

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

Adatok gyűjtésére és feldolgozására szolgáló online rendszer a
felső oktatásban

DIPLOMADOLGOZAT

Témavezető:
Dr. Szántó Zoltán,
Egyetemi adjunktus

Végzős hallgató:
Csegzi Hunor

2023

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ



UNIVERSITATEA
SAPIENTIA

Sistem pentru elaborarea rapoartelor pentru asigurarea calității
în învățământul superior

LUCRARE DE DIPLOMĂ

Coordonator științific:
Dr. Szántó Zoltán,
Șef Lucrări

Absolvent:
Csegzi Hunor

2023

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION**



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

System to collect and elaborate data in superior education

BACHELOR THESIS

Scientific advisor:
Dr. Szántó Zoltán,
Assistant Professor

Student:
Csegzi Hunor

2023

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Programul de studii: Informatică

Viza facultății:

LUCRARE DE DIPLOMĂ

Coordonator științific:

Dr. Szántó Zoltán

Candidat: **Csegzi Hunor**

Anul absolvirii: 2021

a) Tema lucrării de licență:

Sistem pentru elaborarea rapoartelor pentru asigurarea calității în învățământul superior

b) Problemele principale tratate:

Proiectarea și dezvoltarea unei aplicații web pt. uz intern, care ajută la gestionarea rapoartelor.

c) Desene obligatorii:

- Diagramele UML a aplicației
- Manual de utilizare detaliat

d) Softuri obligatorii:

- Modul bazat pe tehnologia RDLC report în C#.NET
- Interfața utilizator
- Baza de date și un layer pt. gestionarea datelor

e) Bibliografia recomandată:

- Kopeć, Adrian, Justyna Bala, and Anna Pięta. "WebGL based visualisation and analysis of stratigraphic data for the purposes of the mining industry." *Procedia Computer Science* 51 (2015): 2869-2877.
- Renders, Steven. *Microsoft Dynamics NAV 2015 Professional Reporting*. Packt Publishing Ltd, 2015.
- Andric, Milan, et al. "Web application as a support system for records of working time, monitoring business processes and activities of company employees." (2016): 514-519.
- Zhou, Yawen. "The network system of scientific research management in local undergraduate colleges and universities based on the web platform." *Big Data Analytics for Cyber-Physical System in Smart City: BDCPS 2019, 28-29 December 2019, Shenyang, China*. Springer Singapore, 2020.

f) Termene obligatorii de consultații:

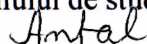
g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 313

Primit tema la data de: 31.03.2021

Termen de predare: 27.06.2023

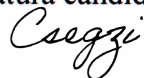
Semnătura Director Departament

Semnătura responsabilului
programului de studiu



Semnătura coordonatorului

Semnătura candidatului



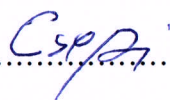
Declarație

Subsemnatul/a ...Csegzi Hunor....., absolvent(ă) al/a specializăriiInformatică....., promoția...2021... cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu-Mureș

Data: 2023.06.12

Absolvent

Semnătura..........

Kivonat

Szakdolgozatomban az egyetemi oktatásban való kimutatások kidolgozása a fő szempont, a minőség biztosítása érdekében. Egy webes alkalmazás fejlesztésébe kezdtem, melyben kimutatások lekérdezése, készítése lehetséges, adatokat tartalmazó táblázatokat módosítani, adatokat törölni, felülírni, illetve Excel formátumban feltölteni. A szoftver segítséget nyújt bárkinek, akinek hozzáférése van ehhez, támogatja az adatok bevitelét, és a megjelenített táblázatokból átlátható adat-halmazok rendezését.

A fejlesztés előtt számos alternatív megoldást vettem számításba, viszont végkövetkeztetésképpen az általam választott technológia bizonyult a leghatásosabbnak. Ez a technológia az RDLC report, mely kezdetben számomra is ismeretlen volt, ezáltal a szakdolgozatom fejlesztési folyamata alatt, és megírása során tanultam meg, és fedeztem fel egyaránt. E megoldás számos előnye közül a legfontosabb a könnyű szerkeszthetőség, melyet dinamikusan XML formátumban képes szerkeszteni, kivitelezni, de nem utolsósorban az általam választott rendszer csomagokkal könnyebben végrehajtható.

A rendszer fejlesztésének irányt adott a kimutatásokat végző személy munkájának megkönnyítése. A táblázatok kidolgozása a múltban is szükséges volt, és továbbra is szükséges lesz, ezért a szoftverem fejlesztése, megírása során szem előtt tartottam mindvégig, fontos volt az időtállóság, hogy a projekt hosszú távon, minimális fejlesztéssel fenttartható és működőképes maradjon. A kutatás során szerzett információkkal és a megcélzott felhasználó igényeit és kéréseit figyelembe véve készítettem el a rendszert, mivel a legfontosabb, hogy a felhasználói közösség elfogadja, és szívesen használja azt. Ez a szolid, analitikus platform leegyszerűsíti az oktatásban részt vevő tanárok mindennapi feladatait is, de ugyanakkor egy szimpla, letisztult, használható felületet is biztosít egyetemi évek alatt.

Az egyszerűséget és hatékonyságot szem előtt tartva készítettem el a dolgozatomban, melynek fő funkcionalitása a kimutatások kidolgozása, egy olyan felhasználói felületbe ágyazva, mely megfelel a jelenleg érvényben levő tervezési feltételeknek, ahol az illetékes felhasználó akár első ránézésre átlátja a szoftvert, és könnyedén tájékozódhat a kívánt funkció eléréséhez. Így az általam megtervezett és kivitelezett rendszer, webalapú egy felhasználó-barát környezetet biztosít, úgy a klienseknek, mint a tervezőnek is.

Kulcsszavak: RDLC, kimutatás, web

Rezumat

În lucrarea de diplomă am dezvoltat un software de elaborarea rapoartelor pentru asigurarea calității în învățământul universitar. Am dezvoltat o aplicație web, pentru interogarea și crearea rapoartelor, introducerea, modificarea și ștergerea datelor, suprascrierea și importarea acestora din format Excel. Softul vine în ajutorul persoanelor care se ocupă cu asigurarea calității la universitate.

Înainte de dezvoltare, am luat în considerare o serie de soluții alternative, însă tehnologia pe care am ales-o s-a dovedit a fi cea mai eficientă. Aceasta este tehnologia RDLC, care mi-a fost inițial necunoscut și mie, așa că am învățat-o și am descoperit-o de-a lungul procesului de dezvoltare al proiectului. Cea mai importantă dintre numeroasele avantaje ale acestei soluții este realizarea și editarea foarte ușoară a tabelelor.

Dezvoltarea sistemului a fost ghidată pentru facilitarea creării rapoartele. Să se realizeze foarte ușor elaborarea și modificarea tabelelor. De asemenea era important ca proiectul să poată fi întreținut și modificat ușor pe termen lung. Am creat sistemul cu informațiile obținute în timpul cercetării și luând în considerare nevoile și solicitările utilizatorului țintă, deoarece cel mai important lucru este să fie acceptată și folosită de către comunitatea de utilizatori. Această platformă solidă, analitică, simplifică, de asemenea, sarcinile de zi cu zi ale cadrelor didactice implicate în educație, dar oferă și o interfață simplă, curată și utilizabilă în anii de facultate.

Mi-am pregătit lucrarea de diplomă având în vedere simplitatea și eficiența, a cărei funcționalitate principală este elaborarea rapoartelor cuprinsă într-o interfață care îndeplinește condițiile de proiectare aplicabile în prezent, în felul în care utilizatorul poate înțelege softul dintr-o privire și îl poate naviga cu ușurință de a accesa funcția dorită. Astfel, sistemul pe care l-am dezvoltat oferă un mediu ușor de utilizat pentru clienți.

Cuvinte de cheie: RDLC, elobarare, web

Abstract

In my dissertation, I developed a system to collect and elaborate data in superior education to ensure quality. I started to develop a web application, in which it is possible to create statements, modify tables containing data, delete, overwrite data, and upload them in Excel format. The software helps anyone who has access to it, enter data, and organize transparent data sets from displayed tables.

Prior to the development, I considered several alternative solutions, but in conclusion, the technology I chose proved to be the most effective. This technology is the RDLIC report, which was initially unknown to me as well, thus I learned and discovered it both during the development process of my dissertation and during the writing process. The most important of the many advantages of this solution is the easy editing, which can be dynamically edited and implemented in XML format, but last but not least it is easier to implement with the system packages I have chosen.

The development of the system was guided by opinions who has to elaborate data to create reports. The elaboration of data has been, and will continue to be necessary, so I kept it important throughout my software development to keep the project sustainable and functional in the long run with minimal development. I created the system with the information obtained during the research and taking into account the needs and requests of the target user, as the most important thing is that the user community accepts it and is happy to use it. This solid, analytical platform also simplifies the day-to-day tasks of teachers involved in education, but provides a simple, clean, usable interface during the years.

With the simplicity and efficiency in mind, I have written my dissertation, the main functionality of which is the elaboration of reports, embedded in a user interface that meets the current design conditions, where the competent user can understand the software and easily find the desired information to access this function. Thus, the system and website I have designed and implemented provides a user-friendly environment for both clients and the designer.

Keywords: RDLIC, elaboration, web

Tartalomjegyzék

1. Bevezető	10
1.1. Téma választás	10
2. Technológiai áttekintés	12
2.1. Beépített szoftverek, könyvtárak	13
2.1.1. Visual Studio	13
2.1.2. SQL Server	15
3. A rendszer specifikációja	16
3.1. Felhasználói követelmények	17
3.2. Rendszerkövetelmények	17
3.3. Szoftverek választása	18
4. Tervezés	20
4.1. Kimutatás tervezése:	22
4.2. Adatbevitel tervezése:	23
4.3. Adatbázis tervezése:	25
5. Kivitelezés	26
5.1. Bejelentkező oldal	26
5.2. Főoldal	27
5.3. Diákok oldal	28
5.4. Diák jelentés oldal	29
5.4.1. RDLC Report manager implementálása	30
5.5. Diák Excelből való bevitel oldal	31
5.5.1. Szükséges könyvtárak	32
5.5.2. Táblázat összehasonlítás	33
Összefoglaló	34
5.6. Tovább fejlesztési lehetőségek	35
Ábrák jegyzéke	36
Irodalomjegyzék	37

1. fejezet

Bevezető

A projekt egy mindennapi problémára való megoldás, mely hosszú percekbe kerülő kimutatások elkészítését rövidítené le pár kattintásra. Egyetemi hallgatóként megtudtam, hogy évről évre, vagy spontánszerű kérdés következtében szükséges kimutatásokat végezni az egyetem számos információ csoportjáról, melyeket utólag felhasználhatnak a szolgáltatások, munkafolyamatok feljavításában. Ezek az adatok rendszerint Excel, vagy hasonló formátumban vannak tárolva, amely használatával nem mindig egyszerű kimutatásokat létrehozni.

