SZAKDOLGOZAT

Lengyel Gábor

Debrecen

2010

Debreceni Egyetem
Informatikai Kar

Orvosi kutatások támogatása

WEB-es alkalmazással

Témavezető:
Dr. Fazekas Gábor
Egyetemi docens

Készítette:
Lengyel Gábor
programtervező informatikus

Debrecen, 2010

Tartalomjegyzék

A SZAKDOLGOZAT TÉMÁJÁNAK ÁTTEKINTÉSE

Bevezetés

Szakdolgozatom célja, egy olyan alkalmazás bemutatása, mely modern Microsoft technológiák, segítségével valósítja meg, a statisztikai orvosi kutatásokhoz szükséges web-es támogatást.
A webes támogatás fontosságának, és lehetőségének bemutatásához előbb ismertetem a kutatások mai működését és a modernizálás lehetőségeit.
A lehetőségek ismertetése utána a választott technológiák, egyszerű bemutatása következik, majd az alkalmazás megvalósításának áttekintése.

A STATISZTIKAI ORVOSI KUTATÁSOK CÉLJÁRÓL

A statisztikai orvosi kutatások más néven epidemiológiai kutatások feladata, országos és (vagy) regionális szinten követni a lakosság általános egészségi állapotát és a különböző betegségek előfordulását. Továbbá megfigyelik a befolyásoló környezeti, szociális és biológiai tényezőket és kockázati elemeket.
A kutatók statisztikai valószínűségek alapján adnak választ a kutatások céljául felvetett hipotézisekre. Ezáltal a betegségeket besorolhatják prioritásuk szerint, a későbbiekben pedig hatékonyabb prevenciós programokat állíthatnak fel. További feladata a betegek és betegségek követése (monitoring).
Az ilyen statisztikai kutatásokhoz szüksége van a kutatóknak a népesség állapotára vonatkozó adatokra, melyet a körzeti orvosok bevonásával tudnak megszerezni a lakosságról.
Ezen adatok megszerzésére szolgálnak a kérdőívek. Melynek tartalmát a hipotézis határozza meg, az általános azonosítok mellett.

A KUTATÁS MAI MŰKÖDÉSE.

A statisztikai orvosi kutatásokhoz az alábbi általános lépések és munkaidők (hónapban) szükségesek a mai (vagyis nem számítógépes) technológiával.

Időbeosztás hónapokban

Feladat/tevékenység	1	2	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
1) ELŐKÉSZÍTÉS															
Vizsgálattervezés															
Szakmai konzultáció															
Vizsgálathoz szükséges nyomtatványok postázása															
2) LEBONYOLÍTÁS															
Vizsgálati adatlapok eljuttatása a megyei központokba															
fizikális vizsgálat															
Minőségellenőrzések végrehajtása															
Adatbevitel és adatkezelés															
3) ADATELEMZÉS, ÉRTÉKELÉS															
Statisztikai analízis															
4) ZÁRÓKONFERENCIA															
Záró konferencia															

1) ELŐKÉSZÍTÉS

Ebben a fázisban történik a kutatás céljának meghatározása, ekkor már körvonalazódik mely adatok lesznek szükségesek és milyen előzetes tanulmányokra van esetlegesen szükség.
A kutatás céljának meghatározása után következnek a szakmai konzultációk a kutatók között, ahol az előzetes tanulmányok alapján, meghatározzák a kérdőívek tartalmát, a felmérés helyszíneit és szereplőit.
A szükséges adatok meghatározás után a megfelelő szakemberek összeállítják a kérdőíveket,

mely kérdőíveknél külön figyelembe kell venni, hogy a kérdések ne befolyásolják a válaszadót, de emellett lefedjenek minden szükséges válaszlehetőséget.

2) LEBONYOLÍTÁS

A kutatás lebonyolításának első lépése a kérdőívek kinyomtatása és postázása a megadott helyszínekre.
A kutatás és a kutató céljától függően, egy kérdőív átlagosan 15-20 oldalas is lehet és 60-80 kérdést is tartalmazhat. Egy kutatás -validitástól függően- pedig átlagosan 3000 darab kérdőívet igényel, tehát 45000 – 60000 oldalt szükséges kinyomtatni és elpostázni az orvosi rendelőkhöz majd a válaszadás után vissza.
Az adatlapok visszaérkezése után következik, a minőség ellenőrzés ahol ellenőrzik, hogy a visszaérkező adatok értelmezhetők e, vagyis nincs e bennük önellentmondás. Ez a fázis egybeesik az adatok rögzítésével, ahol az adatokat felviszik számítógépre, későbbi feldolgozás céljából.
Itt érdemes azonban megjegyezni, hogy a hibásan kitöltött adatlapok adat ellenőrzése nehézkes és gyakran lehetetlen is. Például ha két egymást kizáró adatot kéne, felvinnünk nem tudhatjuk, hogy melyik adat a valódi és melyik a véletlen elírás.

(Pl.:

…

12. kérdés: (Nem válasz esetén hagyja ki a következő kérdést)

Szed szívgyógyszert? Nem

13. kérdés:

Hány szívgyógyszert szed naponta? 2

…

)

Ebben az esetben a mai gyakorlat szerint úgy viszik fel az adatlapot, ahogy az ki lett töltve, vagy az egész adatlapot használhatatlannak tekintik. Ez természetesen a későbbi statisztikai elemzések téves eredményéhez vezethet, de legalábbis pontatlanabb lesz tőle az eredmény.

3) ADATELEMZÉS, ÉRTÉKELÉS

A számítógépre felvitt adatok kiértékelését, statisztikai műveletek segítségével végzik a kutatók.
Ezen műveletekre és az adatbázissal kapcsolatos műveletekre a STATA, SPSS, Epi info nevű programok használhatók a legjobban, mely Európa szinten a jelenleg legjobban elterjedt egészségügyi statisztikai elemző szoftverek.

A KUTATÁS MŰKÖDÉSE WEB-ES TÁMOGATÁSSAL

A statisztikai orvosi kutatások jelenlegi menetét áttekintve a legnagyobb software-sen támogatható pont az adatgyűjtés, itt webes alkalmazások segítségével egyszerre lehet gyorsítani az adatgyűjtést és validabbá lehet tenni az adatokat (vagyis csökkenthetők a hibák száma).

Időbeosztás hónapokban

Feladat/tevékenység	1	2	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
1) ELŐKÉSZÍTÉS															
Vizsgálattervezés															
Szakmai konzultáció															
Vizsgálathoz szükséges kérdőív előállítása															
2) LEBONYOLÍTÁS															
Vizsgálati adatlapok eljuttatása a megyei központokba															
fizikális vizsgálat															
Minőségellenőrzések végrehajtása															
Adatbevitel és adatkezelés															
3) ADATELEMZÉS, ÉRTÉKELÉS															
Statisztikai analízis															
4) ZÁRÓKONFERENCIA															
Záró konferencia											megspórolt idő				

1) ELŐKÉSZÍTÉS

Webes alkalmazás használata esetén sem különbözik az előkészítés feladatköre, megkell határoznunk a kutatás célját és a kutatáshoz szükséges kérdőíveket. A szokásos kérdőív meghatározása után azonban szükséges a válaszok elfogadhatóságának pontosabb meghatározása. A későbbi validitás szűrésének érdekében ez plusz munkát jelent a kérdőív összeállítójának.

De ebben az esetben a kérdőíveket nem kell kinyomtatni pusztán publikálni, kell az alkalmazásban.

2) LEBONYOLÍTÁS

Az orvosok a kérdőív publikálása után azonnal megkezdhetik a kérdőív kitöltését az interneten keresztül. Az alkalmazás azonban a korábbi technológiával ellentétben nem

engedi, hogy hibás adatot vigyünk fel sem egyszerű adat szinten, sem összefüggéseiben, ezt biztosítja a korábban az adatlap pontosítására fordított idő.
További előnye, hogy a kérdőívek kitöltése után a kutatók azonnal használható adathoz jutnak belőle, amiből már kutatás közben végrehajthatnak előzetes teszteket.

3) ADATELEMZÉS, ÉRTÉKELÉS

Az alkalmazás képes a STATA-nak megfelelő formátumban kimenteni az adatokat, így az adatelemzés és értékelés nem különbözik az eddig megszokott módszerektől.
Így nem szükséges új technológiára átállni a kutatóknak.

ÖSSZEFOGLALÁS

Az interneten keresztül történő adatgyűjtés tehát kezdetben nagyobb energia ráfordítás igényel, mivel a kérdőíveket plusz adatokkal kell ellátni. De ez a plusz energia mindenképpen megtérül, a későbbiekben mivel nem kell a kérdőíveket kinyomtatni, kipostázni, kitöltetni és utána egyesével gépre vinni őket, továbbá mindenképpen fontos, hogy a gyűjtött adatok kimutathatóan pontosabbak lesznek az eddigieknél.

A FELHASZNÁLT TECHNOLÓGIÁK ÁTTEKINTÉSE

Az alkalmazás a rétegelt alkalmazásfejlesztésnek megfelelően, három fő rétegre bontható.

Réteg	Technológia	Feladatköre
Kliens oldali alkalmazás	SilverLight 3.0	- Kommunikáció a felhasználóval. - Adat és metódus megjelenítés.
Serveroldali alkalmazás	ASP.NET 3.5	- Biztosított adatelérés meg valósítása.
Adatbázis réteg	SQL SERVER 2008	- Konzisztens adatállapot megőrzése.

