

# Tartalomjegyzék

<b>1. Bevezetés</b>	<b>1</b>
<b>2. Követelmények és környezet</b>	<b>2</b>
2.1. Követelmények . . . . .	3
2.2. Környezet . . . . .	4
2.2.1. C# . . . . .	4
2.2.2. .NET Core Command-Line Interface (CLI) . . . . .	4
2.2.3. Microsoft Visual Studio 2015 Update 3 . . . . .	5
2.2.4. SQL . . . . .	5
2.2.5. Oracle Database Express Edition 11g Release 2 . . . . .	6
<b>3. Grafikus megjelenés</b>	<b>7</b>
3.1. Bejelentkező ablak . . . . .	7
3.2. Főablak . . . . .	8
3.3. Színek . . . . .	9
<b>4. Adatbázis kapcsolat</b>	<b>10</b>
4.1. Kapcsolatok típusai . . . . .	10
4.1.1. Helyi kapcsolat . . . . .	10
4.1.2. Távoli kapcsolat . . . . .	11
4.1.3. Miskolci Egyetem . . . . .	11
4.2. Adatkapcsolat felépítése . . . . .	12
4.3. Adatkapcsolat paraméterezése . . . . .	14
<b>Irodalomjegyzék</b>	<b>15</b>

# 1. fejezet

## Bevezetés

Szakdolgozatom célja egy termelő- és kereskedő vállalatok számára specializálható, grafikus felhasználói felülettel rendelkező adatbáziskezelő szoftver létrehozása volt, amellyel az említett vállalkozások raktározási és logisztikai ügyeit egyszerűen, átláthatóan, megelőző informatikai ismeretek nélkül is lehessen kezelni, azon a logikán alapulva, amit korábban papíron alkalmaztak.

Az alkalmazás elméleti jelentősége az iránymutatás a vállalatok részére, hogy a jelenleg is zajló, negyedik ipari forradalom jegyében lépjenek előre, és zárkózzanak fel a digitális technológia vívmányaival karöltve a hatékonyabb és kevesebb emberi munkavégzés érdekében, a logisztika, raktározás és a valós időben történő nyomkövetés használatával, akár személyi számítógépeken, akár mobil eszközökön. Megelőző kutatásaim és személyes tapasztalataim alapján tisztában vagyok vele, hogy digitális bevándorlóként sokkal nehezebb átlátni és megérteni egy számítógépes program működését, de éppen ezért támasztottuk a programmal szemben azt az alapvető követelményt, hogy könnyen értelmezhető, felhasználóbarát kezelőfelülettel rendelkezzen, úgy, hogy egy informatikában kevésbé jártas személy is megfelelően tudja használni.

A szoftver gyakorlati jelentősége az előző pontban említett elméleti előrelépés alkalmazása esetén valósul meg. A digitális átállás jelentősen leegyszerűsíti és felgyorsítja a munkát, valamint a digitális munkavégzés által csökkentett papírhasználat környezetbarát is.

Szakdolgozatom elméleti és gyakorlati jelentősége rám tekintve, hogy az eddig tanult, jellemzően fiktív adatokra épülő szoftveres megoldásokat már létező, valós problémakörre tudtam kiterjeszteni. Ehhez nagyban hozzájárult az, hogy az egyetemi alapképzés kereteiben megfelelő szintű tudást sajátítottam el ahhoz, hogy valós adatokkal dolgozó szoftvereket készítsek, amik gyakran nagy felelősséggel járnak.

## 2. fejezet

# Követelmények és környezet

A Földön élő emberek túlnyomó része rendelkezik valamilyen infokommunikációs eszközzel és a teljes lakosság 53,6%-a kapcsolódik az internethez. Ez az arány a Földet tekintve csekélynek mondható, különösképp azért, mert jelenleg már a negyedik ipari forradalom zajlik, de az érték azért ilyen alacsony, mert az ún. fejletlen-, fejlődő- és fejlett országok között óriási technológiai és fejlettségi szakadékok vannak. Afrikában ez az arány 18%, itt az emberek általában iskolákban, egyetemeken, nyilvános helyeken vagy közösségi házakban férnek hozzá az internethez, míg Európában ez az érték 84,2%, tehát már a „veteránok” és a „baby-boomerek” között is jócskán akadnak, akik alkalmazzák a technológiát, még akkor is, ha ők a legmesszebből érkezett digitális bevándorlók [1].

Az Európai Unió több tagállamában, többek közt Magyarországon is kötelezővé vált az elektronikus számlázás, így a vállalkozásnak rendelkeznie kell egy elektronikus számlát kezelő és kibocsátó szoftverrel [2], ám ez még nem jelenti azt, hogy minden vállalkozás tulajdonában van olyan alkalmazás is, amellyel a nyilvántartást, logisztikát hatékonyan tudják kezelni. Mivel a világ elindult az egységes digitalizáció útján - nem csak a szórakozás, hanem a munkavégzés terén is -, és a tendencia is jó irányba halad, ezért egy átlátható és hatékony megoldást hoztam létre a fejlődés útjára lépő vállalkozások számára.