Magam analitikus látásúnak tartom, ezért érdekesnek találtam a felhalmozott adatok feldolgozását, azok szűrését, illetve kialakítását, egy áttekinthető formátumba, amely nem csak, hogy érdekes, de tanulságos következtetés vonható le belőle a szolgáltatások javításának érdekében. A rendszer kialakításában szem előtt szerettem volna tartani az egyszerűséget, így különösebb felkészítés nélkül is használható a program. A kimutatások oly módon legyenek elkészítve, hogy minél kevesebb beállítás módosítással a felhasználó széles körben használni tudja azt, minden fajta adat feldolgozására. Az adatok gyűjtésének érdekében egy adat beviteli módszer volt szükség, hogy az adatokat nagyobb mennyiségben is fellehessen tölteni egyszerre.

Összegzésképpen a dolgozatomban bemutatom a szoftver elkészítéséhez megfelelőnek bizonyult rendszereket, és azok előnyeit és hátrányait, illetve a rendszer követelményeit, majd ismertetem a fejlesztést megelőző tervezési folyamatot, melyben előre kigondolt funkciók kialakítását részletezem, utolsóként meg bemutatom minden funkcionalitást egyenként, részletezve, és illusztrálva. [Zho19]

1.1. Téma választás

Minden bizonnyal sokan mondták már, és mindenkihez eljutott a hír, hogy az adat az új arany. Sokan tartják így, és a múltban volt már rá példa, hogy ebből sokan hasznalt húztak, kijátszottak a rendszert, illetve visszaéltek illetéktelenül hétköznapi emberek információival.

Ezek az információt tartalmazó adathalmazok nagyon gyorsan változnak, és halmozódnak, akár a szemünk előtt, amit mi nem tartunk fontosnak, de a világban igenis fontos szerepet játszódnak. Volt rá példa politikai közegű felhasználás, vagy csak egy egyszerű termék elhelyezés formájában.

A felső oktatásban szereplő adatok tengere hatalmas, évről-évre új diákok iratkoznak be, ezek szakon belüli eloszlása komplex hálózat, vagy akár egy egyetemi év kezdésekor új szak indulhat. Kiértékelések, jegyek halmazát kell a rendszer eltárolja, diploma munkák sokaságát, versenyek eredményeit, ösztöndíjak elbírálását és kiosztását szükséges megjegyeznie ennek a szisztematikusan szervezett platformnak. [KP15]

Az emberek olyan adatokat osztanak meg a közösségi médiában, akarva akaratlanul, amik önmagukban nem sors fordító tényezők, de amikor mindezt több ezer ember csinálja meg, azokat feldolgozva már fontos állapotfelmérő eredményekhez lehet jutni. Köztük lehet akár egy felmérés is, melynek nincs közvetlen rossz hatása, hiszen nem minden adatgyűjtés szolgál arra, hogy valaki hasznot húzzon belőle, hiszen az adatok, illetve vélemények halmozása megtörténhet azzal a céllal is, hogy adott szolgáltatást jobbra tegyenek, felhasználók többségének tetszése szerint változtatni, vagy egy könyvtárat alkotni, amihez a hozzá férők könnyedén leszűrhessek a keresett következtetést.

Fontosnak tartottam még az adatok lekérését, átláthatóságát, mivel az egyetemi oktatásban szereplő adatok halmaza hihetetlenül nagy, a rendszerben szereplő diákok és tanárok közötti kapcsolat egyszerű áttekinthetősége a cél.

Az adatok mellett fontos szerepet játszó tényező az idő. Az idő mindenki számára véges, legyen szó határidő miatt, vagy csupán a gyors probléma megoldó tulajdonság miatt, sokan hajlamosak hibázni, ami nem egy végszó, hiszen hibázni emberi dolog, viszont az emberi mulasztás következtében az adathalmazok szűrése téves, pontatlan következtetéshez juttatja a szorgos embert.

Ezeket összevetve vetettem magam bele a dolgozat elkészítésébe, mely témája a minőség biztosítás az egyetem legfontosabb éves időközönként változó adatok érthető megjelenítése érdekében. A fő szempont mellett teret nyert az adatok módosítása is, hiszen ezek az adatok változnak, és szükséges halmazban, vagy akár egyenként változtatni őket. Szükségesnek éreztem az adatok védelme érdekében egy jelszó alapú beléptető rendszer kialakítását, hiszen nem szeretném, hogy kiszivárognak információk. A regisztráció nem volt fontos, mivel egy adminisztrátor szabályozza, hogy ki léphet be. A kitűzött cél egy bárki számára érthető, letisztult felhasználói felület kialakítása volt, minél barátságosabb, és egyszerű rendszerrel, de ugyanakkor gyors és megbízható.

2. fejezet

Technológiai áttekintés

A témában való információ gyűjtés során több lehetséges megoldásra bukkantam. A fejlesztés alapját képező keretrendszer kiválasztásában szerettem volna nevezőre jutni.

Elsőként a Java tűnt a megfelelő döntésnek, rövid kutatás során egy ingyenes report generáló rendszert találtam, aminek neve i-net Clear Reports, amelynek a nyílt forrású Java az alapja, viszont ez nem támogatott olyan széles körben, mint a Microsoft cég által létrehozott RDLC Report manager. E kiválasztási folyamat eredményeként .NET irányába billent a mérleg, hiszen támogatja az általam használt C# nyelvezetet. A .NET több mint 20 programozási nyelvet támogat, így egy multi-programozói környezet. A .NET kiküszöböli a felesleges kódokat, így kevesebb kódolást igényel a fejlesztők számára. A .NET könnyen hibákat javít, moduláris felépítése szerint a fejlesztők szétszedhetik az alkalmazásokat, weboldalakat, majd kijavíthatják azokat az elemeket, amelyre szükség van és újra összeállíthatják anélkül, hogy végig kelljen menni az összes soron. A Java egy nyílt platform, így a biztonság szempontjából kérdéseket vet fel. Sok biztonsági problémával és jogsértéssel kellett számolni a múltban.

Szakedolgozatomban használt C# egy magas szintű programozási nyelv a C++-hoz képest. A C# automatikusan futtatja a memória-kezelést, míg a C++-ban manuálisan kell kezelni a memóriát. A C#-nak nincsenek komplex funkciói, szimpla hierarchiával rendelkezik, és egyszerű megérteni a C++-hoz képest, melynek nyelvezete komplex. A C# segített a tömbök kötött ellenőrzésében, valamint fordítói hibákat és figyelmeztetéseket adott, anélkül, hogy komoly hibákat követtem volna el, melyeket a C++ megenged. A C# egy sokkal védettebb platformot biztosított.

A fejlesztői környezet kiválasztása két lehetséges opció közül történt, mely során a Visual Studio javára döntöttem, az Eclipse fejlesztői felület helyett. Az Eclipse integrált fejlesztői környezet egy híres integrált Java fejlesztői környezet, de újabban körbefogja a C / C ++, a JavaScript / TypeScript, a PHP stb. Nyílt forráskódú platform, akárcsak a Visual Studio, ingyenes hozzáférhetőséget biztosít a programozók számára. Az Eclipse a tervezett felhasználási esetre vonatkozó csomagokat is kínál. MyEclipse továbbfejlesztett képességeket kínál a Visual Studio Enterprise szolgáltatásban. Az Eclipse szolgáltatása azonban jóval lassúbb, mint a Visual Studio parancsai, sokkal több késést észlelni benne. A Visual Studio támogatja az általam használt C# felületet és a .NET keretrendszert, valamint a Java, Python nyelvezetet is akár.

2.1. Beépített szoftverek, könyvtárak

A szoftver alapját képezi az RDLC report manager könyvtár, mely a Microsoft által fejlesztett bővítmény, melynek függelékei elengedhetetlenek. Programozási környezetként a Visual Studio 2019 Community-t használtam, melyet összekötöttem az SQL Serverben általam létrehozott adatbázissal.

2.1.1. Visual Studio

A Microsoft Visual Studio egy integrált fejlesztési környezet, melyet a Microsoft gyártott. Különböző programozási nyelveket támogat, mint például a Visual C# Visual C++, Visual C#, Visual F#, Visual Basic.NET. A programozók és a fejlesztők számára széles körű felületet biztosít. A hagyományos windowsos alkalmazások, applikációk mellett a webes- és konzol alkalmazások továbbfejlesztésére is lehetőségünk adódik felhasználásával. Úgy Windows, mint MacOS programozási platformot biztosít. A .NET keretrendszerén belül új adatbázis technológia nyújt segítséget az adatok gyűjtésében és feldolgozásában. A szakdolgozatom segítségéhez az alábbi lépéseket használtam: megtervezni és tudatosítani a felhasználói felületet, megtervezni és beállítani a feltételeket, megírni a kódot, tesztelni és nem utolsósorban, hibát futtatni.

.NET:

A .NET a Microsoft által kibocsájtott, a programozási nyelvtől kissé eltérő és független szolgáltatást, fejlesztést nyújtó rendszer. Egy új technológiát jelent az adatok számára, egy megfelelő tervezési helyet és futtatási szolgáltatást a különböző nyelvben készült alkalmazások számára. A .NET rendszeren keresztül jön létre a kapcsolat az operációs rendszer és a programozó által megírt alkalmazás között. A Common Language Runtime (CLR) futtatómodul szolgál arra, hogy a mi általunk megírt .NET alkalmazást futtatja a meglevő programozói nyelven. A keretrendszer egy alapvető változást hozott a Microsoft stratégiájában: az alkalmazás fejlesztést kliens-centrikusból szerver-centrikussá tette. Az osztályok kezelése a legfontosabb a keretrendszeren belül, mely nagy mennyiségű előkészített kódja révén állnak a programozók, fejlesztők elé. A megadott osztályok hierarchikusan szervezve vannak, egy megadott logika alapján.

Ezek alapján arra következtethetünk, hogy a ma már használt .NET a szoftverfejlesztés majdnem minden ágát lefedi, legyen szó akár adatbázisokról, akár játékfejlesztésről, vagy kliensmegoldásokról.

