély néhány megyéjében (Maros, Hargita, Kovászna) mennyi azon diákok aránya, akik tovább fejlesztik magukat. E mellett az a tény is fontos, hogy ezen diákok közül hányan szeretnék az anyanyelvükön folytatni tanulmányaikat.

A munkánk eredményesnek bizonyult, akadtak problémák is, amelyeket sikerült közösen áthidalni. Ez a projekt nagyon jól tükrözi azt a munka formát, ami a cégeknél is jelen van. Nagyon sokat tanultunk egy projekt kivitelezéséről, az első lépéstől az utolsóig, valamint egymástól is.

# 6. Jövőbeli tervek

**Az alkalmazás továbbfejlesztési lehetőségei:**

-az adatok interneten való eltárolása, annak érdekében, hogy a kérdőívet bárhonnan ki lehessen tölteni.

-regisztrálás és bejelentkezés, a bejelentkezett felhasználók számára

elérhető lenne a kérdőívek összegzése

-Android/iOS applikáció, a diákok számára elérhetőbb felületet biztosítana

-további kérdések bevezetése, nagyobb kimutatások készítése érdekében

-több kérdőívből álló rendszer létrehozása

# 7. Bibliográfia


1. https://www.highcharts.com/demo/column-stacked-and-grouped
2. http://staruml.io/?fbclid=IwAR2dY1B56_C79qK6UQKpXKzPcT4OB0bkw3eswOJM0Y6QwNW_vkG6R2ciUeg
3. https://www.mongodb.com
4. http://compalg.inf.elte.hu/~tony/Informatikai-Konyvtar/09-Programozas%20C-sharp%20nyelven/Programozas-Csharp-nyelven-Konyv.pdf?fbclid=IwAR0ayHTnlC-zC9TdPf7vcGyYp8JL3HoPCoquARekqNmIWBX_MUifnx-tAbE
5. https://www.tankonyvtar.hu/hu/tartalom/tamop412b2/2013-0002_elektronikus_tanulas/tananyag/JEGYZET-15-2.7._Felmeresek_es_szavazasok.scorml
6. https://rgdi.sze.hu/images/RGDI/honlapelemei/fokozatszerzesi_anyagok/Ramhap_Szabolcs_doktori_disszertacio.pdf
7. http://nyelvek.inf.elte.hu/leirasok/ASP.NET/index.php?chapter=5