Az adatmodellek, sémák és az üzleti logika kalakításakor több meglévő, nem digitális alapon működő nyilvántartást vettem alapul, így az átállás nem jár a meglévő készletek jelentős átcsoportosításával, esetleg néhány kisebb, logikai változás történik. A kezdeti adatfeltöltés időigényes művelet, de használat közben rövid időn belül belátják majd, hogy a jól strukturált adatkezelés, a gyors válaszidők, a hatékony keresési és szűrési funkciók jelentősen felgyorsítják és hatékonyabbá teszik a munkavégzést, hamar megtérül az adatbevitelbe fektetett idő és energia.

Az adatbáziskezelő szoftver jelenleg a termelő- és kereskedő vállalatok részére specializált, de az alapvető üzleti logika és szoftveres megoldás nem csak ezen területnek felel meg, így igény szerint más területen is alkalmazható, többek közt szolgáltató, pénzügyi, fuvarozó, vendéglátó és logisztikai vállalatoknál is. A szoftverbe és az adatbázisba implementált eljárások a megrendelő igénye szerint paraméterezhetők, maximálisan a vállalat működésére szabható megoldások létrehozásával.

## 2.1. Követelmények

A szoftverrel szemben támasztott alapvető követelmények:

- grafikus megjelenési felület,
- könnyen értelmezhető, felhasználóbarát megjelenés,
- gombok és menük alkalmazása, parancssor mellőzése
- egy adatbázissal történő valós idejű, oda-vissza irányú kommunikáció,
- az adatbázishoz kapcsolódás felhasználónévvel és jelszóval történő védelme (bejelentkezőrendszer),
- felkész- és késztermékek közös táblában történő tárolása, megkülönböztetésük egy mező értékével,
- bekerülési (bruttó) árból számított nettó, eladási és akciós ár számítása előre meghatározott matematikai képletekkel,
- a számított értékek védettsége,
- tetszés szerint választható „akciós” termékek nyilvántartása és lehetőség csak ezen termékek megjelenítésére,
- események részletes naplózása,
- napló védelme (utólag nem módosítható, nem törölhető),
- a megjelenített adatok közötti, egyszerre több érték szerinti szűrésének lehetősége,
- keresés az adatbázis elemei közt, egyszerre több mező értékének figyelembevételével,
- a megjelenített adatok tetszés szerinti rendezésének lehetősége,
- a megjelenített táblázatok futási időben történő, tetszés szerinti átrendezése,
- a felhasználó által bevitt adatok ellenőrzése mind adatbázis, mind szoftver oldalról,
- esetleges helytelen/hibás beviteli adatok esetén a hibás mező(k) jelölése, segítség a helyes szintaktikához,
- a program, az adatbázis és az adatkapcsolat által létrejött hibák kezelése.

## 2.2. Környezet

A fejlesztés kezdetén logikailag kellett megterveznem a szoftver működését, és ennek megfelelően kellett megválasztanom a hozzá szükséges környezetet. Ez magában foglalja a futtató operációs rendszert, a fejlesztés során használt programozási nyelvet, a futtatáshoz szükséges keretrendszert, a fejlesztői környezetet, valamint az adatbáziskezelő rendszert. Mivel Magyarországon az elmúlt egy évben és azt megelőzően is magabiztosan uralja a piacot a Microsoft Windows operációs rendszere [3], valamint saját munkáimra és tanulmányaim alatt is jellemzően ezt az operációsrendszert használtam, így a választásom egyértelmű volt.

### 2.2.1. C#

A program készítésekor C# nyelvet használtam, választásom azért esett erre, mert már egyetemi tanulmányaim előtt is használtam ezt a nyelvet, és a BSc alatt is ez volt a választott objektum-orientált programozási nyelvem, valamint a témavezetőm és általam kijelölt célok megvalósítására is alkalmas.

A C# egy erősen típusos, normatív, objektum-orientált programozási nyelv, amelynek alapjául a C++ és a Java szolgált. A fejlesztéskor a Microsoft a C++ hatékonyságát, a Visual Basic kezelhetőségét és a Java platform-függetlenségét próbálta ötvözni, amely sikerült is, 2000-ben, a Professional Development Conference-en mutatták be először a nyilvánosság előtt, és megjelenése óta már a hetedik verziójánál jár. Minden verzió számos újítással érkezett, többek között a párhuzamos programozás támogatásával. Megjegyzendő, hogy a C# fejlesztésének vezetője a kezdetektől az az Anders Hejlsberg, aki a Turbo és Borland Pascal, valamint a Delphi létrehozásakor főmérnökként dolgozott a Borlandnál. A nyelv olyan széles körben alkalmazott, hogy néhány hónappal megjelenése után, 2001 decemberében már szabványosították, erről az Európai informatikai és kommunikációs rendszerek szabványosítási szövetsége ECMA-334 [4] kódnévvel ellátott publikációjában olvashatunk, valamint 2003 óta ISO szabvány is lett, erről jelenleg a Nemzetközi Szabványügyi Szervezet ISO/IEC 23270:2006 [5] szabványának hivatalos kiadványában találunk bővebb információt, de a közeljövőben felváltja majd egy új változat, ami a kurrens újításokat is tartalmazza, ez a későbbiekben ISO/IEC DIS 23270 [6] néven lesz megtalálható.

