Debreceni Egyetem

Természettudományi Kar

Adatbázis alapú alkalmazás fejlesztése Delphiben

Szakdolgozat

Készítette:

Lukács Sándor

IV. programozó matematikus

Témavezető

Dr. Bajalinov Erik

Debrecen

2007

Tartalomjegyzék

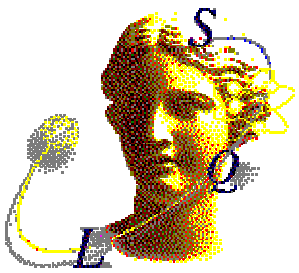
Bevezetés

A mai dinamikusan fejlődő világban már nem létezik olyan terület, ahol ne jelent volna meg a számítógép, körbe vesz minket, hordjuk a zsebünkben, szórakozunk, telefonálunk velük, és az internet révén a távolságok is megszűntek. Pedig ha történetét nézzük, nem rendelkezik valami nagy múlttal. Fejlődésének sebessége szinte hihetetlen, pedig a kezdetek nem adtak okot optimista kilátásokra. A mai modellekhez képest nagyon lassúak voltak, hatalmasak némely terem méret nagyságú volt, az üzemórájuk pedig messze elmaradt a tőlük. Ennek ellenére rengeteg pénzt, időt és energiát fordítottak a fejlesztésére, de miért? Ennek oka igen összetett, könyveket lehetne róla írni, de mégis, személy szerint én a nagyobb hatékonyságot és emberi kényelmi érzetét határoznám meg okokként. Az ember rengeteg energiát képes olyan eszközök előállítására fordítani, amellyel megtehet valamit, amit addig nem, vagy csak lassabban tehette meg. A kulcsszó a gyorsaság, kevesebb idő és/vagy energia-befektetéssel elérni ugyanazt. Először a hatékonyság dominált, bonyolult számítási feladatok elvégzésére használták a számítógépet, de később megjelent a kényelem, mely szélesebb körben is alkalmazható: a számítógépes adatnyilvántartás. A papírmunka valahogy az unalom egyik szinonimája, melyet unalmasnak és hosszadalmasnak tartunk, a lehető legkevesebbet szeretnénk vele foglalkozni. A számítógép felhasználására ezen a területen nem kellett sokat várni, elég hamar megjelent, és egyre inkább dominál. Nemcsak tároljuk rajtuk az adatainkat, fel is dolgoztatjuk azokat, az adatbázis-kezelők megjelenése robbanásszerűen növelte a felhasználhatóságot. Később előtérbe kerültek a felhasználók, mivel az adatbázis-kezelők használatát el kellett sajátítani, ezért az átlag ember nem tudta használni azokat, hacsak rá nem szánta az energiát és az időt, hogy megtanulja. Keretprogramokat fejlesztettek, hogy minél kevesebb energia-befektetéssel és időráfordítással tudja használni azokat. Mára a piac tele van olyan szoftverekkel, melyekkel nyilvántarthatjuk saját és céges adatainkat. A nagy fejlesztőcégek ontják magukból ezeket, bár nagyon jól működnek, hatékonyak, ennek ellenére, mivel emberek készítették őket, lehetnek bennük hibák, és nem feltétlenül illeszkednek a felhasználók egyedi igényeihez. Azok, akik konkrét elképzeléssel rendelkeznek egy szoftverrel kapcsolatban, valószínűleg nem fognak tudni vásárolni olyat, mely ezeknek megfelel. Véleményem szerint jobban járnak, ha személyesen keresnek egy szoftverfejlesztő céget, akik az igényeik alapján elkészítik a cégüknek megfelelő

programot, ez lehet, hogy olcsóbb vagy drágább de fő szempont a sokkal nagyobb hatékonyság. Szakdolgozatom elkészítése során és egy kisvállalkozás raktárkészlet kezelő rendszerét próbáltam megvalósítani. Mely tárolja a cikktörzset, alkalmazottakat, gépjárműparkját alkalmas még bejövő kimenő számlakezelésre, menetlevél átlagfogyasztás kalkulálásra minél nagyobb hatékonyság elérése végett.

Bevezetésként ennyit szerettem volna mondani, a továbbiakban áttérek a Delphi eszközeire, a program fejlesztésének technikai részleteire, és a program működésének bemutatására.

Delphi bemutatása.



A Delphi rendszer a ma használatos programozási feladatok minden területén hatékonyan alkalmazható fejlesztőeszköz. A Delphi fejlesztőinek célja egy könnyen gyors, megbízható programok (alkalmazások) fejlesztését lehetővé tévő eszköz készítése volt. Ezeken belül is kiemelkedő hangsúlyt kapott az *adatbázis alkalmazások* készítésének elősegítése. Nézzük most meg a fejlesztő – a programozó – szemszögéből ezt az eszközt.