Az adott rétegek egymástól jól elszeparáltan helyezkednek el és csak a mellette lévő rétegekkel kommunikálnak.
Az adatbázis réteg tárolja az alkalmazáshoz szükséges adatokat, lehetőséget ad azok kezelésére, adattáblák formájában. Az adatbázisréteg csak a serveroldali alkalmazással kommunikál a megfelelő titkosítások használatával.
A szerveroldali alkalmazás valósítja meg a teljes üzleti logikát, beleértve a felhasználó kezelést és a felhasználók által végrehajtható műveleteket is. Fontos szerepe van abban, hogy a felhasználó csak valid adatokat juttathasson az adatbázisba és csak a számára engedélyezet, műveleteket hajthassa vége.
A kliens oldali réteg szerepe a felhasználó számára érthető és könnyen kezelhető formában közölni az aktuális állapotot beleértve az adatokat és lehetséges műveleteket.

SQL SERVER 2008

A Microsoft SQL Server 2008 a Microsoft adatplatformja, amely relációs adattárolásra, nagyteljesítményű adatintegrációra (ETL), gyors és skálázható multidimenzionális adatelemzésre (OLAP), adatbányászatra és jelentéskészítésre (Reporting) képes. Mindemellett kiváló platform az adat alapú alkalmazások számára.

A Microsoft tervezői eszközökre jellemzően hatékonyan támogatja és együttműködik az egyéb Microsoft által készített tervező eszközökkel, például a Visual Studio-val és az expression software családdal.

A teljesítmény igények meghatározásával megállapítható mely SQL Serverre van szükségünk az alkalmazás gyors és stabil adatbázis kezeléséhez.
Az alábbi csomagok közül választhatunk az SQL Server 2008 esetén:

Enterprise

Az SQL Server 2008 Enterprise egy átfogó adatplatform, amely megfelel a nagyvállalati adatfeldolgozás (OLTP) és adattárházak komoly kihívásainak.

Standard

Az SQL Server 2008 Standard egy teljes körű adatkezelő és üzleti intelligencia platform kiváló kezelhetőséggel és felügyelhetőséggel a kisebb igényű adatbázis alkalmazások számára.

SQL Server specializált változatok

Workgroup

A Workgroup változat egy megbízható adatkezelő és jelentéskészítő platform, amely biztonságos távoli szinkronizációs és felügyeleti lehetőségeket nyújt elsősorban a távoli telephelyeken működtetett alkalmazások számára.

Compact

Az SQL Server Compact egy ingyenes, beágyazott SQL Server adatbázis, amely ideális egyedülálló és offline alkalmazások fejlesztéséhez. Az SQL Server Compact minden Windows® operációs rendszeren képes futni, így mobil eszközökön (Windows Mobile) és PC-ken is (Windows XP, Windows Vista® és Windows 7).

Express

Az SQL Server 2008 Express egy ingyenes SQL Server változat. Ideális tanulási célokra, valamint klienseken működő, vagy kisebb méretű kiszolgáló alkalmazások készítésére. Szoftverfejlesztő cégek a saját alkalmazásuk részeként is adhatják ügyfeleik számára. Az SQL Server Express számos funkciót tartalmaz, pl. az SQL Server 2008 Reporting Services-t, ami egy kiszolgáló alapú platform hagyományos és interaktív jelentések készítésére és kiszolgálására. Tartalmaz még egy grafikus felügyeleti alkalmazást is az SQL Server 2008 Management Studio Express-t az adatbázisok hatékony felügyeletéhez.

A tervezett felhasználók számát (150-200) és a szükséges adatokat figyelembe véve, elégségesek az SQL Server 2008 Express szolgáltatásai és a hozzá tartozó grafikus felület az SQL Server 2008 Management Studio Express szolgáltatásai.

További előny, ha a későbbiekben fejlesztés történik a felhasználhatóság területén, vagyis kiterjesztenénk a statisztikai adatgyűjtés felhasználhatóságát, akkor a technológia megváltoztatása nélkül elégséges egy nagyobb teljesítményű SQL Server-re áttérnünk, hogy a megfelelő teljesítményt kapjunk.

ASP.NET 3.5

2007 november 19.-én adta ki a Microsoft a .Net Framework 3.5-ös változatát, a Visual Studio 2008-cal együtt.

Az ASP.NET mögött a .NET framework áll (innen is az elnevezés), mely alkalmazások fejlesztéséhez és futtatásához használt programozási modell, illetve szoftverfejlesztői platform. Két fő komponense van: a Common Language Runtime (CLR) és a Base Class Library (BCL). A CLR a .NET alapja, egy futtató környezet, a Common Language Infrastructure (CLI) Microsoft-féle implementációjának tekinthető. A CLI egy nyílt specifikáció, melyet a Microsoft fejlesztett ki, és leírja, hogyan kell kinéznie a futtatható kódnak és a futtató környezetnek.
Az ASP.NET teljes szerver és kliens oldali támogatást biztosít a web-es alkalmazásainkhoz. De ma már a komplexebb kliens oldali esetén fokozatosan felváltja a silverlight, míg a ASP.NET megmarad server oldali technológia.

Alkalmazásainkban ASP.NET-tel valósítjuk meg a server oldali üzleti alkalmazásainkat, mely az adatbázistól nyeri az adatot és a megjelenítési réteggel kommunikál.

Az adatbázissal való kommunikáció

Az adatbázissal való kommunikációnak 2 alapvető lehetőség adott:

1) Commandok (parancsok) segítségével, ahol a létrejött adatbázis kapcsolat segítségével SQL parancsokat küldhetünk az adatbázis szervernek. Itt lehetőségünk van bármilyen SQL-beli parancsot továbbítani az SQL szervernek.

A technológia előnye és hátrány is egyben, hogy a kapcsolatot és a parancsok küldését és a válasz adattáblák fogadását a tervezőnek kell megvalósítania.
Ezzel a módszerrel a lehető leggyorsabb műveleteket hajthatjuk végre és a műveletek lehetnek akár DML, DCL, DDL adatbázis utasítások is.
Ezért a teljes körű lehetőségért, cserébe azonban nekünk kell kezelni a teljes kapcsolatot és nekünk kell felügyelni az SQL utasításokat és a válaszokat is.

2) LINQ (nyelvbe ágyazott lekérdezések) és Entity data model.
Abban az esetben használatos, ha a már meglévő táblákhoz kívánunk lekérdezéseket vagy egyéb adatmanipuláló műveleteket létrehozni. Ebben az esetben a táblákhoz létrehozunk egy Entity data model-t mely segítségével már objektum orientáltan kezelhetjük az adatainkat és Linq segítségével lekérdezéseket intézhetünk az adatbázis irányába. Így ez a módszer felügyelt erősen típusos és könnyen megvalósítható. Ellenben az Entity data model szükségessége miatt, csak meglévőtáblák adatainak kezelésére alkalmas.

Kliens oldallal történő kommunikáció

A kliens oldallal (silverlight-tal) való kommunikációra is több lehetőség adódik:

1) Adott az egyszerű webservice mely már egy régebbi technológiának számít és ma már nem szokás kézzel megírni őket. A technológia alkalmas ugyan a server oldali műveletek kipublikálására, de bonyolult fejlesztése miatt ma már nem használatos ebben a formában. A további technológiák teljes mértékben kiválthatják ezt a megoldást, így ennek használatát és ismertetését elhagyom és nem is javaslom.

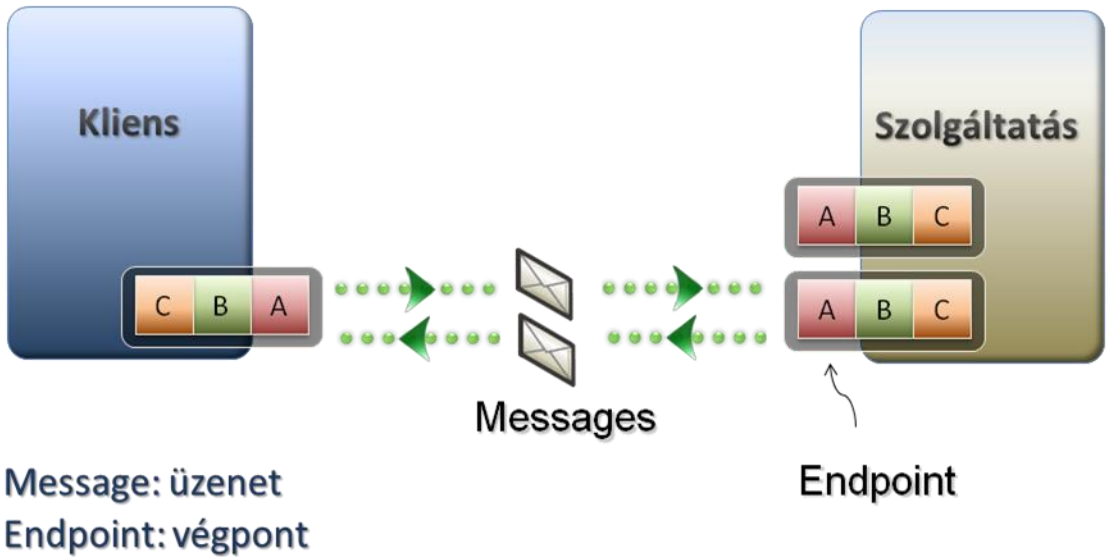
2) A WCF (Windows Communication Foundation) hatékony megoldást biztosít a kommunikációra több platform támogatásával. Helyettesíti az alábbi megoldásokat:
-ASMX webszolgáltatások – interoperabilitás
-.NET Remoting – hatékony .NET-.NET

-WSE – biztonságos webszolgáltatások

-Enterprise Serivces – tranzakció kezelés

-MSMQ – üzenetsorok

Alapvetően a SOAP szemléletet támogatja, vagyis biztonságos szolgáltatásokat fejlesztünk, melyeket kipublikálhatunk az internetre, adott üzenetküldési protokollal és titkosítással.