### 2.2.2. .NET Core Command-Line Interface (CLI)

A már említett, 2000-es Professional Development Conference-en a C#-pal együtt a .NET Core Command-Line Interface (a továbbiakban .NET) keretrendszer is megjelent, ami alapfeltétel a C#-ban írt programok futtatásához. A Visual Studio 6.0 megjelenése óta a Microsoft nem a Win32 környezetben, hanem a .NET környezetben futtatja programjait, utalva ezzel a hálózati munka integrálására, amit az általam készített program is használ. A keretrendszer a használt fejlesztőkörnyezetnek fordítási idejű szolgáltatásokat végez, és az így lefordított alkalmazásoknak futási idejű környezetet biztosít, valamint az alkalmazások számára a már nem használt objektumok memóriabeli felszabadítását automatikusan elvégzi a Garbage Collector (szemetgyűjtő) használatával. A fordító a forráskódot nem natív, hanem egy köztes kódra fordítja le. Ez a köztes kód az MSIL (Microsoft Intermediate Language), ezt továbbfejlesztve, a Visual Studio 2015-ben hivatalosan is megjelent a .NET Compiler Platform (Roslyn), ami a Visu-

al Basic-hez és C#-hoz készült új, fordítói platform, ami közvetlenül a kódanalízist szolgálja, valamint készíthető vele különálló Form alkalmazás, amely a forráskódkezelő eszközökhöz kapcsolódva forráskódok megfigyelésére, statisztikáinak létrehozására használható.

A keretrendszer az alkalmazások típusai szerint több csoportba sorolja az osztálykönyvtárakat, most csak azokat említem, amelyeket érint az alkalmazásom:

- ADO.NET: Adatbázisrendszerek, és a hozzájuk tartozó driverek menedzselése és használata, valamint az adatkapcsolat kiépítése és működtetése
- Windows Form: Windows alapú felhasználói könyvtár, grafikus felhasználói felülettel rendelkező ablakok létrehozásához

A .NET keretrendszert is szabványosították Common Language Infrastructure (CLI) néven, ez az Európai informatikai és kommunikációs rendszerek szabványosítási szövetségénél az ECMA-335 kódnevet kapta [7] és a Nemzetközi Szabványügyi Szervezet az ISO/IEC 23271:2012 [8] kódszámmal jelölte, amikből egyértelműen látszik, hogy a C# és .NET szoros kapcsolatban állnak, kódszámaik közvetlenül egymást követik.

A szoftver futtatásához szükséges, feljebb említett keretrendszer a Windows Update-ből, és a MSDN-ről is elérhető, valamint a mellékelt lemezen is megtalálható a jelenleg legújabb, 4.7.2 verzió offline telepítője az `InstallMeFirst\NET472` könyvtárban, `NDP472-KB4054530-x86-x64-A110S-ENU.exe` néven.

### 2.2.3. Microsoft Visual Studio 2015 Update 3

Elsősorban az objektum-orientált szemléletmód és az Oracle adatbázissal történő megfelelő, robosztus működés volt a szempont a fejlesztői környezet kiválasztásakor. A környezet és a driverek megfelelő beállításainak elvégzése után a kapcsolat szilárd és a Visual Studio is megfelelően kezel minden, Oracle-lel kapcsolatos metódust.

A Visual Studio 2015 már több, mint 15 nyelvet támogat, és már nem csak Windows platformra lehet fejleszteni, hanem többek közt Azure, Android, iOS rendszerekre is, amellyel a alkalmazásom továbbgondolása esetén, és a jelenleg zajló negyedik ipari forradalom jegyében az adatbázis (legyen szó konkrétan készletezésről, leltárról, vagy bármilyen logisztikai műveletről) egy mobil eszközről elérhető úgy, hogy közben a felhasználó fizikailag ott van a raktárban, és az eszköz segítségével, egyedi azonosítókat használva valós időben, valós adatokkal dolgozhat, ide értve a készletfeltöltést – akár vonalkód szkenneléssel –, ellenőrzést, keresést, kivételezést.

### 2.2.4. SQL

Az SQL (Structured (English) Query Language) egy struktúrált, kötött szintaxisú, angol kulcsszavakon alapuló (nem csak) lekérdező programnyelv, amelyet minden jelentősebb relációs adatbáziskezelő rendszer használ (Microsoft SQL, Oracle Database, MySQL, PostgreSQL) [9]. A nyelvet a webes alkalmazások mellett számos PC-n futó adatbáziskezelő is alkalmazza, a nyelv olyan gyorsan bővül és fejlődik, hogy a jelenleg legújabb, ISO/IEC 9075:2016 [10] szabvány az azt megelőzőhöz képest öt év alatt 44 új funkcióval bővült [11].

### 2.2.5. Oracle Database Express Edition 11g Release 2

Az Oracle Database Express Edition az Oracle egyik ingyenes relációs adatbáziskezelője, amely nem csak konzolos, hanem webes interfésszel is rendelkezik, így a fejlesztés is könnyebb, és a felhasználók is relatíve könnyen, egyszerűen kezelhetik az adatbázist. Néhány adminisztrációval kapcsolatos feladathoz szükséges némi parancssori ismeret, de mivel a szoftver kiváló dokumentációkkal rendelkezik, az esetleges nehézségek is megoldhatóak. Mivel az Oracle rendelkezik olyan illesztőprogramokkal, amelyek a fejlesztőkörnyezettel kommunikálnak, így a webes felület elhagyható, elegendő egy ablakban történő kliens-szerver kapcsolat is, amely mellett - többek közt - a jóval kisebb memóriahasználat és az alacsonyabb erőforrásigény szól.