A .NET keretrendszer az első elképzelésekor és kiadásakor szabványosított szoftverfejlesztési keretrendszert biztosított a Microsoft Windows fejlesztés megkönnyítésére. Ma a .NET termék nyílt forráskódú fejlesztői keretrendszert biztosít több csatornához. A .NET Foundation által kifejlesztett és MIT licenc alatt kiadott .NET Framework célja, hogy egyszerűbbé, gyorsabbá és következetesebbé tegye a fejlesztést.

A .NET-hez hatalmas ökoszisztéma kapcsolódik, többek között:

- WPF (Windows Presentation Foundation). A Windows operációs rendszer és az asztali alkalmazások tervezését segítő felhasználói felület eszköz.
- Windows Forms. A .NET Framework GUI könyvtára, amely Windows asztali alkalmazásokhoz használható.
- ASP.NET Forms. Egy webes alkalmazás keretrendszer, amely segít a biztonságos és elérhető webes alkalmazások tervezésében.

C#:

A C# a .NET keretrendszerén belül kifejlesztett programozási nyelv. Web-alapú alkalmazások fejlesztésére alkalmas. Robusztus, megbízható, tartós alkalmazások, programok írására volt tervezve. Modernizálja és leegyszerűsíti a C++-t, interoperációs kompatibilitás más .NET nyelvekkel, valamint egy objektumorientált programozási nyelvként szolgál. Tiszta szintaxisokat tartalmaz. A C# natív, nem menedzselt formában nem működik, tehát erősen függ a .NET-től. Struktúrákat, tömböket, enumerációkat, ciklusokat, elágazásokat tartalmazhat. Jól használható Windows-környezetekben, és az üzleti növekedés során előnyös a skálázhatóság szempontjából.

- Gyors fejlesztési idő. A C számos olyan funkcióval rendelkezik, amelyek lehetővé teszik a fejlesztők számára, hogy más nyelvekhez képest gyorsabban kódoljanak.
- Nagyfokú skálázhatóság. A C statikus kódolási jellege minden programját megbízható terméké teszi, amely könnyen finomítható és módosítható. ...
- Objektumorientált.
- Gyengéd tanulási görbe.
- Nagy közösség.

CSS:

A lépcsőzetes stíluslapok, vagyis a CSS egy stílusleíró nyelv, amely arra a procedúrára vonatkozik, ami meghatározza, hogy melyik stílus legyen használatban az adott szekcióban, ahol nem csak egy a meghatározó. Arra vonatkozik, hogy hogy szeretnénk egy adott oldal külalakját megjeleníteni. Betűtípust, színt, hátteret, szegélyt, bekezdéseket stb. módosíthatunk segítségével. HTML vagy XML dokumentumokban írt információkat módosíthatunk vele. Leegyszerűsíti a weblap szerkesztését azáltal, hogy egyetlen kód-rész módosításával egy egész felület kinézetét megváltoztathatjuk. Szerzői-, felhasználói- és kliens stílust tartalmazhat, mindezek külön manuális bevitele nélkül. Leegyszerűsíti a weblap komplexitását, segít a betöltési idő lecsökkentésében, időt spórolva úgy a programozóknak, fejlesztőknek, mint a felhasználóknak is. CSS tulajdonságai dinamikusan változtathatóak JavaScript segítségével.

RDLC report manager:

Az RDL (Report Definition Language) jelentések általában HOSTED jelentések. Ez azt jelenti, hogy az SSRS szervert kell implementálni. Ezek a Visual Studio beépített bővítményei az SQL Server jelentési nyelvhez. Az SSRS telepítésekor rendelkeznie kell a "Business Intelligence Development Studio" nevű kiegészítővel, amely sokkal könnyebben dolgozik a jelentésekkel, mint nélküle. [Ren15]

Az RDLC-fájlok azonban nem tartalmazznak bizonyos információkat, amelyek a ReportViewer vezérlő tervezési idejében az adatkötési kód automatikus generálásától függenek. Az adatok kézi kötésével az RDLC-fájlok a ReportViewer vezérlőben használhatók. A ReportViewer vezérlő nem tartalmaz semmilyen logikát az adatbázisokhoz való kapcsolódáshoz.

Az RDLC jelentések Client contained jelentések, amelyeket a program sehol nem tárol. A névben lévő plusz "c" azt jelenti, hogy "client". Általában ez az RDL nyelv kiterjesztése, amelyet kizárólag a Visual Studio kliensalkalmazásokban való használatra szántak. A Visual Studióban akkor létezik, amikor egy 'reporting' elem jön létre. A megjelenést testreszabhatja a közvetlenül a kód mögött található kiegészítések.

A paramétereket a felhasználó kell kezelje, és bár implementálható wrapper metódusokkal, számos beállítást igényel. Hátránya hogy a felhasználó csak akkor láthatja a paramétereket a "ReportViewer" vezérlőben, ha az remote üzemmódban van, és egy RDL-jelentéshez fér hozzá. Így a vezérlőn kívül kell létrehoznia saját szövegdobozokat, legördülő gombokat, rádiógombokat, hogy átadhassa azokat a vezérlőnek.

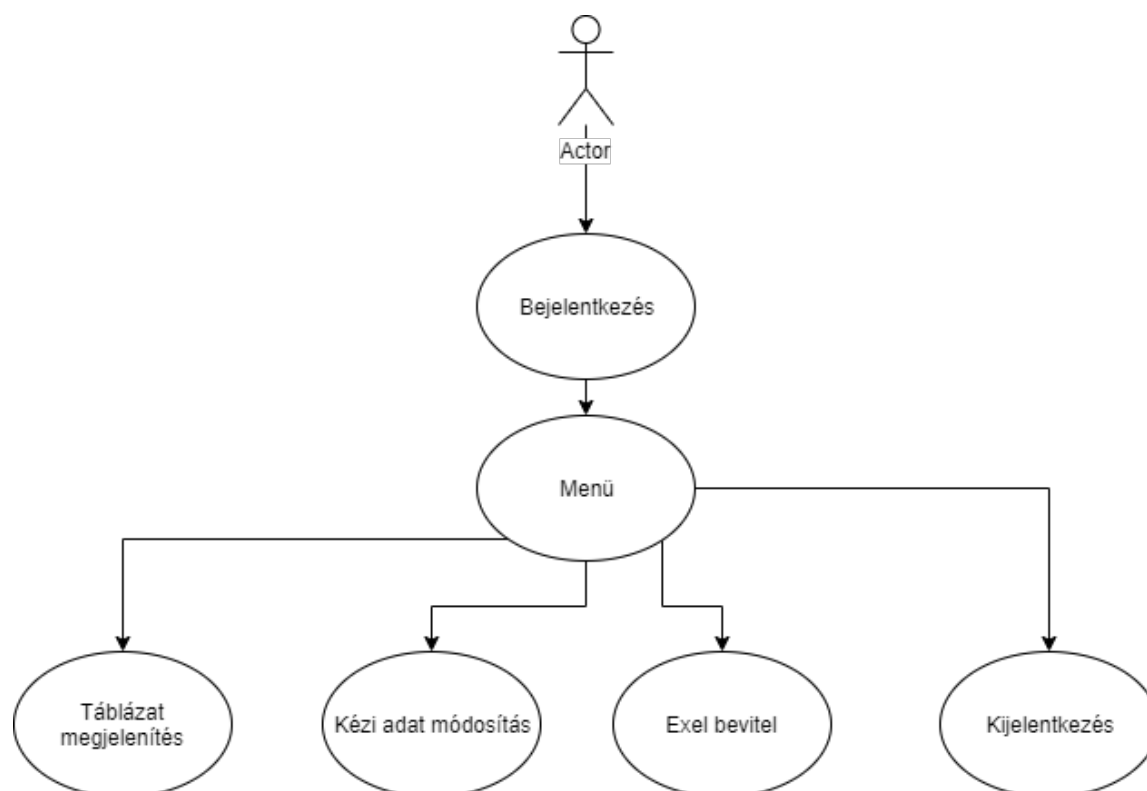
2.1.2. SQL Server

Az SQL (Structured Query Language) egy programozási nyelv. Úgy a programozóknak, mint az elemzőknek, tervezőknek, adatbázis-adminisztrátoroknak ismerniük kell, hiszen ma már minden relációs adatbázis-kezelő használja, támogatja (MySQL, MS Access, Oracle, Sybase, Informix stb). Az SQL szerver az adatbázis szerverén tárolódik, és kliens-szerver hierarchiában használják. Ez azt jelenti, hogy a kliens parancsait a szerver elvégzi, és az adatok így visszajutnak a klienshez. Az SQL nyelv részei: adatdefiníciós nyelv, adatmanipulációs nyelv, lekérdező nyelv és adatvezérlő nyelv. Az adatdefiníciós nyelv (DDL) szolgál az adatbázison belüli részek létrehozására és megsemmisítésére, valamint az adatmanipulációs nyelv (DML) az adatbázisban tartózkodó adatok beillesztésére, módosítására, lekérésére. Az adatbázisunkban az adatok táblázat formájában jelennek meg. Ha módosítunk egy adatot akkor meg kell győződjünk arról, hogy a táblázataink tartalmi és formai megkötéseit betartjuk. Az adatok lekérésekor azonban sokkal nagyobb szabadsággal rendelkezünk, hiszen elvonatkoztatathatunk a táblázataink eredeti adatformátumaitól.

3. fejezet

A rendszer specifikációja

A szoftver elindítása után a felhasználó egy belépési ablakot lát, ahova szükséges megadni létező felhasználónevet, és helyes jelszót. Ezután betöltésre kerül a főmenü, ahonnan további 4 menücsoporthat választható, a létrehozott felhasználók listája, Táblázat szerkesztő, egy kimutatásokat tartalmazó lenyíló fül, egy profil oldal, illetve a képernyő jobb sarkán a kijelentkező gomb. A táblázat szerkesztő oldalon a felhasználó képes módosítani, törölni elemeket. A kimutatásokat tartalmazó oldalakon a felhasználó megtudja tekinteni azokat, lapozásra és letöltésre van lehetőség. A kijelentkező gombbal a felhasználó elhagyja az oldalt, és visszatér a bejelentkező oldalra.