Egy szolgáltatás rendelkezhet több végponttal is, ezen végpontokra érkeznek az üzenetek a klienstől az adott kommunikációnak megfelelően. A biztonságos kommunikációhoz három dolgot kell megadnunk minden WCF szolgáltatás esetén

A- Address – a végpont címe

Meghatározza a végpont elérési címét, a megszokott http://... alakban.

B- Binding – kötés

Az adatkapcsolat pontos módját határozza meg, vagyis azt, hogy az üzenetek milyen tikosítással és milyen magasabb szintű szolgáltatással rendelkeznek.

C- Contract – szerződés

Megadja azt az interface-t mely segítségével kliens oldalon is elérhető a kipublikált osztály. Ezen interface technológiával, bármilyen osztályt kipublikálhatunk, melyet ellátunk a megfelelő jelzőkkel.

3) RIA Service

A silverlight3-mal egy időben megjelent technológia, melynek segítségével hatékonyan publikálhatunk ki a kliens oldalra az adatbázisból, CRUD módon (Creat, Read, Update, Delet) az adat tábláinkat.
Az adattáblákhoz úgynevezett Entity data model-t hozunk létre melyek segítségével már objektum orientált módon és erősen típusosan tudjuk kezelni az adatbázisunkat.
Az így létrehozott Entity data model pedig Domain Service segítségével egyszerűen publikálhatjuk anélkül, hogy a kommunikáció bármely részéről kézi beállítással gondoskodni kéne.

SILVERLIGHT 3.0

2009. július 10.-én a tavaszi MIX09 konferencián mutatkozott be a Microsoft webes multimédiás platformjának, a Silverlightnak a 3.0 verziója.
A Silverlight a Microsoft interaktív böngésző alapú médiaplatformja, melyet korábban Windows Presentation Foundation/Everywhere néven említették. A fejlesztés több mint két évig tartott, mely eredményeként megszülető szoftvercsalád a Microsoft reményei szerint fel fogja venni a versenyt az Adobe népszerű és igen elterjedt Flash platformjával. A platform az Internet Explorer, Firefox, Opera és Safari böngészőkkel működik együtt jelenleg.

A technológia fontosabb előnyei, közé tartozik a teljesen vektorgrafikus megjelenítés, továbbá a layout rendszerű elrendezések, melyek segítségével szabadon méretezhetővé tehetjük a megjelenítési felületünket.
A silverlight 3-ban megjelent új fejlesztések, tovább növelték a felhasználhatóságát.

Ilyen fejlesztés az OutOfBrowser melynek segítségével a webes alkalmazásunkat installálhatjuk a kliens számítógépére, ezzel jelentősen csökkentve az alkalmazás későbbi sávszélesség igényét. Az installálás után, már nincs szükség arra, hogy minden alakalommal letöltsük az alkalmazásunkat az interneten keresztül, csupán a frissülő adatokat kell letöltenünk.
További újítás mely a hármas kiadásban jelent meg a RIA melynek célját már korábban kifejtettem. RIA segítségével gyorsan és hatékonyan fejleszthetünk üzleti alkalmazásokat és publikálhatunk ki adatbázisokat.

AZ ALKAMAZÁS FEJLESZTÉSE

A kutatás céljának bemutatása és a technológia ismertetése után, részletesebben ismertetem a fejlesztett alkalmazást.

TECHNIKAI ELVÁRÁSOK, CÉLKITÜZÉSEK

Az alkalmazásnak két felhasználói típusa van, a nagyobb technikai támogatást igényelő kutatók, akik kisebb létszámban és kevesebb ideig használják a rendszert, de gazdagabb felhasználói felületet igényelnek. Másik felhasználói réteg a körzeti orvosok (röviden orvosok), akiknek kötött számítógépes és kiépített internetes rendszerük van.
Ezért célkitűzés, hogy az alkalmazás erőforrás igénye ne haladja meg a körzeti orvosi rendelőkben rendelkezésre álló rendszerek lehetőségeit és mindenképpen fontos a lehető legkisebb sávszélesség igény.

Ezen igényeket kielégítendő alkalmazom a silverlight OutOfBrowser lehetőségét melynek az előnyeit korábban kifejtettem, ezzel és optimalizálási végig gondolásokkal minimalizálom a sávszéllesség igényét.

A KUTATÁS VÉGREHAJTÁSÁNAK FOLYAMATA

1) Kutatás létrehozása

   Kutatást a bejelentkezet kutató hozhat létre a saját nevéhez tartozóan. A kutatás ebben a fázisban szerkesztés státuszban van, így ekkor még lehetőség van a hozzá tartozó alap adatok módosítására és a kérdőív szerkesztésére. A kutatás módosítására csak az őt létrehozó kutatónak van joga.
   A kérdőívet kérdésekből állítjuk össze, ezek a kérdések pedig 6 lehetséges kategóriába tartozhatnak, és tulajdonságaikkal tehetők egyedivé.

2) Kutatók hozzárendelése

   A kutatásokhoz hozzá kell rendelnünk a kutatásokban résztvevő orvosokat is, ekkor a korábban a felhasználók közt bejegyzett orvosokat rendelhetjük hozzá a kutatásokhoz. A bejegyzett orvosok a kutatás aktiválása után láthatják a saját adatlapjukon a kutatásokat, míg a hozzá nem rendelt kutatásokról nem kapnak értesítést.

3) Utólagos módosítások végrehajtása

   A kutatónak ebbe a fázisban még lehetősége van bármilyen változtatás végrehajtására.

4) A kutatás aktiválása

   A kutató aktiválja a kutatást, ezek után már nincs lehetősége a kutatás adatainak módosítására és a kérdőív megváltoztatására. Ekkor jön létre a kutatáshoz tartozó adatbázistábla és ekkor a kutatás státusza aktívvá változik. Ezek után lehetővé válik az adatgyűjtés, a kutatáshoz rendelt orvosoktól.

5) Válaszgyűjtés

   Az orvosok az adatlapjukon megjelenő kérdőívekre válaszolhatnak, és amennyiben a válaszok megfelelnek a beállított határoknak és formátumoknak akkor az orvos rögzítheti

a kutatás válaszait. A kérdőívekre az orvosok egészen addig válaszolhatnak, amíg a kutatatás aktív állapotban van.

6) A kutatás zárolása

   A kutatás zárolását a kutató hajthatja végre, ezzel befejezve a kutatást, vagyis befejezettre állítja a kutatás státuszát. Utána már nincs lehetőség további adatgyűjtésre csupán a már megszerzett adatokat menthetjük a STATA formátumnak megfelelően.

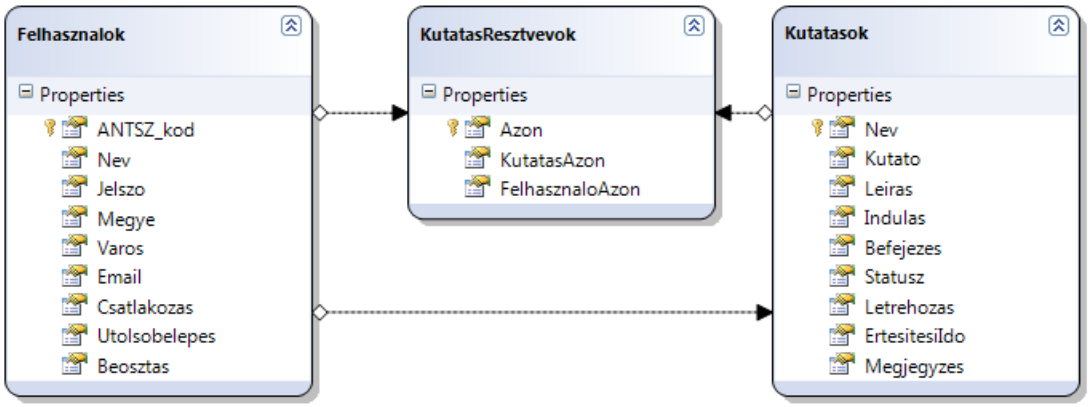
7) Az adatok kigyűjtése

   A kutatónak a kutatatás aktiválása után lehetősége van a beszerzett adatok kimentésére a megfelelő formátumban.

AZ ADATBÁZIS RÉTEG

Az adatbázisrétegben három statikus táblából és tetszőleges számú dinamikus táblából áll. A statikus táblák az alkalmazás működéséhez és a felhasználók kezeléséhez szükségesek, míg a dinamikus táblák egy-egy kutatáshoz tartoznak, létrehozásuk és kezelésük az alakmázásból történik.

A STATIKUS TÁBLÁK



Ezek az adatbázis táblák állandóan léteznek csupán a tartalmuk változik így ezek a alkalmazás statikus adatbázisai.

A Felhasználók táblában, tárolódnak az orvosok és a kutatók adatai, mint például a jelszava, a neve és ANTSZ_kódja. A tábla minden rekordja egy-egy felhasználóhoz tartozik a teljes tábla lekérdezéséhez és módosításához csak a kutatóknak van joga, az orvosok csupán saját adataikat változtathatják meg.

A Kutatások táblában minden rekord egy-egy kutatás állandó adatait tartalmazza, ezen adatok minden kutatáshoz hozzátartoznak és ezen adatok segítségével kezelhetjük kutatásainkat. A kutatásokat a kutatók hozzák létre, ekkor egy új rekordbejegyzés kerül az adattáblába.

Azt, hogy egyes kutatásokat mely orvosok tölthetik ki azt a KutatásRésztvevők tábla határozza meg, mégpedig úgy, hogy minden egyes kapcsolathoz egy rekordot rendel. (Az ebben a tálában található azonosító oszlopnak nincs igazi szerepe csak a ráépülő EntityDataModel-hez szükségesek.)