A tanulmányaim alatt az Adatbáziskezelés I-II. tantárgyakból az Oracle Database Express Edition adatbáziskezelő rendszert használtam, így számos funkcióját megismerhettem, megtanultam a kliens oldal használatát, különböző eljárásokat. A szakdolgozatomhoz szükséges szerver oldal konfigurációját pedig magam kellett elsajátítanom, a rendelkezésre álló dokumentációból (megjegyzendő, hogy az Oracle dokumentációi kiemelkedően jók és részletesek), így végül egy jól működő rendszert tudtam használni úgy, hogy mind a szerver, mind a kliens oldal működését a megfelelő szinten elsajátítottam.

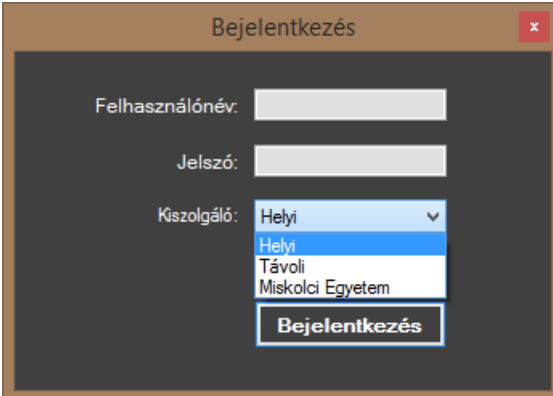
## 3. fejezet

# Grafikus megjelenés

A program működése során két fő ablakkal találkozhatunk, az egyik - és megjelenési sorrendben az első - egy bejelentkező ablak (LoginForm), a másik pedig maga a grafikus felületű adatbáziskezelő (MainForm). Ezen kívül találkozhatunk még felugró ablakokkal, ezek jellemzően valamilyen információ megjelenítése esetén vagy hibák kezelésekor jelennek meg, de ezek programozástechnikailag ún. üzenetdobozok (MessageBox), nem a klasszikus ablakok (Form), mivel ezek tájékoztatást adnak egy adott eseményről.

### 3.1. Bejelentkező ablak

A program indulásakor egy  $350 \times 250$  pixel méretű, átméretezést, teljes- és kis méretbe helyezést nem engedélyező bejelentkező ablak jelenik meg, ahol az inicializálás során két mező, egy legördülő menü, valamint egy gomb jön létre. Az első két mező a felhasználónév és a hozzá tartozó jelszó beírására szolgál, a legördülő menüből pedig kiválaszthatjuk a kiszolgálót, ez látható a 3.1 ábrán.



3.1. ábra. A bejelentkező ablak (LoginForm)

A sikeres bejelentkezés feltétele, hogy a beírt felhasználónév és jelszó létezzen az adott adatbázisban és a mezők értékei helyesen legyenek megadva.

A bejelentkező ablakban kiválasztásra kerülő kapcsolattípusok alapján a kiszolgáló lehet:



- **Helyi**, ebben az esetben a kliens és a szerverhez egy gépen található, egy időben futnak és . Bővebben a 4.1.1 pontban.
- **Távoli**, ebben az esetben a szerver és a kliens külön gépen (vagy virtuális gépen) található, és szintén egy időben futnak. Amennyiben ez az opció kerül kiválasztásra, a legördülő menü alatt megjelenik egy újabb mező, ahova az elérési címet adhatjuk meg, valamint mögötte egy információs ikon, ami ha rákattintunk, vagy fölé visszük az egeret egy segítségnyújtó üzenetet jelenít meg a helyes szintaxissal. Ha a kiszolgáltót valamelyik másik értékre állítjuk, a cím mező és vele együtt az információs ikon eltűnik. Bővebben a 4.1.2 pontban.
- **Miskolci Egyetem**, ebben az esetben a program a Miskolci Egyetem Informatikai Intézetében található Általános Informatikai Intézeti Tanszék dedikált Oracle szerveréhez kapcsolódik. Bővebben a 4.1.3 pontban.

## 3.2. Főablak

A főablak legnagyobb részét egy adattábla (DataGridView) tölti ki, az adatkapcsolaton keresztül ide érkeznek az adatbázisból betöltött értékek. Ennek megjelenése leginkább egy klasszikus táblázatkezelőhöz hasonlít, azzal a különbséggel, hogy itt az oszlopok nevei a fejlécben vannak jelölve, amik az SQL lekérdezéseken keresztül és a program által előre meghatározottan is megadhatóak. Az oszlopok szélessége dinamikusan, automatizált módon méreteződik a benne szereplő értékek alapján. A tábla elemeit az adott csoportosítás alapján rendezhetjük a rendezni kívánt oszlop fejlécére kattintva, kövekvő illetve csökkenő sorrendben, az értékek alapján.

A 3.2 ábrán a főablak látható közvetlenül a bejelentkezés után. Amennyiben a bejelentkezőablak mezőibe írt belépési adatok helyesek, a főprogram inicializálódása után rögtön megtörténik az SQL lekérdezés, amivel az adattábla feltöltődik.