3.1. ábra. Rendszer bemutató

3.1. Felhasználói követelmények

A szoftverbe való belépést egy beléptetési rendszer használata előzi meg, melynek szerepe az illetéktelen személyek távol tartása. (lásd 3.1 ábra első ablaka) Ezen oldalon nem lehetséges a regisztráció, csak előre engedélyezett és létrehozott felhasználónévvel és jelszóval lehet belépni. A felhasználó beírhatja a könnyedén észrevehető ablakokba a saját adatait, mely, ha téves jelezve van a megfelelő helyen, illetve ha törölni szeretné, arra megfelelő gomb biztosít megoldást. A sikeres belépés után a felhasználó a fő menü oldalra van átirányítva (lásd 3.1 ábra második ablaka), ahol lehetősége van további ablakok megnyitására az oldal tetején elhelyezett navigációs sávban. Megtekintheti a felhasználókat, akik illetékesek belépni, a beiktatott tanárok listáját, és azok adatait, lehetősége van azt kézzel egyenkénti módosításokat végrehajtani. Harmadikként egy leugró almenü nyílik meg, ahol három táblázat megjelenítése lehetséges, amin belül lehetőség van a táblázat letöltésére PDF, Word és Excel formátumban. A menü jobb oldali részén egy gomb található, ahol a felhasználó kijelentkezhet az oldalról.

3.2. Rendszerkövetelmények

A rendszer hibátlanul fel kell ismerje a helyes bejelentkezési felhasználónevet, és annak megfelelő jelszavát, ellenkező esetben jelzi, hogy nem helyes. Belépés után a főmenü torzítás nélkül minden képernyő méretre be kell töltsön, azon belül minden gomb, és funkció hibamentesen működik. [Sha]

A táblázatok betöltése után lehetőség kell legyen letölteni, vagy adott esetben módosítani azt. Biztonsági előírás miatt, a rendszer kilépés után engedélyezheti az az utáni visszalépést, csak beléptetési adatok beírásával.

a) Funkcionális követelmények, mire képes a rendszer.

A szoftver fő funkcionalitása az adatok megjelenítése, ezt véve alapul a felhasználó láthatja, módosíthatja, illetve letöltheti a kívánt táblázatot.

b) Nem funkcionális követelmények, megszorítások.

A rendszer megfelelő futása érdekében létre kell hozni egy adatbázist. Amennyiben minden beállítás megtörtént elengedhetetlen az adatbázissal való kapcsolat létrehozása. (kapcsolati sztringek konfigurálása)

3.3. Szoftverek választása

A rendszer kialakításához választott rendszerek fő szempontja a kompatibilitás volt. Minél egyszerűbb kapcsolat teremtés az adatbázissal.

Visual Studio előnyei:

- legtöbbet nyújtó integrált fejlesztői környezet a C# projekteknek
- könnyű kapcsolat teremtés más Microsoft termékekkel
- pluginok sokasága, akár harmadik fél részéről
- egyszerű navigálás
- áttekinthetőség
- hibamentes
- gyors

Visual Studio hátrányai:

- nagyon sok tárhelyet foglal el
- ha nem tanulás céljából használjuk, akkor fizetni kell érte
- nehézkes projekt importálás, ami más környezetben íródott
- Microsoft termék

SQL server előnyei:

- biztonságos adat tárolás
- időt megtakarító karbantartást igényel
- egyszerű telepítés
- elterjedt a használata, ezáltal sok használati útmutató létezik
- .NET-el való jó kapcsolat

SQL server hátrányai:

- Költséges
- Bonyolult interfész

RDLC report előnyei:

- Az ügyfeleknek nem lesz szükségük jelentési kiszolgálóra
- Az ügyfeleknek nem kell frissíteniük meglévő rendszereiket
- Nem kell az összes jelentést újra elkészíteni
- Futtatható jelentések anonim hozzáféréssel

RDLC report hátrányai:

- A jelentéskomponensek nem változatosak
- A végfelhasználó nem tudja testre szabni a jelentés formátumát
- A részletes adatok címe nem jeleníthető meg minden oldalon

Github előnyei:

- Könnyű beillesztés harmadik féltől származó alkalmazásokba
- Verziókezelő rendszer és egyszerű együttműködés egy projekten
- Nagyon gyors keresés a projekten belül
- Nagy közösség és könnyen megtalálható segítség

Github hátrányai:

- Árazás
- Biztonság
- Egy rossz összefésülést nagyon nehéz visszaállítani

4. fejezet

Tervezés

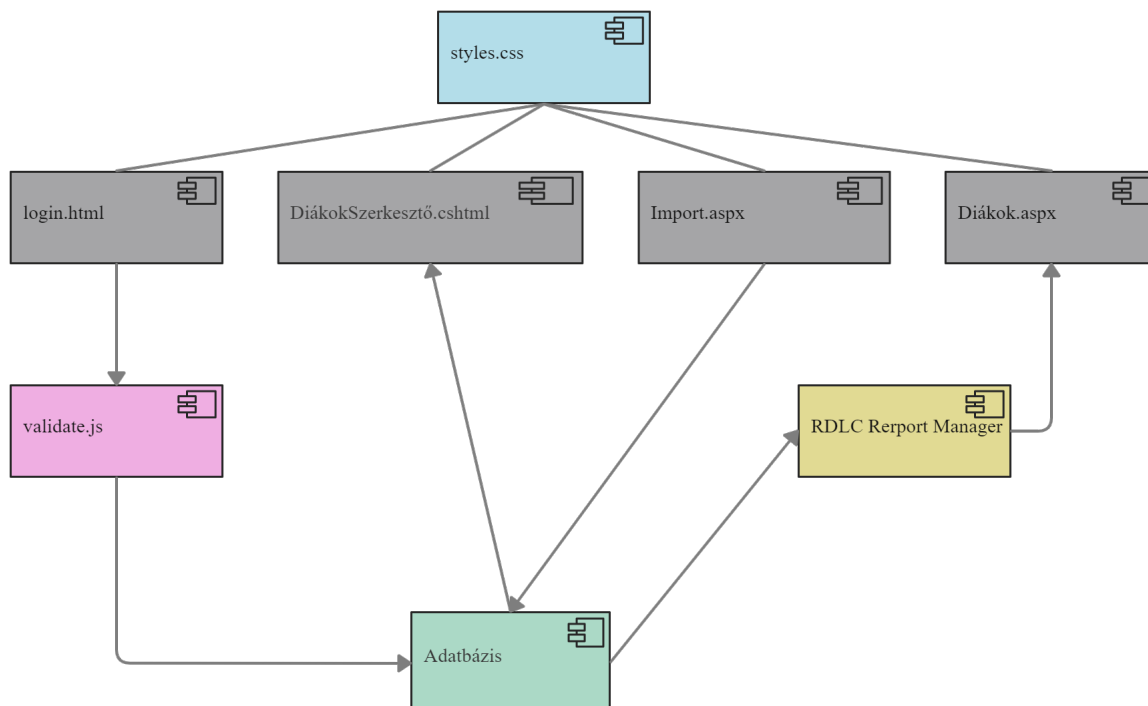
A tervezés kezdeti szakaszában egy követelménydokumentumot szerettem volna készíteni, együtt a klienssel, hogy a cél mindenki számára egyértelmű és azonos legyen. Ennek eredményeképpen tisztáztuk a kérdéseket, hogy mit szeretne, mit lehet implementálni, és melyik módszer lenne a legmegfelelőbb a cél elérésének érdekében. A legjobb koncepció kompatibilitása, bővíthetősége és használhatósága érdekében több lehetséges utat is kipróbáltam.

Egyértelmű döntés volt, hogy a szoftver C# nyelvezetben legyen, .NET keretrendszerben, mivel azt szerettük volna, hogy a program a lehető legkevesebb beavatkozással hosszú időre tartós maradjon.

A cél egyértelművé vált azonnal. Egy web alkalmazás létrehozása, ami bárki számára másodpercek alatt megérthető, lényegre törő, fölösleges funkciók nélkül. A letisztult, mai normáknak megfelelő stílus használata mellett fontos volt a program alapköveit képező funkcionalitások. Adatok tárolása, bevitele, és azok dinamikusan szerkeszthető kimutatókba való ágyazása.

Egy ingyenes használatú adatbázis kezelő rendszert használva szerettem volna szervezni az adatokat, ami már a tervezés kezdeti szakaszában rossz megközelítésnek bizonyult, mivel implementálása, és használata is nehéz volt, ugyanakkor sem idő, sem pénz szempontjából nem volt kifizetődő. Így jutottam el az SQL Server használatához, amely Microsoft termékként egyszerű összeköttetést biztosított a .NET keretrendszerrel.

A modern elgondolás érdekében a CSS nyelv használatát vettem igénybe, ezáltal az oldal megjelenése könnyen változtatható köszönhetően a dinamikus szerkeszthetőségnek. Az SQL Server után olvasása közben sok pozitív szempontjáról szereztem tudomást. Elsősorban az SQL Server széleskörű használati táborra igazolta számomra, hogy ez a legbiztonságosabb jelenleg a piacon, ami köszönhető a mögötte álló nagy fejlesztő csapatnak. Ezért úgy gondoltam, hogy a bejelentkezés folyamatában szükséges adatok biztonságát erre bízom, és nem igényel további titkosítást.



4.1. ábra. Komponens diagram

A rendszer tervezése során kiépítettem egy egyszerű vázlatot, **(lásd 4.1 ábra)**, ahol a megjelenítendő oldalak kapcsolatban vannak egymással. A bejelentkező oldalhoz szükséges kulcspárok az adatbázisban vannak tárolva, az annak megfelelő titkosítással. Minden táblázat megjelenítésének szüksége van az adatbázissal való kapcsolatra, kivételt tesz az Import oldal, mely adatok igénylése helyett, adatok feltöltésére szolgál, ezért szükséges egy olyan folyamat kialakítása, amely az adatokat rendszerezi az előre elkészített adatbázis táblázataiban. Az RDLC report rendszernek szüksége van hasonló adatok legkérdezésére, és egy script segítségével lesz megjelenítve az oldalon. Minden oldal stílus kialakításához szükséges design elemek egy .css formátumú fájlban vannak elmentve, ezáltal kiküszöbölve az oldalak közti stílus különbségek lehetőségét.

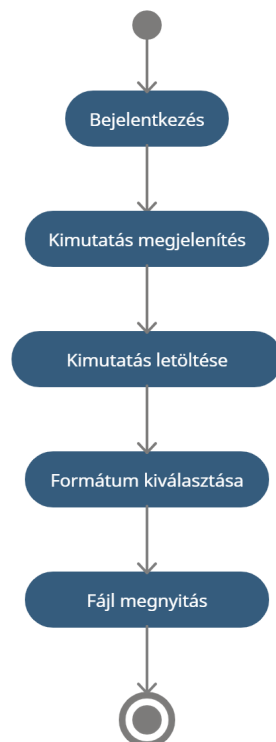
4.1. Kimutatás tervezése:

Következő tervezési szempontként a modularitást vettem figyelembe.