A DINAMIKUS TÁBLÁK

A dinamikus táblák az alkalmazás futása közben jönnek létre és törlődnek. Ezek a táblák egy-egy kutatáshoz tartoznak. A táblák neve megegyezik a kutatás nevével az oszlopai pedig a kutatás kérdései. A tábla mindig csak a kutatás megkezdésekor jön létre mivel ezek után már nincs lehetőség a kérdések módosítására. A táblát a kutatásban résztvevő orvosok töltik fel a válaszaikkal.

AZ ADATBÁZISBAN VÉGREHAJTHATÓ MŰVELETEK

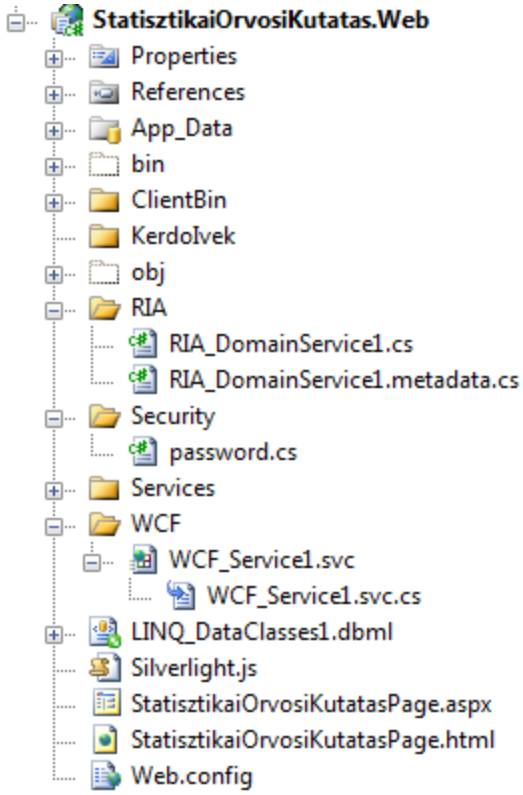
Az alkalmazásban kétféle jogkör van a nagyobb hozzáféréssel rendelkező kutató és kisebb hozzáférésű orvos.

A kutatónak az azonosítása után jogában áll újfelhasználót felvinni vagy régieket törölni. Továbbá jogában áll a saját nevéhez tartozó kutatások adatainak módosítására vagy új kutatások létrehozására.

Az orvosnak azonosítása után jogában áll a neki feltett kérdőívekre válaszolni, továbbá ezekről a kutatásokról olvashatja az adatokat, de nincs joga ezeken az adatokon változtatni. Az orvosnak továbbá ahhoz sincs joga, hogy más felhasználó adatain változtasson.

SERVEROLDAL

A serveroldali alkalmazás ASP.NET technológiával van megvalósítva és az alábbi mappaszerkezettel rendelkezik:



A mappák egy részét a visual studio automatikusan hozza létre, ezek a mappák és fájlok elengedhetetlenek az ASP.NET működéséhez, ezekben a mappákban vannak leírva a konfigurációs beállítások és egyéb adat fájlok.
Az alábbi mappák és osztályok valósítják meg a lényegbeli alkalmazást:

LINQ_DataClasses1.dml

Ez a fájl tartalmazza a `Felhasznalok, Kutatasok, KutatasResztvevok` osztályokat melyek megvalósítják az adatbázis tábláknak a linq általkezelhető specifikációját. Ezen osztályokkal kezelhetjük a linq-val az adatbázis tábláinkat. Alábbiakban a kutatások osztály

részlete található a részlet csupán a kutatások osztály 11 tulajdonsága közül 4-et mutat be (Név, kutató, Leírás, Indulás).

```
      [Table(Name="dbo.Kutatasok")]
      [DataContract()]
      public partial class Kutatasok : INotifyPropertyChanging,
INotifyPropertyChanged
      {

            private static PropertyChangingEventArgs emptyChangingEventArgs
= new PropertyChangingEventArgs(String.Empty);

            private string _Nev;

            private int _Kutato;

            private string _Leiras;

            private System.Nullable<System.DateTime> _Indulas;

            …

    #region Extensibility Method Definitions
    partial void OnLoaded();
    partial void OnValidate(System.Data.Linq.ChangeAction action);
    partial void OnCreated();
    partial void OnNevChanging(string value);
    partial void OnNevChanged();
    partial void OnKutatoChanging(int value);
    partial void OnKutatoChanged();
    partial void OnLeirasChanging(string value);
    partial void OnLeirasChanged();
    partial void OnIndulasChanging(System.Nullable<System.DateTime> value);
    partial void OnIndulasChanged();
    …
    #endregion

            public Kutatasok()
            {
                  this.Initialize();
            }

            [Column(Storage="_Nev", DbType="NVarChar(50) NOT NULL",
CanBeNull=false, IsPrimaryKey=true)]
            [DataMember(Order=1)]
            public string Nev
            {
                  get
                  {
                        return this._Nev;
                  }
                  set
                  {
                        if ((this._Nev != value))
```

```
                        {
                                this.OnNevChanging(value);
                                this.SendPropertyChanging();
                                this._Nev = value;
                                this.SendPropertyChanged("Nev");
                                this.OnNevChanged();
                        }
                }
        }

        [Column(Storage="_Kutato", DbType="Int NOT NULL")]
        [DataMember(Order=2)]
        public int Kutato
        {
                get
                {
                        return this._Kutato;
                }
                set
                {
                        if ((this._Kutato != value))
                        {
                                if
(this._Felhasznalok.HasLoadedOrAssignedValue)
                                {
                                        throw new
System.Data.Linq.ForeignKeyReferenceAlreadyHasValueException();
                                }
                                this.OnKutatoChanging(value);
                                this.SendPropertyChanging();
                                this._Kutato = value;
                                this.SendPropertyChanged("Kutato");
                                this.OnKutatoChanged();
                        }
                }
        }

        [Column(Storage="_Leiras", DbType="NVarChar(MAX)")]
        [DataMember(Order=3)]
        public string Leiras
        {
                get
                {
                        return this._Leiras;
                }
                set
                {
                        if ((this._Leiras != value))
                        {
                                this.OnLeirasChanging(value);
                                this.SendPropertyChanging();
                                this._Leiras = value;
                                this.SendPropertyChanged("Leiras");
                                this.OnLeirasChanged();
                        }
                }
```

```
}

[Column(Storage="_Indulas", DbType="Date")]
[DataMember(Order=4)]
public System.Nullable<System.DateTime> Indulas
{
	get
	{
		return this._Indulas;
	}
	set
	{
		if ((this._Indulas != value))
		{
			this.OnIndulasChanging(value);
			this.SendPropertyChanging();
			this._Indulas = value;
			this.SendPropertyChanged("Indulas");
			this.OnIndulasChanged();
		}
	}
}

…
}
```

Az osztály megvalósítja az `INotifyPropertyChanging` interface-t mely segítségével a tulajdonságok (property-k) értesítést küldhetnek a megjelenítési felületnek a tulajdonság változásáról. Ehhez szükséges az interface megvalósítása mellett a `this.SendPropertyChanged("***")` metódus meghívása is mellyel a konkrét értesítést küldjük a felületnek. Ezzel a megoldással az osztály felület függetlenül valósíthatja meg a funkcióját, a felület pedig csak később kerül megvalósításra a felsőbb rétegben.
Az osztályban megfigyelhetők továbbá ugynevezet jelölések (tag-ek) melyek segítségével specifikációt adhatunk meg az osztályról, és az osztály egyes tulajdonságairól, hozzá kötve az osztályt az adatbázistáblánkhoz.

A `[Table(Name="dbo.Kutatasok")]` `[DataContract()]` jelölésekkel hozzákötjük az osztályt egy táblához melynek a neve: dbo.Kutatasok.

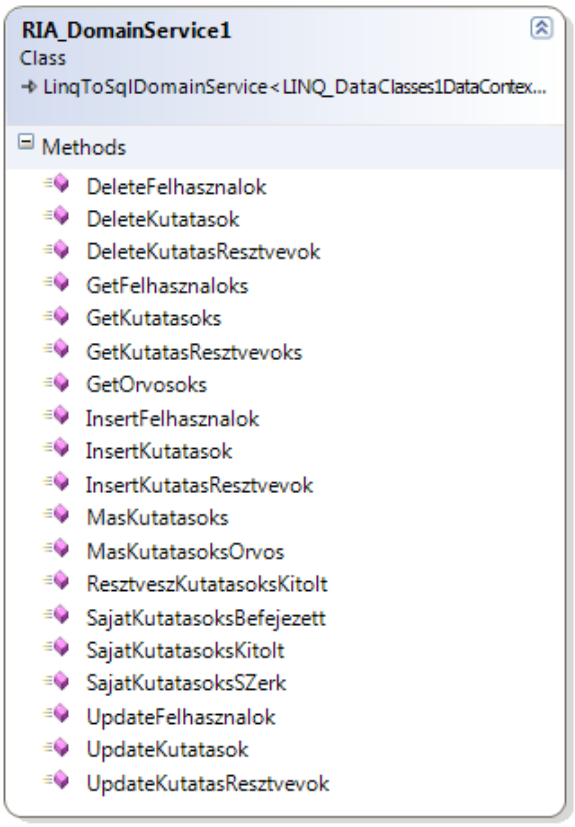
A `[Column]` jelölésekben megadhatjuk a tulajdonságnak az adatbázistálában meglévő jellemzőit ilyenek lehetnek például az alábbiak:

`[Storage="_Nev"]`	A táblában hozzátartozó oszlop nevét határozza meg.
`[DbType="NVarChar(50) NOT NULL"]`	Így adható meg az adatbázistábla beli oszlop tipusa.
`[CanBeNull=false]`	Így rögzíthető, hogy az érték tartalmazhat vagy nem tartalmazhat null értéket.
`[IsPrimaryKey=true]`	Osztályonként egy property-hez rendelhető érték mely a primarykey-t határozza meg.
`[DataMember(Order=1)]`	Az oszlop sorszámát adhatjukmeg.