Termék ID	F/K	Gyártó	Típus	Termék típus	Szín	Darab	Bekerülési ár	Nettó ár	Bruttó ár	Akció	Akciós ár	Hozzáadási idő
PRMELPIR	K	PROFILE	MINI SS FRONT	AGY ELSO	PIROS	2	20000	28000	35560	1	22400	2018.05.05. 16:11:56
PRMELFEK	K	PROFILE	MINI SS FRONT	AGY ELSO	KEK	5	20000	28000	35560	0	22400	2018.05.05. 16:11:56
MANPROFEH	K	MANKIND	PROMETHEUS	AGY ELSO	FEHER	4	12000	16800	21336	1	13440	2018.05.05. 16:11:57
MANPROPIR	K	MANKIND	PROMETHEUS	AGY ELSO	PIROS	4	12000	16800	21336	1	13440	2018.05.05. 16:11:57
MANPROOLA	K	MANKIND	PROMETHEUS	AGY ELSO	OLAJ	4	12000	16800	21336	1	13440	2018.05.05. 16:11:57
PRMELFEK	K	PROFILE	MINI SS FRONT	AGY ELSO	FEKETE	5	20000	28000	35560	0	22400	2018.05.05. 16:11:56
MANKPPPOL	K	MANKIND	PP	AGY ELSO	POLIR	3	8000	11200	14224	0	8960	2018.05.05. 16:11:57
CINSOLKRO	K	CINEMA	SOLID	AGY ELSO	KROM	7	6000	8400	10668	1	6720	2018.05.05. 16:11:57
CINSOLFEK	K	CINEMA	SOLID	AGY ELSO	FEKETE	5	6000	8400	10668	0	6720	2018.05.05. 16:11:57
PRMELKRO	K	PROFILE	MINI SS FRONT	AGY ELSO	KROM	5	20000	28000	35560	0	22400	2018.05.05. 16:11:56
MANKPPFEK	K	MANKIND	PP	AGY ELSO	FEKETE	4	8000	11200	14224	0	8960	2018.05.05. 16:11:57
ODYSV3FEK	K	ODYSSEY	V3	AGY HATSO	FEKETE	3	18000	25200	32004	0	20160	2018.05.05. 16:11:57
ODYSV3ZOL	K	ODYSSEY	V3	AGY HATSO	ZOLD	5	18000	25200	32004	1	20160	2018.05.05. 16:11:57
PRIREMPOL	K	PRIMO	REMIX	AGY HATSO	POLIR	6	23000	32200	40894	0	25760	2018.05.05. 16:11:57
PRIREMFEK	K	PRIMO	REMIX	AGY HATSO	FEKETE	4	23000	32200	40894	0	25760	2018.05.05. 16:11:57
PRIMXPOL	K	PRIMO	MIX	AGY HATSO	POLIR	4	20000	28000	35560	1	22400	2018.05.05. 16:11:57
PRIMXFEK	K	PRIMO	MIX	AGY HATSO	FEKETE	3	20000	28000	35560	1	22400	2018.05.05. 16:11:57
CINEVXLIL	K	CINEMA	VX	AGY HATSO	LILA	1	22000	30800	39116	1	24640	2018.05.05. 16:11:57
CINEVXNAR	K	CINEMA	VX	AGY HATSO	NARANCS	2	22000	30800	39116	1	24640	2018.05.05. 16:11:57
CINEVXFEK	K	CINEMA	VX	AGY HATSO	FEKETE	4	22000	30800	39116	0	24640	2018.05.05. 16:11:57
PRMIHAKRO	K	PROFILE	MINI SS REAR	AGY HATSO	KROM	5	30000	42000	53340	0	33600	2018.05.05. 16:11:57
PRMIHAFEK	K	PROFILE	MINI SS REAR	AGY HATSO	FEKETE	5	30000	42000	53340	0	33600	2018.05.05. 16:11:57
ODYSV3FEH	K	ODYSSEY	V3	AGY HATSO	FEHER	3	18000	25200	32004	1	20160	2018.05.05. 16:11:57
PRMIHAFEK	K	PROFILE	MINI SS REAR	AGY HATSO	KEK	5	30000	42000	53340	0	33600	2018.05.05. 16:11:57
PRMIHAPIR	K	PROFILE	MINI SS REAR	AGY HATSO	PIROS	2	20000	28000	35560	1	22400	2018.05.05. 16:11:57

3.2. ábra. A főablak (mainForm)

### 3.3. Színek

A felhasználói felület kialakításakor a könnyű kezelhetőség mellett törekedtem a színeket úgy beállítani, hogy huzamos ideig történő használat esetén is kímélje a szemet, mégis kontrasztos és átlátható maradjon.

A bejelentkező ablak háttérszíne sötétszürke, az ezen található mezők és gombok alapszínei világosszürkék, a mezőket és a legördülő menüt kitöltő betűszín fekete, egyedül a feliratok fehérek, a jobb kontraszt érdekében. A bejelentkező gomb alapszíne megegyezik a háttérszínnel, valamint az ezen található felirat fehér, ez is a jobb kontraszt érdekében.

A főprogram háttérszíne megegyezik a bejelentkező ablakéval, a mezők és menük alapszíne világosszürke, és a benne lévő betűszín fekete. A főablakban található adattábla színeinél is törekedtem arra, hogy a szemet kíméljem, mivel elég nagy részt foglal el az ablakból. Ennek színei is sötétszürkék, a cellák hátterei sötétebbek, míg a fejléc és a sorjelölők világosabb szürkék, a megkülönböztethetőségért. A kijelölés színe kék, mert az a kedvenc színem.