Első próbálkozásként egy sajátos megközelítést próbáltam ki, amelyben magam szerkesztettem meg a táblázatokat, és írtam meg a letöltést biztosító funkciókat, ami jól működött, viszont megfeledeztem egy fontos szemponttól, amit a klienst említett, a bővíthetőséget. A beszélgetések során kiderült, hogy az adatok önmagukban továbbra is azonos típusúak maradnak, viszont a táblák szerkezete változhat, ezért szükség volt egy könnyen szerkeszthető szerkezetre.

Tanulva a kompatibilitási nehézségeket okozó adatbázis kezelő rendszer kipróbálása után, arra a következtetésre jutottam, hogy megkellene próbálni ugyancsak a [2ma] Microsoft által fejlesztett RDLC report manager használatát, ami ugyanakkor kockázatos volt, mivel én még nem dolgoztam hasonló technológiával, és megkellett tanulnom annak felépítését.

Az RDLC report előnyös megoldást biztosít a táblázatokkal való munkában, mivel többek között beépített funkciókat tartalmaz, a kimutatások lapozására, táblázatban való adatok keresésére, de nem utolsó sorban a teljes kimutatás letöltésére. A felhasználó az RDLC által biztosított funkcióval letöltheti a megjelenített kimutatást PDF-ben, Excel táblázatban, és Word dokumentumként. A kiválasztott formátumra kattintva, a rendszer automatikusan letölti a kimutatást a böngésző által meghatározott szabványos mappába. Ez a funkció hasznosnak bizonyult, mivel nem igényelt további fejlesztést. (lásd 4.2 ábra)



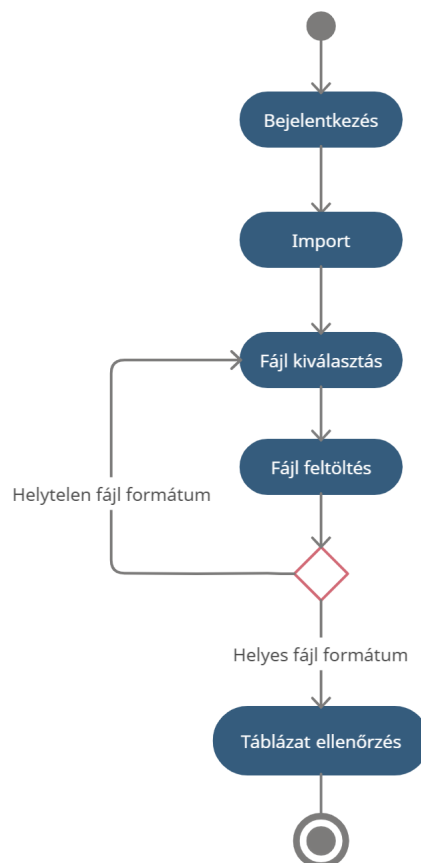
4.2. ábra. Megjelenített táblázat letöltés

4.2. Adatbevitel tervezése:

Az adatfeltöltés Excel táblázatból szükséges, mivel hosszas folyamat soronként feltölteni a táblázatokat. A cél egy olyan funkció készítése, amely nem csak feltölti a táblázatot, hanem ellenőrzi is az adatokat, elkerülendő a duplikálás, a rossz név kitöltés. Tévedni emberi, ezért bárkivel megesik, hogy egy másodpercnyi figyelmetlenség egy elíráshoz vezet. Ezért a funkcionalitásnak szüksége van egy köztes tábla létrehozásához, annak érdekében, hogy az adatokat megszűrje, és csak a lényeges, megerősített adatok kerüljenek fel a végső táblázatba, amit a felhasználó is lát.

SQL utasítással szeretném elérni a programban egy ideiglenes táblázat létrehozását, ami csak addig létezik, amíg összehasonlításra kerül a cél-táblázat adatai, az ideiglenes táblázat adataival.

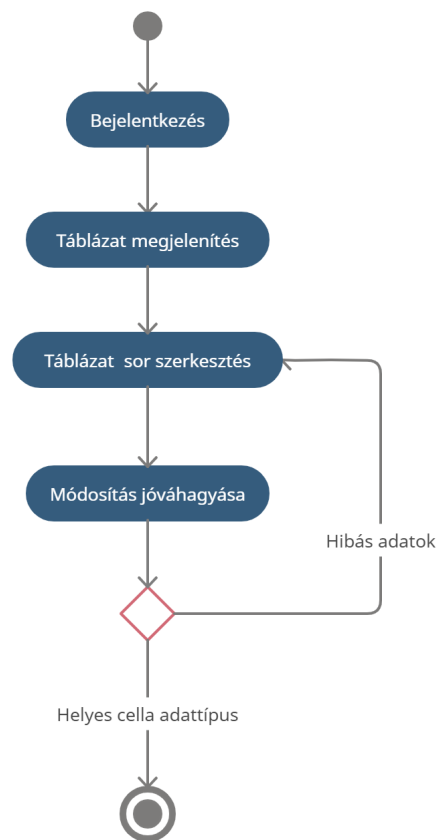
A rendszernek fájl feltöltése után szükséges ellenőriznie, hogy a feltöltendő fájl formátuma eltér-e az Excelben lévőétől. Ha a fájl nem felel meg a feltételnek, vagy fájl kiválasztása nélkül szeretnék a bevitel gombra kattintva fájlt feltölteni, akkor a felhasználó kell kapjon egy hiba üzenetet a megfelelő utasítással, hogy mit rontott el, másképp helyes formátumú fájl feltöltése után a rendszer küld egy hitelesítési üzenetet, és a felhasználó ellenőrizheti, hogy sikeresen végbe ment-e a folyamat. (lásd 4.3 ábra)



4.3. ábra. Táblázat módosítás excel táblázatból

Kívánt funkció volt, hogy adott változtatást a végrehajtó jóvá kell hagyjon, mielőtt véglegesen megváltozik. Ezt a szempontot figyelembe véve, a LINQ szintaxis tűnt a megfelelőnek, viszont nem volt stabil a használata, ami kockázatot jelentett a program működésében, így arra a következtetésre jutottam, hogy jobb a kevesebbet nyújtó, de biztonságosabb megoldás.

A weboldalról való táblázat szerkesztése során, lehetőség van az adott sor minden egyes cellájának módosítására, de figyelembe kell venni a cella formázásának beállításait, illetve nem lehetséges az azonosító módosítása egy már létező azonosítóra. Hiba esetén a rendszernek egy hiba üzenetet kell kiírnia a táblázat alján a módosítást elutasító indokkal. A módosítás mellett lehetőség kell legyen minden oszlop törlésére soronként. (lásd 4.4 ábra)

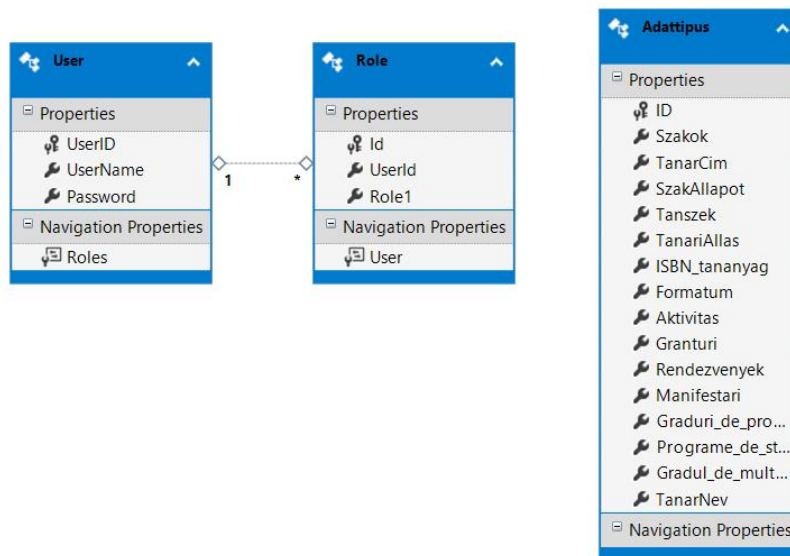


4.4. ábra. Táblázat módosítása weboldalról

4.3. Adatbázis tervezése:

Az egyszerűsége törekedve, szerettem volna minél egyszerűbb adatbázist létrehozni. E célból csak bejelentkezési felhasználónevet és jelszót tárolom. Ami fontos a személyi jogok tiszteletben tartása miatt is, ezért nem volt cél a felhasználó személyes adatait tárolni.

Az adatok gyors és dinamikus kezelése érdekében egy nagyobb táblázat létrehozása mellett döntöttem, ami utólagosan a felhasználói felületből szerkeszthető. Ebből a célból úgy gondoltam, hogy egy táblázatra lesz szükségem az adatok tárolására, de ez megkell, hogy feleljen minden adat típusnak.



4.5. ábra. Adatbázis tervezése

5. fejezet

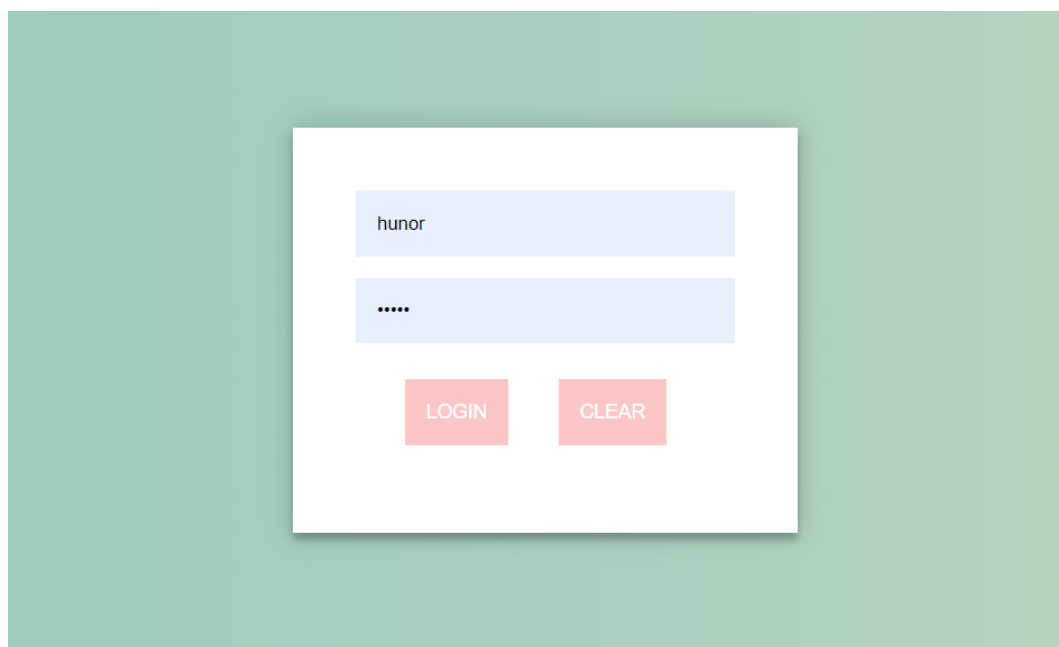
Kivitelezés

5.1. Bejelentkező oldal

A szoftver elindítása után az első oldal amit a felhasználó lát az a bejelentkező, ahol elengedhetetlen a felhasználónév és jelszó kitöltése ahhoz. Elrontott próbálkozás esetén van egy Clear gomb, mely kitörli a már beírt felhasználónevet és jelszót. A belépést való próbálkozást a Login gombbal lehet, mely helyes adatok esetén tovább visz a főmenüt.