Bár az osztályt egyszerű szabályok alapján kézzel is megírhatnánk azonban erre nincs szükség, mivel a visual studio grafikus szerkesztőt biztosít, melynek segítségével az adatbázis tábla egyszerű kiválasztásával létrehozhatjuk a hozzá tartozó kódot, mely a táblánknak megfelelően lesznek ellátva jellemzőkkel.

Ha ennek megfelelően megvalósítottuk ezeket az osztályokat, akkor a továbbiakban osztályokként kezelhetjük adatbázis tábláinkat, és linq segítségével lekérdezéseket hajthatunk végre rajta.

RIA



RIA_DomainService1.cs fájl-ban szerepel a `RIA_DomainService1` osztály melyet szintén autómatikusan generáltatam a visual studioval és felhasználtam benne a korábban létrehozot LINQ_Data osztályokat. Az osztály feladata a linq osztályokhoz autómatikusan megvalósítani a kezelő felületet beleértve a lekérdezést az adatfelvitelt a törlést és a módosítást is.

```
namespace StatisztikaiOrvosiKutatas.Web.RIA
{
    using System;
    ...
    using System.Data.Linq;
    using System.Web.DomainServices.LinqToSql;
    using StatisztikaiOrvosiKutatas.Web;

    [EnableClientAccess()]
```

```
    public class RIA_DomainService1 :
LinqToSqlDomainService<LINQ_DataClasses1DataContext>
    {
        public IQueryable<Felhasznalok> GetFelhasznaloks()
        {
            return this.Context.Felhasznaloks;
        }

        public IQueryable<Felhasznalok> GetOrvosoks()
        {
            return this.Context.Felhasznaloks.Where(g => g.Beosztas ==
"Orvos");
        }

        public void InsertFelhasznalok(Felhasznalok felhasznalok)
        {
            this.Context.Felhasznaloks.InsertOnSubmit(felhasznalok);
        }

        public void UpdateFelhasznalok(Felhasznalok currentFelhasznalok)
        {
            this.Context.Felhasznaloks.Attach(currentFelhasznalok,
this.ChangeSet.GetOriginal(currentFelhasznalok));
        }

        public void DeleteFelhasznalok(Felhasznalok felhasznalok)
        {
            this.Context.Felhasznaloks.Attach(felhasznalok);
            this.Context.Felhasznaloks.DeleteOnSubmit(felhasznalok);
        }

        public IQueryable<Kutatasok> GetKutatasoks()
        {
            return this.Context.Kutatasoks;
        }

        public IQueryable<Kutatasok> SajatKutatasoksSZerk(int _azonANTSZ)
        {
            return this.Context.Kutatasoks.Where(f=> f.Kutato==_azonANTSZ
&& f.Statusz=="SZERKESZT");
        }
        public IQueryable<Kutatasok> SajatKutatasoksKitolt(int _azonANTSZ)
        {
            return this.Context.Kutatasoks.Where(f => f.Kutato ==
_azonANTSZ && f.Statusz == "KITOLTES");
        }
        public IQueryable<Kutatasok> SajatKutatasoksBefejezett(int
_azonANTSZ)
        {
            return this.Context.Kutatasoks.Where(f => f.Kutato ==
_azonANTSZ && f.Statusz == "BEFEJEZETT");
        }

        public IQueryable<Kutatasok> ResztveszKutatasoksKitolt(int
_azonANTSZ)
        {
```

```
            return this.Context.Kutatasoks.Where(f => f.Statusz ==
"KITOLTES" && f.KutatasResztvevoks.Any(g => g.FelhasznaloAzon ==
_azonANTSZ));

        }

        public IQueryable<Kutatasok> MasKutatasoks(int _azonANTSZ)
        {
            return this.Context.Kutatasoks.Where(f => f.Kutato !=
_azonANTSZ && f.Statusz != "SZERKESZT");
        }

        public IQueryable<Kutatasok> MasKutatasoksOrvos(int _azonANTSZ)
        {
            return this.Context.Kutatasoks.Where(f =>
f.KutatasResztvevoks.Where(h=>h.FelhasznaloAzon == _azonANTSZ).Count()==0
&& f.Statusz != "SZERKESZT");
        }

        public void InsertKutatasok(Kutatasok kutatasok)
        {
            this.Context.Kutatasoks.InsertOnSubmit(kutatasok);
        }

        public void UpdateKutatasok(Kutatasok currentKutatasok)
        {
            this.Context.Kutatasoks.Attach(currentKutatasok,
this.ChangeSet.GetOriginal(currentKutatasok));
        }

        public void DeleteKutatasok(Kutatasok kutatasok)
        {
            this.Context.Kutatasoks.Attach(kutatasok);
            this.Context.Kutatasoks.DeleteOnSubmit(kutatasok);
        }

        public IQueryable<KutatasResztvevok> GetKutatasResztvevoks()
        {
            return this.Context.KutatasResztvevoks;
        }

        public void InsertKutatasResztvevok(KutatasResztvevok
kutatasResztvevok)
        {

this.Context.KutatasResztvevoks.InsertOnSubmit(kutatasResztvevok);
        }

        public void UpdateKutatasResztvevok(KutatasResztvevok
currentKutatasResztvevok)
        {

this.Context.KutatasResztvevoks.Attach(currentKutatasResztvevok,
this.ChangeSet.GetOriginal(currentKutatasResztvevok));
        }
```

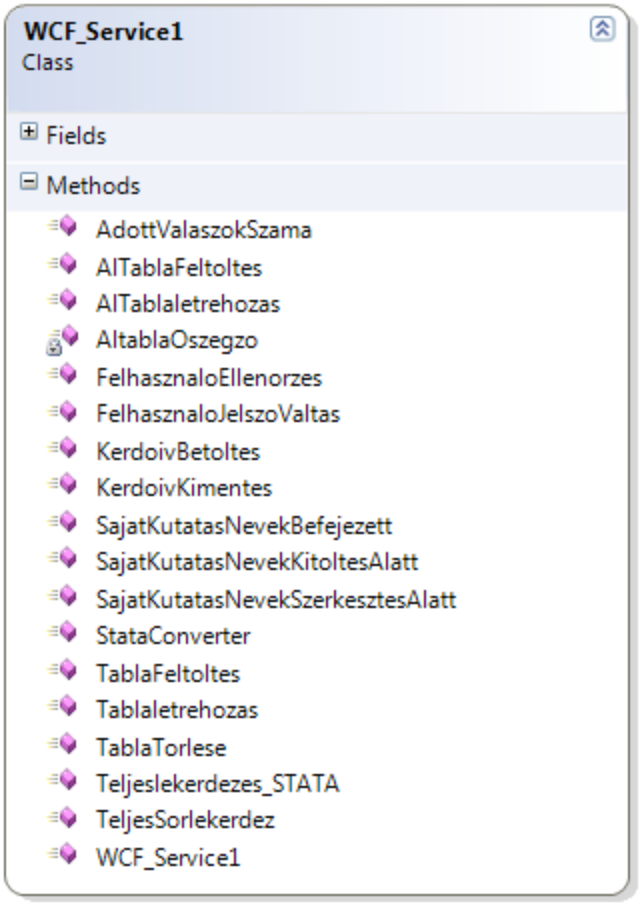
```
        public void DeleteKutatasResztvevok(KutatasResztvevok
kutatasResztvevok)
        {
            this.Context.KutatasResztvevoks.Attach(kutatasResztvevok);

this.Context.KutatasResztvevoks.DeleteOnSubmit(kutatasResztvevok);
        }
    }
}
```

Az osztályt a visual studio automatikus kódgenerátora hozta létre, majd a feladatnak megfelelően lett lekérdezésekkel bővítve, ilyen lekérdezések a `GetOrvosoks`, `SajatKutatasoksSZerk`, `SajatKutatasoksKitolt`, `SajatKutatasoksBefejezett`, és a `ResztveszKutatasoksKitolt` ezzen lekérdezések megvalósítására azért van szükség, hogy ne keljen minden felhasználónak a teljes adatbázist lekérdezni, csupán a szükséges adattáblák kerüljenek letöltésre.

WCF (Windows Communication Foundation)

Ez a mappa tartalmazza a wcf-fel kipublikált osztályt, ezzel a technológiával vannak kipublikálva azok a metódusok melyek visszatérési értéke nem `IQueryable<>` tipusú így nem lehet RIA segítségével kipublikálni.
Itt kelett megvalósítani a dinamikus adatbázis műveleteket, mintpéldául a táblalétrehozást az adatfeltöltést és lekérdezést továbbá az adatbázistörlést.

Az osztály hosszára és felépítésére való tekintettel ezuttal nem ismertetem egyben az osztály teljes tartalmát csupán az osztály szerkezetét és a fontosabb metódusok céljait.



bool FelhasznaloEllenorzes (string Nev, string jelszó)

Ezzel a metódussal tudjuk eldönteni egy felhasználónévről és jelszóról, hogy az hozzá tartozik e, egy korábban bejegyzet felhasználóhoz. Ehez felhasználja a security osztályt így az adatbázisban nem a jelszó hanem annak egy kódolt alakja van tárolva.

bool FelhasznaloJellszovaltas (string Nev, string regijelszo,string Ujjelszo)

Itt tudjuk megváltoztatni a saját jelszavunkat, de csak abban az estben ha megtudjuk adni a mostani jelszavunkat.