## 4. fejezet

# Adatbázis kapcsolat

Az Oracle Database Express Edition adatkapcsolat szempontból két fő komponensből áll, az egyik a szerver, a másik a kliens. Ez egy hierarchiailag alárendelt kapcsolat, a kliens kapcsolódik a szerverhez, adatokat kér, amiket a szerver feldolgoz és - ha a kliens megfelelő jogosultsággal rendelkezik - kiszolgál. A megfelelő működéshez egy szervernek futnia kell, amihez kapcsolódhat(nak) a kliens(ek). A szerver és kliens települhet ugyan azon eszközre (sőt, egyszerre lehet telepítve 32 és 64 bites verzió is, ami több esetben javasolt is), valamint futásuk is történhet egy időben és egy számítógépen, így a kis infrastruktúrával és alacsony költségvetéssel rendelkező vállalkozásoknak is megfelelő megoldás lehet. A több klienst használó vállalatok pedig jellemzően egy külön erre dedikált szerverhez kapcsolódnak, és arra nem is kötelező az adatbázis kezelő szoftver telepítése, mert az Oracle beépített adminisztrációs felületéről (rendelkezik grafikus felhasználói felülettel és parancssorral is) a legtöbb szükséges teendőt el lehet végezni, ám a szervert érintő rendszerbeállítások kizárólag parancssoron keresztül érhetőek és vezethetők el.

### 4.1. Kapcsolatok típusai

#### 4.1.1. Helyi kapcsolat

Amennyiben a szerver és a kliens egy számítógépen fut, a kapcsolat típusa *Helyi kapcsolat*, tehát a klienssel egy lokális kiszolgálóhoz kapcsolódunk. Ilyen esetekben - bár egy eszközön futnak - a kliens és a szerver logikailag elkülönül egymástól, viszont a hálózati címzés a helyi hálózaton át valósul meg, tehát az elérés történhet a 127.0.0.1 IP címen, vagy a localhost néven keresztül. Ez hasznos lehet több klienst használó vállalat esetén, ahol a kliens gépek a lerakatokban, vagy külső raktárakban találhatóak, de a központi raktárban vagy szerverszobában található fő szervergépről is elérhető és menedzselhető az adatbázis.

Bizonyos esetekben fennálhat a vállalat kérése, hogy csak a helyi kapcsolatot használó számítógépről legyen elérhető a napló tábla, ami a beépített eszközzel is elérhető, de célszerű ilyenkor is az alkalmazást használni, a már megszokott kezelőfelület miatt. Ez néhány soros módosítás után könnyen megvalósítható, így a kliens és szervergépeken futó programok már külön instanciák.

### 4.1.2. Távoli kapcsolat

Távoli kapcsolat esetén a felhasználónak lehetősége van egy általa meghatározott kiszolgálóhoz kapcsolódni, ehhez szüksége van az adott szerver IP címére, portjára és a rajta futó adatbázis verziójának megnevezésére. Ennek formálisan is meg kell felelnie, de a szintaktikához segítséget nyújt a bejelentkező ablakon található információs ikon. Alkalmas kliens gépek esetén külső raktárakból történő hozzáférésre, valamint adott szerver elérésére bárholnan, amennyiben a szerver és a kliens online, az átjáráshoz használt port nincs blokkolva a tűzfal által, továbbá a felhasználó rendelkezik érvényes hozzáféréssel az adatbázishoz.

Amennyiben az előző pontban említett kérés fennáll a vállalat részéről, úgy a csak távoli kapcsolattal rendelkező instanciák jellemzően nem kapnak jogot arra, hogy a napló táblát lássák. A napló tábla egyébként is alacsony szinten védett, de így információhoz sem juthatnak azok, akiknek egyébként sem lenne hozzá jogosultságuk.

A távoli kapcsolat kézi beállításához segítség a következő, 4.1.3 pontban.

### 4.1.3. Miskolci Egyetem

Ez az opció a szoftver megjelenésének és működésének bemutatásához készült. Ebben az esetben a szoftver a Miskolci Egyetem Általános Informatikai Intézeti Tanszékén található Oracle szerveréhez kapcsolódik, ez a kapcsolat jellemzően a tanulmányi időszakokban érhető el, akkor megbízhatóan működik. Az alkalmazás, az információ- és adatátvitel, valamint a hozzá tartozó kapcsolat megfelelően tesztelhető rajta. Az ehhez tartozó belépési adatok:

Mivel ez is egy távoli elérés (a külön listázás csak a kényelmesebb használat érdekében történt), ugyan ez az adatkapcsolat kipróbálható a *Távoli* opció használatával is, a belépéshez szükséges adatok, valamint az egyetemi Oracle adatbázis szerver elérhetősége:

Felhasználónév: *duritt*

Jelszó: *t6716g*

Cím: 193.6.5.58:1521/XE

Az adatbázis, amelyhez a szoftver ezen a kapcsolaton keresztül kapcsolódik elérhető a Miskolci Egyetem által a tanulmányi időszakban biztosított szerver APEX felületén is, ahol grafikusan és parancsokkal is módosíthatjuk az adatbázis tartalmát, valamint lekérdezéseket, kimutatásokat, metaadatokat is lekérhetünk, emellett web-es alkalmazást is generálhatunk ezen keresztül. Ennek elérése:

«ellenorizni»

Cím: 193.6.5.58:8080/apex

Munkaterület: *duritt*

Felhasználónév: *admin*

Jelszó: *t6716g*

## 4.2. Adatkapcsolat felépítése

Az Oracle adatbázis-kapcsolat felépítésénél az elnevezések tetszés szerintiek lehetnek, de célszerűek a beszédes nevek, hogy közben, vagy esetleg a későbbiekben a kód a fejlesztő számára átlátható legyen (*kivételt képeznek ezalól a nagy költségvetésű fejlesztést igénylő szoftverek, amelyeknél a cél a visszafejtés lehető legnehezebbé tétele*), felépítés szerint:

- `OracleConnection con;`

Egy `OracleConnection` típusú objektum, ami valójában egy kapcsolatleíró string, mely az Oracle adatbázisokkal való kapcsolatot kezeli.