Rossz próbálkozás esetén a felhasználó egy hibaüzenetet kap, úgyszintén, ha valamely mező üresen maradt.

A biztonságot, illetve felhasználónév hitelesítést Authorize módszerrel [\[htt\]](#) valósítottam meg, Session-t létrehozva. A felhasználónév és jelszó objektumokként vannak tárolva, mely modell az adatbázis egy táblájával van összekötve.

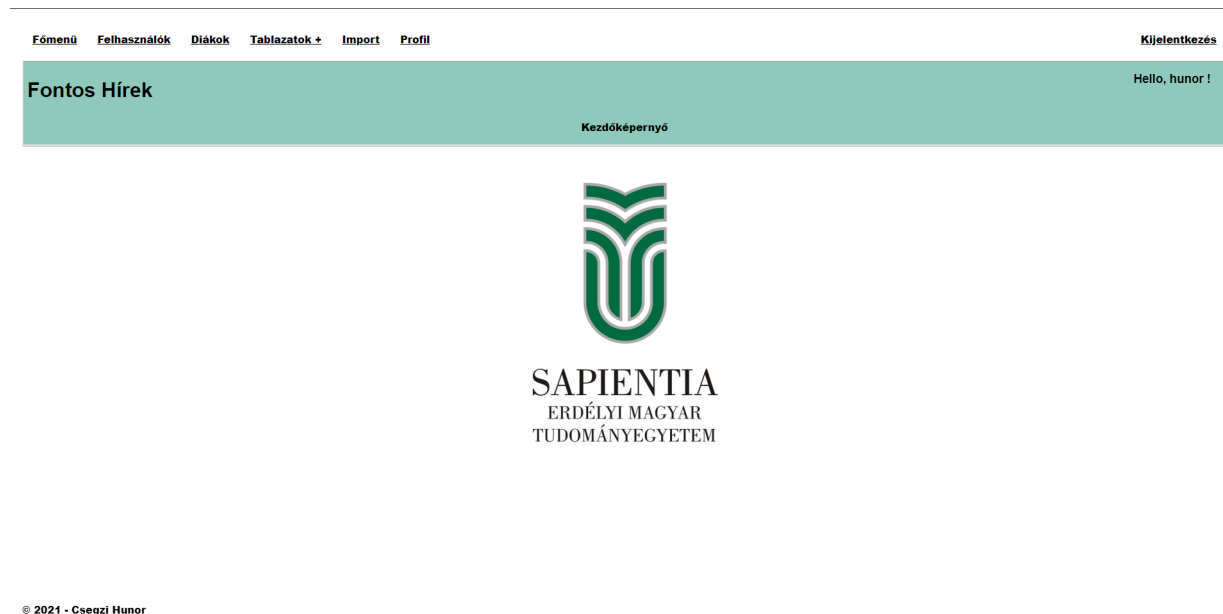


5.1. ábra. Bejelentkező oldal

5.2. Főoldal

A főoldal egy letisztult, egyszerű tervezésű, innen elérhető bármelyik funkció a navigációs sávból. Bal oldalra kerültek elhelyezésre az oldal további eszközeivel kapcsolatos menük, és a jobb oldalra a Kijelentkezés gomb, mely megszakítja a felhasználó további böngészésének lehetőségét, és visszatéríti a bejelentkező oldalra. [CSS]

A tervezés során figyelembe vettem a szemnek kedvező színválasztást, amely kedvező feltételt biztosít a munka elvégzéséhez, elkerülve a fárasztó, és túl zsúfolt menü gombok elhelyezését. Fontos tényező volt a felhasználóbarát menü, azért hogy első ránézésre a felhasználó bármilyen fennakadás nélkül tudjon eligazodni.



5.2. ábra. Fő oldal

5.3. Diákok oldal

A Diák oldal betöltése után a felhasználó egy táblázatot lát, mely hét különböző, a diák fontosabb információit tartalmazó mezőből áll, egy azonosító mező, amely egyedi, továbbá a neve, a szak amelyre jelentkezett, a szak koordinátora neve, a státusza, illetve a legfontosabb az elégedettségi szint mutató.

Ezen belül minden sorban módosítás végezhető el minden oszlop mezőjére külön-külön, amelyet menteni kell a táblázat bal oldalán megjelenített mentést mutató ikonnal. Ugyanakkor a sorok törölhetőek, a 'Törlés' ikonnal, aminek következtében a felhasználó értesítést kap, hogy az adott sor törlése került.

A táblázatot GridView formátumban írtam, a könnyű és viszonylag egyszerű szerkezet kialakítása miatt, melynek hátránya, hogy nem módosítható dinamikusan. A táblázat utolsó sora mindig üres marad, mivel itt lehetőség van új adat bevitelére.[\[bM\]](#)

Ez a táblázat elérhető a Táblázatok+ gombra kattintva, és a Diákok jelentést választva.

Főmenü	Felhasználók	Diákok	Táblázatok +	Import	Profil	Kijelentkezés
Diákok táblázata						
ID	Képzés	Szak vezető tanár neve	Diák neve	Jelentkezés éve	Aktiválás	Elégedettségi tényező
111	Informatika	Bács Brigitta	Pál Gergely	2016	Abszolvált	5
112	Informatika	Bakos Georgina	Király Zsanett	2017	Abszolvált	5
113	Informatika	Pap Nikoletta	Bakos Krisztián	2017	Abszolvált	5
114	Informatika	Pap Nikoletta	Horváth Zsombor	2018	Aktiv	4
115	Informatika	Pap Nikoletta	Sipos Flórián	2016	Abszolvált	5
116	Informatika	Pap Nikoletta	Somogyi Dorka	2016	Abszolvált	4
120	Informatika	Pap Nikoletta	Nemes Emese	2016	Abszolvált	3
132	Informatika	Pap Nikoletta	Zobor Endre	2016	Abszolvált	2
222	Informatika	Bács Brigitta	Kiss Péter	2019	Felfüg.	1
333	Informatika	Bakos Georgina	Major Ákos	2020	Aktiv	5
444	Informatika	Bács Brigitta	Major Veronika	2020	Aktiv	5
555	Informatika	Bakos Georgina	Gergely Gergely	2020	Aktiv	5
565	Informatika	Bakos Georgina	Ismeretlen alak	2020	Aktiv	5
985	Informatika	Bács Brigitta	Király Flórián	2016	Abszolvált	5
630	Informatika	Bakos Georgina	Apród Valentin	2017	Abszolvált	5
945	Informatika	Pap Nikoletta	Kapocs Árpád	2017	Abszolvált	5
224	Informatika	Pap Nikoletta	Nemes Tibor	2018	Aktiv	4
518	Informatika	Pap Nikoletta	Balázs Jakab	2016	Abszolvált	5
781	Informatika	Pap Nikoletta	Kozma Gyula	2016	Abszolvált	4
194	Informatika	Pap Nikoletta	Pap Zsolt	2016	Abszolvált	3
307	Informatika	Pap Nikoletta	Bognár Balázs	2016	Abszolvált	2
550	Informatika	Bács Brigitta	Vass Roland	2019	Felfüg.	1
323	Informatika	Bakos Georgina	Mezei László	2020	Aktiv	5
801	Informatika	Bács Brigitta	Hegedűs Bettina	2020	Aktiv	5
552	Informatika	Bakos Georgina	Sándor Gabriella	2020	Aktiv	5

5.3. ábra. Diákok oldal

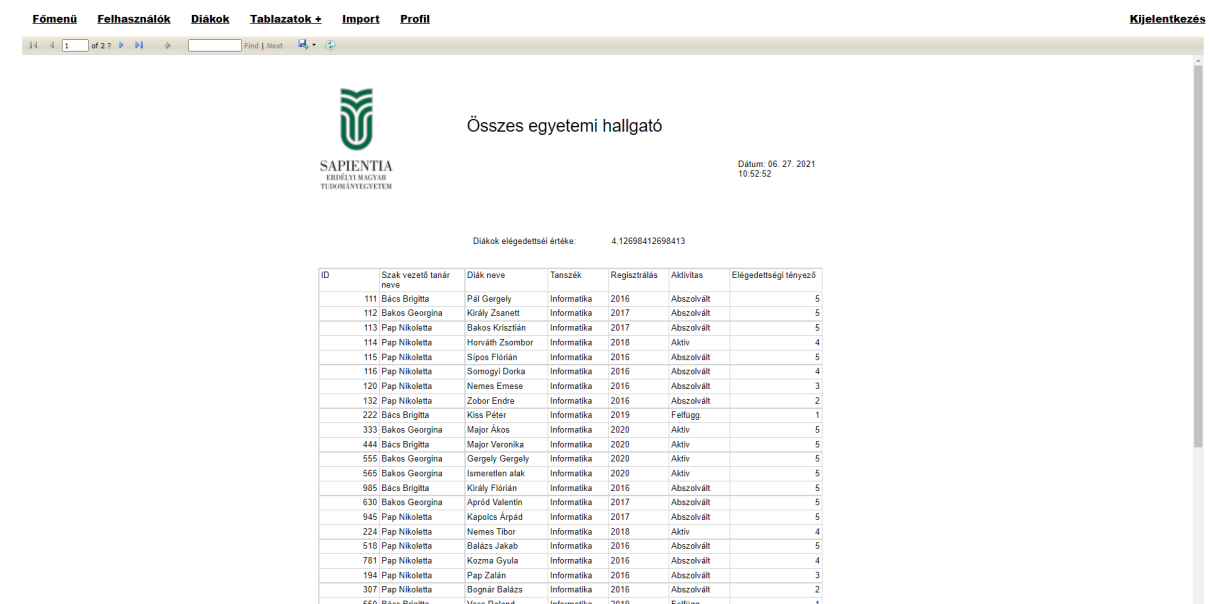
5.4. Diák jelentés oldal

Ezen az oldalon a felhasználó egy RDLC report formában látja a táblázatot, aminek segítségével, a táblázat letölthető három formátumban, javasolt Excel formátumban a torzítás miatt, mivel a táblázat széles.