A Delphi megtervezésekor az egyik jelszó a következő volt: *Legyen látható (és elkészíthető) tervezési időben, ami csak lehet!*

- vizuális tervezés és a kódgenerálás szervesen összekapcsolódik
- A fent említett összerendelésnek a megvalósítására vezette be a Delphi a komponens fogalmát. *A komponens egy osztály, amely valamely látható* (pl. párbeszédpanel vezérlőelemek) *vagy nem látható* (pl. adatbázis elemek, nyomtató, rendszereszközök, kivételek, sőt maga az alkalmazás) *erőforrás interface - étvalósítja meg*.

Delphi fejlesztő eszköz kialakulásának rövid történelmi lépései

- **Wirth Pascal**: Wirth az Algol 60-ból kiindulva specifikálja a Pascalt.
- **Turbo Pascal**: A Borland megjelenteti az első Pascal-verzióját, amely nagyjából a Standard Pascal szolgáltatásait kínálta.
- **Turbo Pascal 3.0**: Már nem egyszerű sorfordító.
- **Turbo Pascal 4.0**: Elkészül az IDE őse. Megjelenik a modularitás (unit-ok).
- **Turbo Pascal 5.0**: A DOS-os IDE kialakul. Még nem OOP.
- **Turbo Pascal 5.5**: MEGJELENNEK AZ OOP ALAFOGALMAI.
- **Turbo Pascal 6.0**: Már integrált OOP, de még nem ismer olyan alapvető fogalmakat, mint információ-elrejtés.
- **WinPas 1.0**: Az első Windows alapú verzió.
- **Borland Pascal 7.0**: Komolyabb OOP támogatás, de még mindig információ-elrejtés nélkül.

- **Delphi 1.0**: Új szemlélet, osztályfogalom, információ-elrejtés, komponens, kivételkezelés, property. Az első valóban OOP szemléletű verzió. (Itt már nem egy kiegészítő eszköz az OOP, hanem a nyelv szerves része!). Legmegkapóbb szolgáltatásai a form alapú és valóban objektumközpontú megközelítés, a villámgyors fordítóprogram, a nagyszerű adatbázis támogatás, a hagyományos Windows – programozással való szoros együttműködés, és a komponensek voltak. Mindezek hátterében pedig az egyszerre erőteljes és rugalmas Object Pascal nyelv ált.

- **Delphi 2.0**: 32-bites változat; többszálú programozás; az űrlap nem csak TForm komponens lehet, hanem ennek leszármazottja is (virtuális űrlapöröklés); debuggerét kibővítették a taszk állapot figyelésével; Object Repository; a Visual Form Inheritance segítségével virtuálisan származtathatunk párbeszédpaneleket a teljes űrlap kódjából; adatbázis tallózó; Új string típusok (AnsiString, ShortString). Multi-Record objectum,

továbbfejlesztett adatrács. Az OLE Automation, a variant adattípus, és a Windows 95 teljes körű támogatása, a long string adattípús.

- **Delphi 3.0**: Sok új eszköz (Visual Component Library, SQL Explorer); újabb technikák (ősosztály metódusának újradefiniálására, saját komponensek adhatók a bázisosztályhoz); több mint száz előre definiált komponens (Tree View, Rich Edit, List View); ISAPI és NSAPI DLL- ek készítése. Adatbázis platformok támogatása (Oracle, Sybase, Informix, DB2) natív módon, vagy tetszőleges ODBC adatbázis használata; Lehetőség ActiveX komponensek használatára:

    - ActiveX Creation: segítségével egy lépésben létrehozhatunk ActiveX elemeket
    - Active Forms: a Delphi alatt készített ablakokat alakítja át Internetes ablakká
    - Active Web Deployment: átalakítja az alkalmazásokat Web alatti alkalmazássá
    - COM: könnyen kezelhető környezet COM és DCOM fejlesztéshez

    Broker Technologies:

    - Remote Data Broker: adatátadás engedélyezése a kliens számára
    - Business Object Broker: többszálú objektumok tárolása arra az esetre, ha a kapcsolat megszakadna
    - Constraint Broker: segítségével akkor is lehet dolgozni, amikor éppen nincs kapcsolat az adatbázissal
    - Web Broker: információk gyors elterjesztése a Web - en HTTP objektumokat tehetünk a saját formunkra, és átállíthatjuk ennek attribútumait.

- **Delphi 4.0:** Windows 98 új komponenseinek támogatása is beépült e verzióba. Újfajta toolbar-ok: (végre) lehetőség van az Office 97-hez hasonló dokkolható toolbar-ok

használatára Új IDE, jobban áttekinthető forráskód az AppBrowser-rel A forráskódban hiperlink - eket helyezhetünk el így könnyebb az áttekintés A Code Explorer is segít akár a jobb kód áttekintésben vagy a dokumentálásban A könnyebb hibakeresést sok adalék segíti Támogatja az alábbi MS szabványokat: MTS, ISAPI, COM / DCOM, ActiveX. Továbbá az NT Service-t is közvetlenebb módon, wizard-dal támogatja.