- `DataSet ds = new DataSet();`

A `DataSet` példányosítása a lekérdezett adatok adattáblában történő tárolásához. Ez egy adatbázis a memóriában, amelynek szerepe, hogy az adattábla osztályban létrehozott objektumokat tárolja.

- `DataTable dt = new DataTable();`

Az adattábla (`DataTable`) objektum `dt` példánya egy logikai táblát hoz létre a memóriában az adatbázis-szerverről áthozott adatok alapján. Amikor ez megtörténik, a program közvetlenül innen olvas majd, és a felhasználó által létrehozott módosítások is itt tárolódnak. Ennek tartalma lehet:

- *Táblák:* Logikailag összetartozó értékek halmaza, elrendezése sorokban és oszlopokban történik.
- *Mezők:* Cellák, melyekbe az adott cella típustól függően adatot vihetünk be. Itt értékellenőrzés történik, azaz egy szám típusú cellába nem írhatunk karaktersorozatot.
- *Sorok:* Jellemzően egy adott rekordhoz kapcsolódó értékek sorozata, a logikai szeparálást segítve külön cellákba rendezve, így a sorban lévő mezők alkotnak oszlopokat.
- *Oszlopok:* Az említett elrendezés függőlegesen összetartozó elemei, például telefonszámok egymás alatt. Létezhetnek oszlopok, amelyek elemeinek különbözőeknek kell lenniük, jellemzően ezek az elsődleges kucskok, egyedi azonosítók (ID).
- *Nézetek:* Lekérdezés vagy több tábla kiválasztott oszlopai egymás mellett. Használatuk olyan esetekben történik, amikor több táblából szeretnénk adatokat egymás mellett úgy látni, hogy egy új nézőpontból közelíthessük meg a vizsgált tartalmat.
- *Indexek:* A táblában lévő keresés felgyorsítására alkalmazott eszköz, jellemzően nagy méretű adatbázisokban használják, megvalósításai: hash-kód vagy bináris fa. Az optimalizálás és a megfelelően alkalmazott indexelés a nagy méretű adatbázisokban a futási időt  $\mathcal{O}(n)$ -ről  $\mathcal{O}(\log n)$ -re csökkenti. Ez azt jelenti, hogy a nem megfelelően indexelt adatbázisok lineáris (az elemek számával egyenes arányban lévő) futási idejűek, tehát a az elemszámok növekedésével a futási idő is azonos mértékben nő, így ha az elemszám a duplájára nő, a futási idő is megduplázódik. A megfelelően indexelt adatbázisok esetén a futási idő logaritmikus, ami gyakorlatilag konstansnak vehető a logaritmus függvény miatt.

- *Kényszerek*: A kényszerek a táblákra és az abban szereplő oszlopokra vonatkozó, a megfelelő működés, konzisztencia biztosítását szolgáló szabályok. Több felhasználós környezetben a párhuzamos hozzáférés miatt problémás lehet, például ha egy időben szeretnének létrehozni megegyező azonosítójú terméket. Szintaktikailag a DDL része.
  - *Kapcsolatok*: Ahogy a relációs adatbázis elnevezés is utal rá, ezen adatbázis működésének alapja a táblák, mezők és értékeik közti kapcsolat, így ezeket is tárolni kell.
  - *Metaadatok*: Adatok az adatokról, amelyek a hatékonyság, integritásőrzés, adatvédelem biztosítását szolgálják.
- `OracleDataAdapter da = new OracleDataAdapter();`  
Ez az összekötő kapocs a fizikai adathalmaz és a memóriában tárolt, strukturált adathalmazok között.
  - `OracleCommandBuilder builder = null;`  
Egy nem örökölhető osztály, amely automatikusan generálja azokat a parancsokat, amelyekkel a felhasználó által, futásidőben történő adatmódosításokat összeegyezteti a `DataSet`-ben lévő értékekkel.
  - `OracleCommand cmd = new OracleCommand();`  
Az ehhez kapcsolódó átadott paraméterekkel (`commandText` (a lekérdezés szövege), `OracleConnection connectionString` (a kapcsolódást biztosító string)) egy új példányt hoz létre az `OracleCommand` osztályon belül, amellyel a lekérdezést majd végrehajtja.
  - `string connectionString = String.Format("User Id={0};Password={1};" + "Data Source={2};Pooling=false;", input_uid, input_pw, server_address);`  
Ez a kapcsolódást biztosító string, azaz karaktersorozat.  
Jelen esetben egy formázott stringről van szó, mert az azonosító, jelszó és az adatforrás értékei a felhasználó általi input alapján kerülnek átadásra, de lehet előre definiált, statikus érték is. Erről bővebben a 4.3 szekcióban.
  - `con = new OracleConnection(connectionString);`  
Ez az `Oracle` osztályon belül létező *Connection* típus, ami az adatbázis eléréséért felel. Használatakor egy új példány jön létre az osztályban. Ez metódusként használható, így a zárójelben átadott `conString`-gel a kapcsolódáshoz szükséges adatok is átadásra kerülnek. Ennek értékei kerülnek átadásra a már létrehozott `con` változóba, és a későbbiekben ennek használatával jön létre a közvetlen adatkapcsolat.
  - `con.Open();`  
Ha a példányosított `OracleConnection` megfelelő adatokkal rendelkezik, a `.Open()` metódus létrehozza a kapcsolatot a kliens és a szerver között. Amennyiben nincsenek megfelelő adatok, a metódus akkor is lefut, de a kapcsolódás sikertelen lesz.