List< List<string[]>>KerdoivBetoltes (string kutatasnev)

A kérdőívben szereplő, kérdéseket osztályokká alakítva 17 tulajdoságot tartalmaznak, ezek felépítését később tárgyalom, most elég annyit tudnunk rólluk, hogy az osztályok

megvalósítják a Load metódust mellyel beálíthatjuk egy korábbi mentésből az állapotukat. Ezen állapotok fáljból történő deserializálását valósítja meg ez a metódus.

```
[OperationContract]
public List<List<String[]>> KerdoivBetoltes(string _azonNev)
{
   Stream stream = null;
   try
   {
    stream = new FileStream(path + _azonNev + ".bin",
    System.IO.FileMode.Open);

    IFormatter formatter = new BinaryFormatter();
    List<List<String[]>> result =
    (List<List<String[]>>)formatter.Deserialize(stream);
    stream.Close();
    return result;
   }
   catch
   {
    new ArgumentException("Hiba a" + _azonNev + ".bin nevű file
megnyitásánál!");
    return null;
   }
}
```

A `[OperationContract]` jelzővel azt határozzuk meg, hogy az adot metódust kipulikáljuk a kliens számára helyes WCF kapcsolat esetén.

A metódus kódjában látható, hogy a kutatás nevével megegyező nevű binárisan serializál file-t keres és azt alakítja vissza a megfelelő struktúrába. Hiba esetén pedig jelentést küld a felhasználónak.

void KerdoivKimentes (List< List<string[]>> adat)

Ez a metódus a fordított párja az előzőnek, vagyis a Kérdések osztály listájának az állapotát képes kimenteni egy fájlba.

```
public void KerdoivKimentes(string _azonNev, List<List<String[]>> _kerdoiv)
  {

   Stream stream = new FileStream(path + _azonNev + ".bin",
System.IO.FileMode.Create);
   IFormatter formatter = new BinaryFormatter();
```

```
    formatter.Serialize(stream, _kerdoiv);

    stream.Close();
   }
```

A kutatáshoz tartozó bináris file változása esetén felülkel írnunk a már korábban létezet file-t, így nincs szükség az esteleges ütközés keressére.

string StataConverter(string _ertek)

A gyűjtött adatok STATA formátumban történő kimentéséhez szükséges némi átalakításra adatokban és a szerkezetben is, ez a metódus az adatok átalakítását végzi el:

```
public string StataConverter(string _ertek)
        {
            string result;

            switch (_ertek)
            {
                case "Igen":
                    {
                        result = "1";
                        break;
                    }
                case "Nem":
                    {
                        result = "0";
                        break;
                    }
                case "Nem tudja":
                    {
                        result = "99";
                        break;
                    }
                case "Nincs adat":
                    {
                        result = "9999";
                        break;
                    }
                default:
                    {
                        result = _ertek;
                        break;
                    }
            }

            return result;
        }
```

A kódból is jól látszik, hogy a statában bizonyos konstansokhoz számbeli értékeket kell rendelnünk, ezek az érték párok pedig a következők:

Igen => 1 Nem => 0 'Nem tudja' => 99 'Nincs adat' => 9999

Ezen értékpárok a most STATA-t használóknál megszokottak így a kezelésük szükséges.

Tablafeltoltes, Tablatorles, Tablaletrehozas

Ezek a metódusok a szokásos sql parancsok segítségével valósítják meg a szerepüket, a műveletekhez minden esteben szükség van a tábla nevére és a megfelelő sql parancsra melyet string formájában adunk át. Tovább az alábbi sablonnak kell megfeleljen a metódus:

```
datacontext = new DataContext(constr);
sqlConnection = new SqlConnection(constr);
sqlConnection.Open();

  SqlCommand sqlCommand = new SqlCommand(SQLPARANCS, sqlConnection);

sqlCommand.ExecuteNonQuery();

sqlConnection.Close();
sqlConnection.Dispose();
```

Itt látható, hogy elösször létrehozzuk a kapcsolatot majd végrehajtjuk a parancsot legvégül zárjuk a kapcsolatot.

string Teljeslekerdezes_STATA (string Nev)

Ez a metódus szolgál a STATA formátumban történő kimentésért.

A STATA formátum elég általános szabályokra épül az első sora az oszlopok nevét tartalmazza tabbal elválasztva, majd a további sorokban az egyes rekordok helyezkednek el, elemeik szintén tabbal vannak elválasztva.

A formátum egyetlen igazi különlegessége a már korábban említet értékpárok, melyek cseréjéről gondoskodnunk kell.

Ezzel ismertettem a server oldal fontosabb osztályait és azok metódusait.

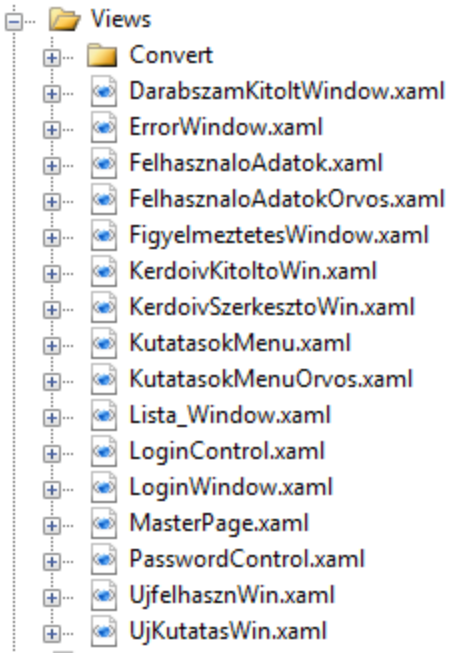
KLIENS OLDAL

A kliens oldal létrehozására a már többször is említett SilverLight3-mat használom, gazdag funkcionalitása és korábban említett előnyei miatt.

A kliensoldali célkitűzés mindenképpen az volt, hogy a felület jól kezelhető és átlátható legyen, és emellett megfeleljen minden szükséges igénynek.

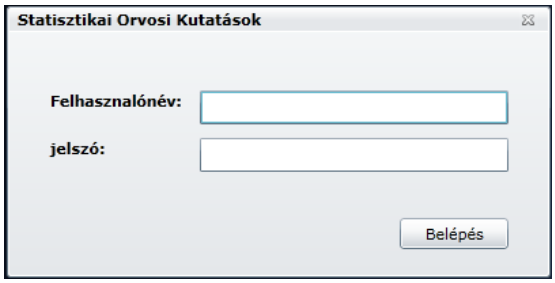
A kliens oldali alkalmazás legfontosabb feladata a felhasználóval való kommunikáció, tehát grafikus megjelenítést ad a serveroldali függvények használatához.

A felület UserControlokból és úgynevezett ChildrenWindows-okból épülnek fel, ezek külön-külön helyezkednek el az egyes file-okban a kliens oldali Views mappában.



BEJELENTKEZÉS

A honlapra történő navigáláskor a bejelentkezési felületet látjuk meg



Mivel a rendszerbe nem megengedet a nem azonosított felhasználók belépése, így a rendszer a helyes felhasználónév és jelszó megadásáig nem engedi a belépést.
Az azonosítás után kiderül, hogy kutatóról vagy orvosról van e szó, és a felület ennek megfelelően engedélyezi a műveleteket.

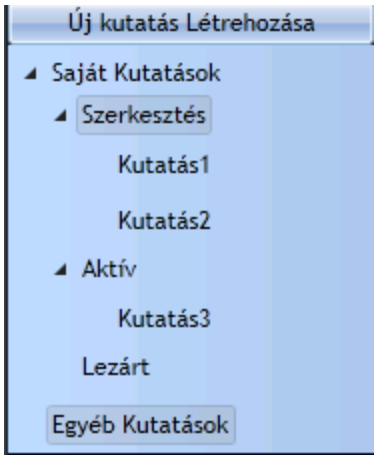
A KÖZÖS FELÜLET

A felületnek egységes keretet ad a MasterPage.xaml mely az alapfelületek közötti koordinálást és a kijelentkezést biztosítja.

A felhasználó két lap között navigálhat, a Kutatások fül alatt kezelheti kutatásait, míg a felhasználóadatok fül alatt kezelheti a felhasználókat.
A fejrész tartalmazza továbbá az alkalmazás logóját és nevét. Majd a jobb oldali sarokban emlékeztetőül szerepel a bejelentkezet felhasználó neve és ANTSZ kódja.