Az RDLC formátum XML sémával rendelkezik, ami egyszerűsíti a táblázat szerkeszthetőségét, és lehetőséget nyújt a táblázat mellett a kimutatás testre szabásához. A funkció megoldást nyújthat akár több kimutatás elhelyezésére egy oldalon, mivel annak kerete dinamikusan testreszabható.

Az RDLC report használatával az adatok feldolgozása érdekében függvényeket lehet használni oszlopok átlag, összeg számítás, vagy oszlopok csoportosításához. [And16]

A 'Táblázatok+' menüben található még két kimutatás, amelyek neve 'Szak', és 'Egyetem', mindkettő hasonlóképpen letölthető, és lapozható, mint a már fent említett táblázat.



ID	Szak vezető tanár neve	Diák neve	Tanszék	Regisztrálás	Aktivitas	Elégedettségi tényező
111	Bács Brigitta	Pál Gergely	Informatika	2016	Abszolált	5
112	Bács Georgina	Király Zsuzsanna	Informatika	2017	Abszolált	5
113	Pap Nikolett	Bakos Krisztián	Informatika	2017	Abszolált	5
114	Pap Nikolett	Hornáth Zsombor	Informatika	2018	Aktiv	4
115	Pap Nikolett	Sipos Flórián	Informatika	2016	Abszolált	5
116	Pap Nikolett	Somogyi Dorka	Informatika	2016	Abszolált	4
120	Pap Nikolett	Nemes Emese	Informatika	2016	Abszolált	3
132	Pap Nikolett	Zebor Endre	Informatika	2016	Abszolált	2
222	Bács Brigitta	Kiss Péter	Informatika	2019	Felügy	1
333	Bács Georgina	Major Ákos	Informatika	2020	Aktiv	5
444	Bács Brigitta	Major Veronika	Informatika	2020	Aktiv	5
555	Bács Georgina	Gergely Gergely	Informatika	2020	Aktiv	5
565	Bács Georgina	Ismeretlen alak	Informatika	2020	Aktiv	5
985	Bács Brigitta	Király Flórián	Informatika	2016	Abszolált	5
630	Bács Georgina	Apród Valentin	Informatika	2017	Abszolált	5
945	Pap Nikolett	Kapocs Árpád	Informatika	2017	Abszolált	5
224	Pap Nikolett	Nemes Tibor	Informatika	2018	Aktiv	4
510	Pap Nikolett	Balázs Jakab	Informatika	2016	Abszolált	5
781	Pap Nikolett	Kozma Gyula	Informatika	2016	Abszolált	4
194	Pap Nikolett	Pap Zsolt	Informatika	2016	Abszolált	3
307	Pap Nikolett	Bognár Balázs	Informatika	2016	Abszolált	2
550	Bács Brigitta	Vass Roland	Informatika	2019	Felügy	1

5.4. ábra. Diák report készítés

5.4.1. RDLC Report manager implementálása

Az RDLC eléréshez használni kell:

- Microsoft.Reporting.WebForms;

Ezt a könyvtárat használva elérhetjük a ReportViewer eszközt, melyet elhelyezhetünk az oldalon tetszés szerint. Az RDLC reportok megfelelő használata érdekében az adatbázisból DataSet-et szükségesek, melyek tartalmazzák a report adat típus lehetőségeit. Ezután elkészíthetjük az XML kiterjesztésű fájlt, ahol egy táblázatot létrehozva, egyenként berakhatjuk a megjeleníteni kívánt oszlopokat. Ezekhez lehet utólag hozzáadni, vagy kitörölni. Minden oszlop tulajdonságai konfigurálható, és az adatok csoportosíthatóak. A felhasználó bármilyen sorrendben láthatja az a megjelenített oszlopokat. [[Gan](#)]



5.5. ábra. Diák jelentés oldal

5.5. Diák Excelből való bevitel oldal

A már említett 'Diákok' táblázat manuális adat feltöltése mellett, lehetőség van Excel táblázatból való adatok bevitelére is. Ehhez az 'Import' ablakot kell megnyitni, ahol a bal felső sarokban kikell választani a számítógépről kívánt Excel dokumentum elérését, és utólag a feltöltés gombra kattintani.

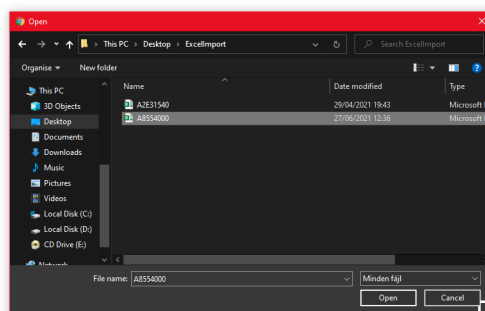
Fontos, hogy a kívánt táblázat tartalmazza ugyanazon oszlop neveket, mint a frissíteni kívánt adatbázis oszlop nevei.

Főmenü Felhasználók Diákok **Tablázatok +** Import Profil

Kijelentkezés

Excel Import

Fájl kiválasztása Nincs fájl kiválasztva Import



© 2021 - Csegezi Hunor

5.6. ábra. Diák import oldal

A funkciót fontosnak véltem, mivel az idő során hatalmas mennyiségű adat halmozódhat fel, melyek kézi bevitele túl időigényesnek bizonyulna. Ennek elérése érdekében ugyancsak a Microsoft által fejlesztett funkciók segítségével egy parancssort írtam.

A funkció az Excel köré épült, hiszen ez egy hatékony eszköz, amely világszerte beépült az üzleti folyamatokba – legyen szó részvények vagy kibocsátók elemzéséről, költségvetés-tervezésről vagy adatok listázásáról, és rendezéséről. Táblázatkezelő programként az Excel nagy mennyiségű adatot képes tárolni egy vagy több munkalapot tartalmazó munkafüzetekben. Azonban ahelyett, hogy adatbázis-kezelő rendszerként szolgálna, mint például az Access, az Excel adatelemzésre és -számításra van optimalizálva. Mivel az Excel sokoldalú felhasználást biztosít, ezért fontos azok minden fájlformátumban való beolvasása is, úgy XLS-fájlban melyek bináris adatfolyamként összetett fájl formájában vannak elrendezve, az [MS-XLS]-ben leírtak szerint, vagy XLSX-fájlban, Office Open XML-formátumon alapulva, amely tömörített XML-fájlokban tárolja az adatokat ZIP formátumban.

5.5.1. Szükséges könyvtárak

Ehhez a megoldáshoz használnom kellett a C(Sharp) által támogatott SQL parancsokat engedélyező könyvtárakat. Az adatbázissal való kapcsolatot az SqlConnection objektummal jöhetett létre, ezen keresztül volt használható a csatlakozási sztring. [Dro]

Ezen belül OleDb (Object Linking and Embedding Database) egy kapcsolati metódus, segítségével létrehozható egy híd az applikáció és a kívánt adatforrás között

Az SQL parancsok végrehajtásának eléréséhez szükség volt:

- System.Data;
- System.Data.OleDb;
- System.Data.SqlClient;

```
public ActionResult Index(HttpPostedFileBase file)
{
    if (file == null || file.ContentLength == 0)
    {
        ViewBag.Error = "Csak Excel formátum elfogadott";
        return View("Index");
    }
    else
    {
        if (file.FileName.EndsWith(".xls") || file.FileName.EndsWith(".xlsx"))
        {
            string filename = Guid.NewGuid() + Path.GetExtension(file.FileName);
            string filepath = "~/excelfolder/" + filename;
            file.SaveAs(Path.Combine(Server.MapPath("~/excelfolder"), filename));
            InsertExceldata(filepath, filename);

            return View("Index");
        }
        else
        {
            ViewBag.Error = "Helytelen fájl formátum<br>";
            return View("Index");
        }
    }
}

1 reference
private void ExcelConn(string filepath)
{
    //kapcsolat teremtés
    string constr = string.Format(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={0};
                                   Extended Properties=""Excel 12.0 Xml;HDR=YES;""", filepath);
    Econ = new OleDbConnection(constr);
}

1 reference
private void InsertExceldata(string filepath, string filename)
{
    string fullpath = Server.MapPath("~/excelfolder/") + filename;
    ExcelConn(fullpath);

    // tobb sheet
    string query = string.Format("Select * from [{0}]", "Sheet1$");
    OleDbCommand Econ = new OleDbCommand(query, Econ);
    Econ.Open();
    DataSet ds = new DataSet();
    OleDbDataAdapter oda = new OleDbDataAdapter(query, Econ);
    Econ.Close();
    oda.Fill(ds);
    con.Open();
    // Idéiglenes táblázat létrehozás
    SqlCommand cmd = new SqlCommand("Create table #MyTempTable(ID int, Szakok nvarchar(50),
```

5.7. ábra. Kapcsolat az adatbázissal

Az **5.7-es ábrában** látható a fájlból való beolvasás menetét, amiben első lépésként ellenőrzöm a beolvasandó fájl formátumát, vagy tartalmának bizonyosságát, hogy nem üres. Ha a fájl helyes, és nem üres, következőkben a fájl mentésére kerül sor egy előre elkészített könyvtárban, aminek célja a fájlhoz való hozzáférés a következőkben.

Az ExcelConn függvény a kapcsolat teremtést szolgálja, amelyet az InsertExceldata függvényben meghívunk. Az OleDb api segítségével a fájlból lekérjük az egész adatot az első lapról. Továbbiakban egy ideiglenes táblázat létrehozását éreztem szükségesnek az adat duplázások elkerülésének érdekében.