## 4.3. Adatkapcsolat paraméterezése

A bejelentkező ablakon keresztüli bejelentkezéskor a kapcsolatot és az adatbázishoz való hozzáférés érvényességét maga az adatbázis-szerver ellenőrzi. A LoginFormba beírt felhasználónév és jelszó, valamint a manuális kapcsolódás esetén a kézzel beírt elérési cím (a helyinél és a távolinál is létezik cím, de ez a minidig ugyan az, így nem szükséges sem kézzel beírni, sem megjeleníteni) egy a kapcsolat felépítéséhez elengedhetetlen, a kapcsolódáshoz tartozó azonosítási és kapcsolódási adatokat tartalmazó stringbe (connectionString) kerül, így minidig a bejelentkező ablakba írt érték és a választott kapcsolat alapján, dinamikusan kap értéket.

A paraméterátadás megvalósítása a forráskódban:

```
string conString = String.Format(
    "User Id={0};Password={1};" +
    "Data Source={2};Pooling=false;",
    input_uid, input_pw, server_address);
```

ahol:

- *{0}* a felhasználónév (minden esetben a mezőbe írt érték adódik át),
- *{1}* a jelszó (minden esetben a mezőbe írt érték adódik át),
- *{2}* az adatkapcsolat címe (itt csak akkor adódik át a mezőbe írt érték, ha a listából a *Távoli* van kiválasztva, egyébként választástól függően (*Helyi* vagy *Miskolci Egyetem*) előre definiált értékek adódnak át).

Ha az alkalmazást megrendelő fél nem szeretne a szoftverbe beléptető funkciót (ami nem javasolt, de a védelem megoldható különböző operációs rendszerbeli mechanizmusokkal, harmadik fél által készített szoftverekkel, vagy akár az eszköz teljes izolálásával, ugyanis a lokális szerveren futó adatbázishoz nem szükséges aktív internetkapcsolat), abban az esetben a connectionString felépítésének és tartalmának nem szükséges dinamikusnak lennie, statikus módon is megoldható:

```
string conString = "User Id=Vallalat001;Password=Jelszo123;" +
    "Data Source=127.0.0.1:1521/XE;Pooling=false;"
```

ahol:

- a felhasználónév: Vallalat001
- a jelszó: Jelszo123
- az adatforrás: 127.0.0.1:1521/XE, amiből:
  - a szerver IP címe: 127.0.0.1
  - a port: 1521
  - az adatbázispéldány neve (SID): XE

# Irodalomjegyzék

- [1] International Telecommunication Union: *ICT Facts and Figures 2017*. International Telecommunication Union, 2017.
- [2] Nemzeti Adó- és Vámhivatal, *Online Számla*, <https://onlineszamla.nav.gov.hu>, Internet, 2018.
- [3] StatCounter, *Operating System Market Share Hungary: July 2017 - July 2018*, <http://gs.statcounter.com/os-market-share/all/hungary>, Internet, 2018.
- [4] ECMA International: *ECMA-334 C# Language Specification, 5th Edition*, ECMA International, 2017.
- [5] International Organization for Standardization, *ISO/IEC 23270:2006(E), Second Edition*, International Organization for Standardization, 2006.
- [6] International Organization for Standardization, *ISO/IEC DIS 23270*, <https://www.iso.org/standard/75178.html>, Internet, 2018.
- [7] ECMA International: *Common Language Infrastructure (CLI), 6th Edition*, ECMA International, 2012.
- [8] International Organization for Standardization, *ISO/IEC 23271:2012(E), Third Edition*, International Organization for Standardization, 2012.
- [9] Kovács László: *Adatbázisok tervezésének és kezelésének módszertana*, Computer-Books, Budapest, 2004.
- [10] International Organization for Standardization, *ISO/IEC 9075:2016, Fifth Edition*, International Organization for Standardization, 2016.
- [11] Modern SQL: *What's New in SQL:2016*, <https://modern-sql.com/blog/2017-06/whats-new-> Internet, 2017.



## CD-melléklet tartalma

A dolgozat PDF változatát a `dolgozat.pdf` fájlban találjuk.

A dolgozat  $\text{\LaTeX}$  segítségével készült. A forrásfájlok a `dolgozat` jegyzékben találhatók.

A dolgozathoz 4 művészi szűrő került implementálásra. Ezek a `filters` jegyzék alábbi aljegyzékeiben kaptak helyet.

- `cartoon-style`
- `pencil-sketch`
- `cartoon`
- `aquarelle`

A példákban szereplő minták a forrásfájlok mellett találhatók. A program lefordítása után az közvetlenül kipróbálható.