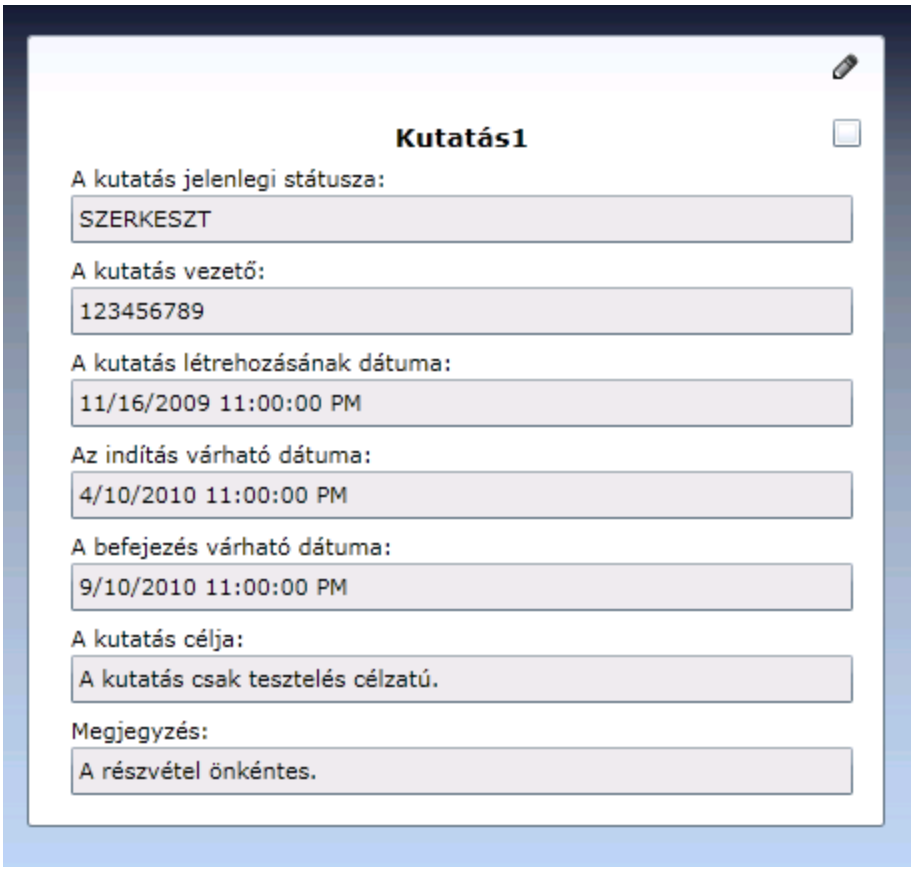
A KUTATÁSOK SZERKESZTÉSÉNEK FELÜLETE

A kutatások kiválasztásához egy fa szerkezet áll rendelkezésünkre, ennek előnye, hogy az alkalmazás csak azon kutatásokat tölti, le a szerverről melyekre ténylegesen szükségünk van, és a felhasználónak mégis rögtön egyértelmű a felület kezelése.



(A felhasználó felületen látszik, hogy a bejelentkezet kutatónak három kutatása van, ebből kettő még csak tervezési fázisban egy pedig már Aktív állapotban.)

A kiválasztott kutatás adatait a felület részletesebben is megjeleníti:



(A felület igazítása nincs befejezve)

VÉGREHAJTHATÓ MŰVELETEK

A kutatások fázisától függően jelennek meg a kutatásokon végrehajtható műveletek. Ezek a műveletek az alábbiak:

Kutatás Aktiválása – Ez a gomb csak akkor elérhető, ha a kijelölt kutatás, szerkesztés fázisban van, aktiválás esetén a felület még egyszer rákérdez a művelet végrehajtására és figyelmezte, hogy nincs lehetőség visszavonásra. Ha megerősítjük a műveletet akkora

kutatás státusz megváltozik és Aktív állapotba kerül. Ebben az állapotban pedig megkezdődhet az adatgyűjtés.

Kutatás befejezése – Ez a gomb csak akkor elérhető, ha a kijelölt kutatás, aktív fázisban van, aktiválás esetén itt is biztonsági figyelmeztető kérdés jelenik meg. Folytatás esetén pedig a kutatás Befejezett állapotba kerül.

Kutatás törlése – Bármilyen fázisba lehetséges a kutatást törölni, de mivel teljes adatvesztéssel jár így figyelmeztetést kapunk az adatmentésre.

Kérdőív szerkesztése - Ez a lehetőségünk csak a szerkesztés állapotban lévő kutatások esetén van meg, szerkesztés esetén az alkalmazás betölti a korábban elmentet kérdőívet és ezen enged módosításokat a szerkesztő felülettel.

Gyűjtött adatok kimentése – Már Aktív állapottól elérhető, az időközbeni adatmentés céljából. Befejezet fázisban csak ez a menüpont és a kutatás törlése menüpont érhető el.
Válaszadás – Ez a lehetőség csak a bejelentkezett orvosnak jelenik meg, de neki csupán ez az egy lehetőségük van. Válaszadás esetén megjelenik a kérdőív,de ezúttal nincs lehetőségünk a módosítására, pusztán válaszokat adhatunk.

A KÉRDŐÍV SZERKESZTŐ FELÜLETE

Az alakalmazás legfontosabb pontja, a kérdőív szerkesztő felülete, ennek a felületnek könnyen használhatónak és magától értetődőnek kell lennie, de emelet a kérdésekről rengeteg információt meg kell tudnunk adni. A kérdések tulajdonságaik alapján hat tipusba lettek besorolva.

A kérdés tipusok ismertetése előtt kiemelném a kérdések közös tulajdonságait.

Név – A kérdések nevéből generálódnak a kutatáshoz létrejövő adatbázistábla oszlopai.

Sorszám – Főleg a külső dokumentációhoz és a későbbi kérdés azonsosításhoz fontos. Az alkalmazás működésében nincs jelentősége.
(Egy-egy kérdésnél ez az első baloldali elem.)

Csoport – Ezzel az értékkel lehet összerendelni a kérdések egy csoportját halmazokba, az alkalmazásban erre az érétékre hivatkozhatunk, hogy beállíthasuk a kérdések függőségét.

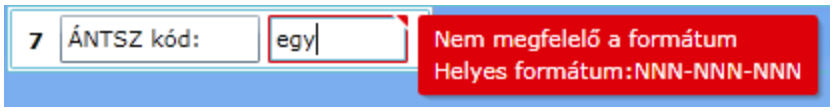
Kérdésszövege – Ebben van megfogalmazva a valódi kérdés a páciensre vonatkozóan.

(Megjelenítésben ez a balról a második mező.)

Érték – Ide kerül a kérdésre adható válasz. Azt, hogy a válasz elfogadható e az dönti el, korábban milyen beállításokat adtunk meg a kérdésről.

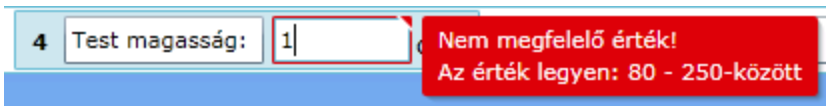
A KÉRDÉSEK TIPUSAI

szöveges kérdés - Olyan kérdések melyek alapesetben bármilyen választ elfogadnak, és a válaszokat szövegként értelmezik. Bizonyos regex stilusú szükítések alkalmazásával elérhető továbbá, hogy szűkítsük az adható válaszlehetőségeket. például az NNN-NNN-NNN formátum megadásával megadhatjuk, hogy ÁNTSZ stílusú választ fogadjon csak el.

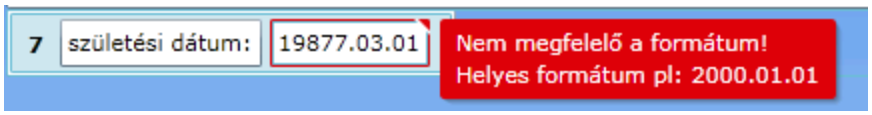


Látható, hogy a felület nem fogadja el a helytelen választ és erről hibajelentéssel értesíti is a felhasználót.

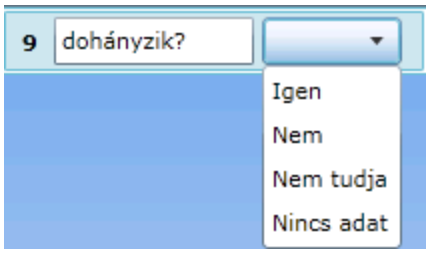
számkérdés – Olyan kérdések melyek válaszként csak számot fogadnak el. Továbbá megadható a kérdéseknél az elfogadási határt is melyel finomíthatjuk a megadható válaszokat, ezzel elkerülve gépelésből vagy félreértésből származó hibákat. Az ilyen tipusú kérdéseknél megadható továbbá a mértékegység is, gondolva itt a metrikus értékek szerepére.



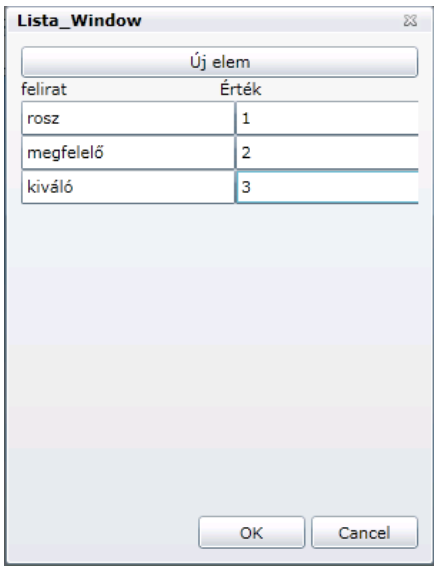
Dátum – Olyan speciális szöveg tipusú kérdés melynél az elfogadható válaszok formátuma kötött és a formátum az alábbi regexel van megadva: `[12][90][0-9][0-9].[01][0-9].[0123][0-9]` ezzel a megadással biztosítom, hogy az ilyen tipusú kérdés a válasz mezőben csak dátum formátumot fogadjon el.



Igaz/hamis – Az első speciális értékmezővel rendelkező kérdéstipus az igaz/hamis tipusú kérdések. Az értékmezője nem egy szövegmező(textbox) mint a korábbiakban hanem egy legördülő lista. Ilyen kérdés alakalmazandó akkor ha eldöntendő kérdést akarunk feltenni pl: dohányzik?
Azonban a legördülő listában az igényeket figyelembevéve nem csak az igen és nem válasz szerepel hanem a nincs adat és a nem válaszol is, mivel ez a két opció minden esetben lehetőség kell, hogy legyen.