5.5.2. Táblázat összehasonlítás

A táblázat feltöltés oly módon valósult meg, hogy elsősorban az adatok egy ideiglenes táblázatba kerülnek mentésre, és egy SQL parancs összehasonlítja az adatbázisban meglévő adatokkal, és a megegyező ID-vel rendelkező sorokat felülírja, míg az új ID-val rendelkező sorokat beszúrja a táblázat elejére. [HaB]

```

DataTable dt = ds.Tables[0];

//Masolas a temp tablebe
SqlBulkCopy bulkCopy = new SqlBulkCopy(con);
bulkCopy.DestinationTableName = "#MyTempTable";
bulkCopy.ColumnMappings.Add("ID", "ID");
bulkCopy.ColumnMappings.Add("Szakok", "Szakok");
bulkCopy.ColumnMappings.Add("TanarCim", "TanarCim");
bulkCopy.ColumnMappings.Add("TanarNev", "TanarNev");
bulkCopy.ColumnMappings.Add("Tanszek", "Tanszek");
bulkCopy.ColumnMappings.Add("TanariAllas", "TanariAllas");
bulkCopy.ColumnMappings.Add("Aktivitas", "Aktivitas");
bulkCopy.ColumnMappings.Add("Gradul_de_multumire", "Gradul_de_multumire");

bulkCopy.WriteToServer(dt);
//Felülírás
SqlCommand cmd2 = new SqlCommand("UPDATE " +
    "Table_A " +
    "SET " +
    "Table_A.Szakok = Table_B.Szakok, " +
    "Table_A.TanarCim = Table_B.TanarCim, " +
    "Table_A.TanarNev = Table_B.TanarNev, " +
    "Table_A.Tanszek = Table_B.Tanszek, " +
    "Table_A.TanariAllas = Table_B.TanariAllas, " +
    "Table_A.Aktivitas = Table_B.Aktivitas, " +
    "Table_A.Gradul_de_multumire = Table_B.Gradul_de_multumire " +
    "FROM " +
    "Adattipus AS Table_A " +
    "INNER JOIN #MyTempTable AS Table_B " +
    "ON Table_A.ID = Table_B.ID", con);

cmd2.ExecuteNonQuery();
SqlCommand cmd3 = new SqlCommand("INSERT Adattipus(ID,Szakok,TanarCim,TanarNev,Tanszek, TanariAllas,Aktivitas,Gradul_de_multumire) " +
    "SELECT * FROM #MyTempTable tt WHERE NOT EXISTS(SELECT 1 FROM Adattipus yt WHERE yt.ID = tt.ID)", con);

//Parancs végrehajtása
cmd3.ExecuteNonQuery();
//Kapcsolat megszakítás
con.Close();
}

```

5.8. ábra. SQL utasítás

Az **5.8 ábrán** látható a már említett ideiglenes táblázat létrehozása, amint bulkCopy funkcióval sorrol sorra másolom át az adatokat. A Sql parancs segítségével egy felülírást végzek el, hogy ellenőrizhessem, hogy a már szerveren található táblázat adatai és a frissen létrehozott ideiglenes táblázat adatai között található-e hasonlat. Az adatok ellenőrzési feltétele egy ID, ami minden sornak egyedi. Amint az ideiglenes táblázat ellenőrző folyamata során a táblázat elmentette a közös elemeket, következő lépésben beszúrom azokat, amik nem találhatók meg benne.

Összefoglaló

Dolgozatomban egy kimutatásokat feldolgozó rendszert készítettem el .NET keretrendszerben, C(sharp) programozási nyelvet használva, és CSS, illetve JavaScript-el kiegészítve. Első sorban a cél kitűzéseem ismertettem, a munkafolyamatok egyszerűsítése, és az időt figyelembe véve egy sokkal hatékonyabb módszer kialakítása. Indokoltam az eszközök választását, amit végsősoron elmondhatok, hogy minden általam kitűzött célt elértem a Microsoft által fejlesztett programokkal, csomagokkal, illetve könyvtárakkal, ezáltal a továbbiakban egy hatékony, biztonságos alapot képezve a jövőbeli továbbfejlesztésekre. Az RDLC Report manager bevált, és használata pozitív volt, mivel egy nagyon érdekes és új technológiát volt alkalmam megismerni, egy dinamikusan változó, de ugyanakkor kézenfekvő beállításokkal. Elégedett vagyok a Microsoft termékek közötti könnyed és egyszerű összekapcsolási lehetőségekkel, amelyek elkészítéséhez számos leíró- útmutató létezik.

Bemutattam a szoftverek választási folyamatát, hiszen számos megoldás létezik már kimutatások létrehozásához, mind nagyon jó a saját módján, mindnek megvannak az előnyei és hátrányai, annak függvényében, hogy a fejlesztő miként szeretne dolgozni. Ez egy pozitív felfedezés volt számomra, mert alkalmam volt olyan szoftverek, megoldások után nézni, tanulmányozni, és kipróbálni azokat, amikről másképp lehetséges, hogy egyáltalán nem szerzek tudomást. Az RDLC egy könnyen beépíthető szolgáltatás, amely tökéletesen használható a legkisebb kimutatástól a legösszetettebbig. Számos funkciója elérhető, az adatok rendezése, csoportosítása, egyedi kifejezések szerkeszthetősége mellett számos adattípustól függő eredmény kimutatható, többet között átlag, összeg.

A dolgozat további részében dokumentáltam a tervezés folyamatát, illetve a szoftver kivitelezését, amelyből kiderült, hogy a mai lehetőségeket számításba véve több úton is ellehet érni ugyanazt a célt, csak nem ugyanaz a folyamat, amin keresztül megyünk a cél eléréséig. Az adatbázis tervezése, az adathalmazok bevitele, és módosítása érdekében törekedtem a számomra legkedvezőbb, de ugyanakkor leghatékonyabb megoldás kiválasztásában, amit úgy gondolok, hogy sikerült elérnem.

Nem utolsó sorban, az így létrejött szakdolgozatommal egy olyan platformot sikerült megvalósítanom, amely a felső oktatásban úgy hasznos, mint nélkülözhetetlen. Szolid rendszerem az oktatásban dolgozók munkáját leegyszerűsíti, időt spórolva meg az adatok bevitelkor, rendezésekor és lekérdezésekor. A felső oktatásban részt vevő diákok száma akár több ezret is elérheti, melyek adatai szükségesek a nyilvántartáshoz, iskoláztatáshoz, és az egyetem befejezéséhez. Az adataik sokaságára van szükségünk ahhoz, hogy egy megfelelő ellenőrzéskor, lekérdezéskor minden egyes lehetőséget megkapjunk, ezeket hierarchikusan eltároljuk, egy olyan adat-rendszert létrehozva, amely egymásra épül, és egymásból van építve.

Ennek a nagy mennyiségű, komplex, halmazott adatnak a kezelése csakis egyszerű megoldással és lekérdezésekkel valósulhat meg, így teljes látáskört biztosítva a felhasználónak. Egy adott témakörben nem minden adatra van szükségünk, ezeknek szelektálása létfontosságú egy jó rendszer létrehozásában, így időt spórolva. Ebben leginkább az RDLC reportok nyújtották a kielégülést. Így elmondhatom, hogy szakdolgozatom úgy egyesíti a komplexet az egyszerűvel, hogy a felhasználó hétköznapi tennivalóját, munkáját nagyban megkönnyíti, áttekinthetővé teszi.

5.6. Tovább fejlesztési lehetőségek

A jövőben elérhető célként tűztem ki magamnak egy új módszer fejlesztését az Excelből való adatok bevitele érdekében, mely a kiértékelés után, lehetőséget biztosít a felhasználó számára, hogy elfogadja vagy elutasítsa az adatok felülírását.

Egy olyan funkcionalitás fejlesztését, amely segítségével a szoftver egy táblázat oszlopait megszüríhetjük, és csak az kerül a képernyőre, amit a felhasználó szeretne látni. Ennek következtében nem lenne szükség egynél több táblázatra, ezen belül az adatok kimutathatóak egy előre megadott intervallum szerint is.

Lehetőséget szeretnék biztosítani a felhasználóknak jelszó cserére a weboldal keretén belül, ne csak az admin segítségével létrehozott jelszót lehessen használni.

Továbbá lehetőséget szeretnék biztosítani jogokkal való bejelentkezést, mely során nagyobb hozzáférési ranggal rendelkező felhasználók, elrejthetnek adott táblázatokat a nem kívánt személyek számára.

<https://github.com/HuNNi31/allamvizsga-2.1/tree/master>

Ábrák jegyzéke

3.1. Rendszer bemutató	16
4.1. Komponens diagram	21
4.2. Megjelenített táblázat letöltés	22
4.3. Táblázat módosítás excel táblázatból	23
4.4. Táblázat módosítása weboldalról	24
4.5. Adatbázis tervezése	25
5.1. Bejelentkező oldal	26
5.2. Fő oldal	27
5.3. Diákok oldal	28
5.4. Diák report készítés	29
5.5. Diák jelentés oldal	30
5.6. Diák import oldal	31
5.7. Kapcsolat az adatbázissal	32
5.8. SQL utasítás	33

Irodalomjegyzék

- [2ma] 2maggiesMSFT. <https://docs.microsoft.com/en-us/sql/reporting-services/>.
- [And16] Milan Andric. Web application as a support system for records of working time, monitoring business processes and activities of company employees. *Procedia Computer Science* 51, pages 514–519, 2016.
- [bM] Doc by Microsoft. <https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.gridview?view=netframework-4.8>.
- [CSS] CSS. <https://www.w3schools.com/css/>.
- [Dro] Ronald Drosier. <http://www.ronaldrosier.net/blog/2013/04/11/>.
- [Gan] Manimekalai Ganesan. <https://www.c-sharpcorner.com/article/create-rdlc-report-in-c-sharp-asp-net/>.
- [HaB] HaBo. <https://stackoverflow.com/questions/12521692/c-sharp-bulk-insert-sqlbulkcopy-update-if-exists>.
- [htt] Vetrivel D <https://youtu.be/0yHeholX4I>.
- [KP15] Justyna Bała Kopeć, Adrian and Anna Pięta. Webgl based visualisation and analysis of stratigraphic data for the purposes of the mining industry. *Procedia Computer Science* 51, pages 2869–2877, 2015.
- [Ren15] Steven Renders. Microsoft dynamics nav 2015 professional reporting. *Packt Publishing Ltd*, 2015.
- [Sha] Syed Shanu. <https://www.c-sharpcorner.com/uploadfile/asmabegam/asp-net-mvc-5-security-and-creating-user-role/>.
- [Zho19] Yawen Zhou. The network system of scientific research management in local undergraduate colleges and universities based on the web platform. *Big Data Analytics for Cyber-Physical System in Smart City*, pages 28–29, 2019.