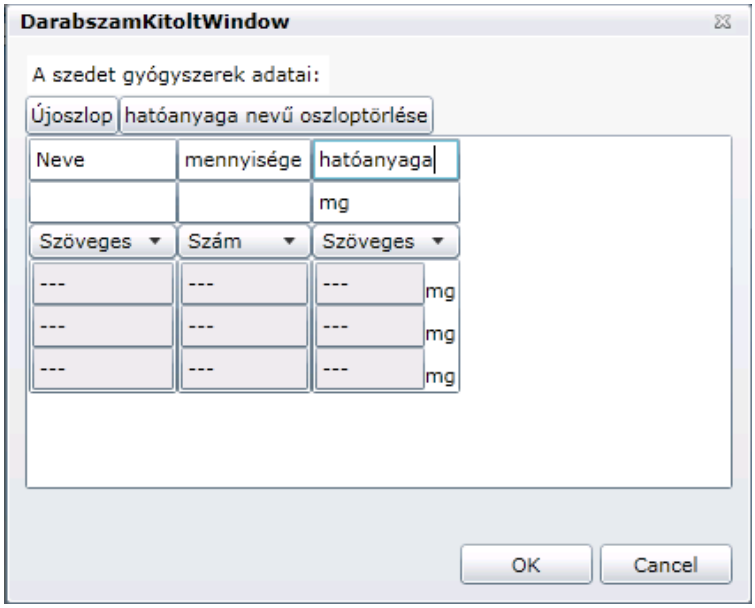
Lista – Az előzőnél is speciálisabb kérdés tipus. Itt is legördülő lista jelenik meg a válasz adáshoz, de ebben az esetben a kérdőív szerkesztésénél lehetőség van a választható lista elemek megváltoztatására. Ilyen kérdés tipusra lehet szükség például a jellemezze a közérzetét kérdésnél ahol a lehetséges válaszok rossz, megfelelő, kiváló.



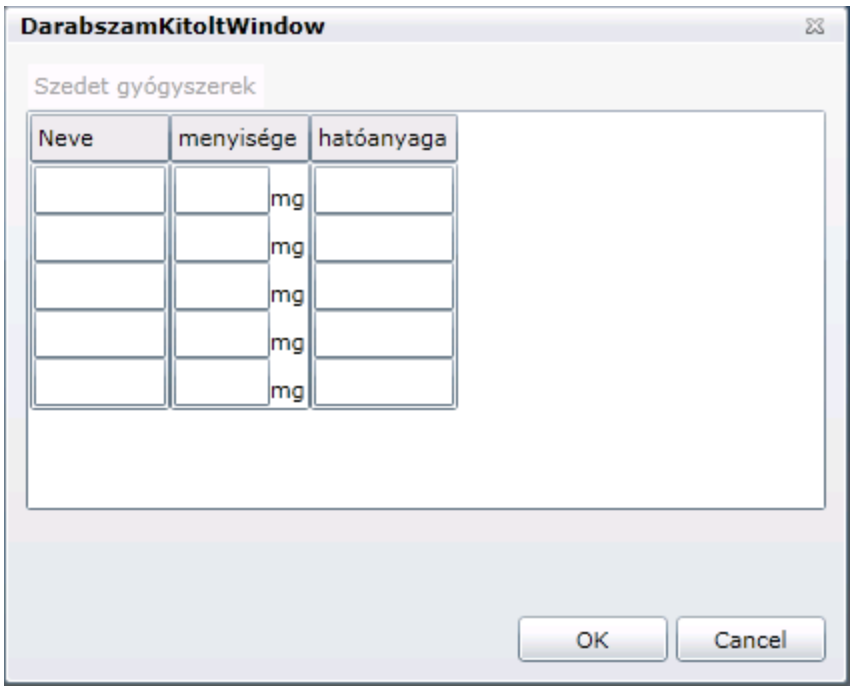
A lista elemeit beállító panel

Darabszám – Ez a legspeciálisabb kérdés tipus, mivel ez a kérdés tipus magában tartalmaz egy altáblázatot melynek sorait az adott válasz adja, és kitöltési fázisban jelenik meg érték változáskor.

Az altáblázat oszlopait szerkesztési állapotban állíthatjuk be, ekkor adhatók meg az oszlopok nevei és az itt elfogadható válaszok tipusai és mértékegységei.



Az altábla beállításának a panelje.



Az altábla kitöltési nézetben. 5-ös válasz esetén

A KÉRDŐÍV ÖSSZEÁLLÍTÁSA

A kérdőív összeállításához egy igen kényelmes szerkesztő felület áll rendelkezésünkre, melynek baloldalán helyezkednek el a kérdétipusok melyeket DragAndDrop (fogd és vidd) módszerrel helyezhetünk el a kérdőív felületén. Az aktuálisan kijelölt kérdés tulajdonságait pedig a jobboldalon állíthatjuk be.
A kérdések egyéni beállítsa után érdemes, beállítani a logikai összefügéseket, vagyis azt, hogy mely kérdések azok amiktől logikai függésbe van egy adot csoport. Ilyen függőség beállítása esetén csak akkor jelennek meg az alárendelt kérdések ha a kapott válasz megfelel a függőség beállításainak.

ÖSSZEFOGLALÁS A FELÜLETRŐL

A kliensfelület a bejeltkezésünktől kezdve tudja, hogy mely műveletek azok amiket jogunkban áll végrehajtani és csak ezeket jeleníti meg.
Amenyiben kutatóként léptünk be a felület folyamatosan figyelmeztet és tájékoztat az olyan műveletek esetén melyek hatása végleges és nem visszavonható. Ezzel megakadályozva a véletlet műveleteket.
A kérdőív szerkesztésére egy kézenfekvő felületet biztosít melynek kezelése magától értetődik, és nem ad lehetőséget hibás művelet végrehajtására.
Orvosként belépve alapesetben a felület csupán annyi információt szolgáltat számunkra mely a munkánkhoz elegedhetetlenül szükséges, de lehetőséget ad az egyéb kutatásokról történő információ szerzésre is.
Nem enged hozzáférést a kutatások szerkesztéséhez, csupán a válaszadásra van lehetőségünk. Válaszadásnál a kérdések csak a specifikácóban meghatározot értékeket fogadják el, és az ott beállítottaknak megfelelően viselkednek. Hibás vagy elégtelen válaszadás esetén nem engedi, hogy a válaszainkat felvigyük az adatbázisba, így megakadályozva azt, hogy az adatbázis hibás adatokat tartalmazzon. A felület azonban nem csak megakadályozza a hibákat hanem, a hiba okáról részletes hibaüzenettel is szolgál.

TOVÁBB FEJLESZTHETŐSÉG

A program továbbfejlesztésének első lépése mindenképpen a felület tovább egyszerűsítését és a használhatóság növelését célozza meg.
A teszteléseknél felmerült az igény a szerkesztő felület tisztán billentyűzetről történő irányítására, ez a tervezési fázisban még nem volt igény, így az alkalmazás nincs rá felkészítve. Ennek pótlása bizonyos felületek újratervezését és egyes kódok újra gondolását igényli.
A felület használata ugyan egyértelmű, de a félreértések elkerülésért és az új felhasználó gyorsabb betanulásáért érdemes lenne az alkalmazást súgóval kiegészíteni.
A Microsoft termékei között ugyan szerepel, olyan eszköz melyek könnyedén készíthetünk videó súgót, de tekintve, hogy korábban cél volt a minél alacsonyabb sávszélesség megőrzése, így ez a technológia nem célszerű ebben az esetben. Célszerűbb egy a feladatok lépéseit összefoglaló leírást készíteni, és a műveletekhez felugró súgókat illeszteni.
Az alkalmazás ettől a két hiányosságtól eltekintve megvalósítja a kívánt elvárásokat, így nincs szükség a tovább fejlesztésére.

Azonban a kutatásokhoz lehetséges még segítő software-eket készíteni. A következő terület, amit mindenképpen érdemes támogatni az a kutatási adatok elemzése. Mely a mai megoldások szerint konzolosan történik jelentősen kibővítet sql parancsokkal, a STATA program segítségével. Ez a felület egyrészt kényelmetlen másrészt nem túl hatékony. Ezért célszerű lenne egy alapvetően grafikus alapú rendszerrel a helyettesítése.

A korábban bemutatott alkalmazás továbbfejlesztésének egy másik lehetősége, ha kiterjesztjük a felhasználásának a területeit és nem csak orvosi kutatásokra, de bármilyen adatgyűjtéssel foglalkozó tevékenységre alkalmassá tesszük.

Köszönetnyilvánítás:

Az alkalmazás felépítésében az igények felmérésében és az orvosi szemszögből való megközelítésben nagy segítségemre volt Dancs Péter a debreceni egyetem orvos tudományi hallgatója. Akinek ezúton is sok sikert kívánok az élethez és köszönöm a segítségét.
És bár szakmai segítséget nem nyújtott, de van még valaki, aki nélkül soha nem értem volna célt, ö pedig édesanyám. Neki ezúton is köszönök minden támogatást az életben.

Irodalomjegyzék

Orvosi forrás:

- Ádány Róza

  Megelőző orvostan és népegészségtan

  Medicina könyvkiadó zrt. Bp 2006

Informatikai könyvek:

- Laurence Moroney

  Microsoft Silverlight 3 - első könyv

  Szak Kiadó Bp 2009

- George Shepherd

  Microsoft ASP.NET 2.0 lépésről lépésre

  Szak Kiadó Bp 2009

Honlap:

- **www.Devportal.hu**
- **http://silverlight.net/learn/**
- **http://mtaulty.com/CommunityServer/blogs/mike_taultys_blog**

Orvosi támogató az epidemiológia területéről, és alkalmazás tesztelő:

- **Dancs Péter**