- **Delphi 5.0**: A Delphi 5 annyi újdonságot tartalmaz, hogy felsorolni is nehéz lenne. Csak néhány ezek közül: - bővített adatbázis-támogatás (ADO és InterBase adatkészletekhez) - új MIDAS változat internetes szolgáltatásokkal. A TeamSource változatkezelő eszköz, a más nyelvre való fordítás lehetősége, illetve a keretek, valamint számos új komponenst is tartalmazott.

- **Delphi 6.0**: Mindezen szolgáltatásokat – a Component Library for Cross-Platform (CLX) segítségével kiegészítve a rendszer független fejlesztés lehetőségével, valamint a bővítette a futásidejű könyvtárat is. Bevezette a dbExpress adatbázis monitort, kivételes támogatást nyújtott a Web szolgáltatásokhoz és az XML- hez, s emellett erőteljes webfejlesztési keretrendszerrel, barátságosabb keretrendszerrel és seregnyi új komponenssel rendelkezett.

- **Delphi 7.0:** Az új szolgáltatásokat még hatékonyabbá és megbízhatóvá teszi a SOAP-támogatás a DataSnap bevezetés, és a legújabb megoldásokhoz az például az XP – témák támogatásához is segítséget nyújt, de ami talán a legfontosabb, hasznos külső eszközök egész sorát biztosítja, a RAVE jelentéskészítő motorról az INTRAWEB webalkalmazás fejlesztőn keresztül a ModelMaker tervezési környezetig. Végezetül azzal, hogy megjelent benne a Borland első olyan Pascal / Delphi – fordítóprogramja, amely nem az Intel típusú processzorhoz, hanem a. NET CIL platformjához készült. Ezáltal a Delphi egy új világra is ablakot nyitott.

- **Delphi 2005**: Több nyelv és Windows SDK támogatása - A Delphi 2005 biztosítja a modern Windows fejlesztéshez szükséges nyelveket és SDK támogatást. Mivel mind a Delphi, mind pedig a C# fejlesztést támogatja, ez az egyetlen igazi olyan Windows termék, amely ugyanazon eszközből és ugyanazon nyelv alapján (Delphi) támogatja a natív Win32 és. NET fejlesztést. Ugyanakkor az ASP. NET, ADO. NET, VCL. NET és VCL for Win32 megoldásokat is támogatja. ALM megoldások integrációja - A Delphi 2005 célja, hogy a fejlesztőknek a StarTeam® és az Optimizeit™ integrációjával rálátást biztosítson az alkalmazás-életciklus különböző fázisaira. A StarTeam integráció célja, hogy leegyszerűsítse a forráskód erőforrások menedzsmentjét és növelje a csapat kommunikációját, míg a mellékelt Optimizeit Profiler for .NET segít az egységtesztek automatizálásában, valamint az alkalmazás minőségének és teljesítményének általános továbbfejlesztésében. Gyors vállalati MDA fejlesztést tesz lehetővé - A Delphi 2005 ECO II megoldása vállalati szintű gyors modell alapú architektúra (Model Driven Architecture, MDA) megoldást biztosít a NET- hez, amely lerövidíti a bonyolult alkalmazások fejlesztését, javítja minőségüket, és megnöveli karbantarthatóságukat. Az ECO II az objektumok önműködő diagrammszerű ábrázolásának, valamint létrehozatalának teljes megoldása, amely rugalmasan méretezhető, fejlett vállalati objektum funkciókkal (pl. visszavonás/ismétlés, verziókezelés és tranzakciók) ellátott. NET objektum gyorsírótárat kínál. Leegyszerűsíti és lerövidíti a Windows fejlesztést - A Delphi 2005 számos innovatív IDE funkciót kínál, amely hozzájárul a napi fejlesztői munka megkönnyítéséhez, megnöveli a hatékonyságot, és leegyszerűsíti a kód karbantartását. Olyan funkciókat kínál, mint a fejlett kód refaktorizálás, Help Insights és Error Insights (súgó és hiba vizsgálat), SyncEdit (szinkronizált szerkesztés), History Management (régi elemek menedzsmentje), és a Delphi nyelv új továbbfejlesztései. A Delphi Advantage for ADO. NET célja, hogy az adatbázisokhoz kapcsolódó. NET alkalmazások fejlesztését minden szempontból lerövidítse és leegyszerűsítse mind Delphi, mid pedig C# alatt.

És végül a legújabb fejlesztés a Delphi 2007

- A Borland Software - ból a múlt év végén kivált CodeGear kedden jelentette be Delphi nevű, integrált és gyors fejlesztést lehetővé tevő környezetének legújabb kiadásait. A két friss Delphi közül az egyik a Windows legújabb kiadására, a Vista-ra történő fejlesztést teszi lehetővé a programozók számára, míg a másik kiadás a webes projektekben használt PHP - ben dolgozók számára jelenthet igazi csemegét majd. A Delphi 2007 for Win32 a fejlesztőrendszer korábbi kiadásaihoz képest elsősorban továbbfejlesztett osztálykönyvtára tekintetében tér el, amely immár a Vista Aero felülete által nyújtott bővítések kihasználására is képes. Ezen kívül a VCL webes komponensei is jelentős fejlesztéseken estek át, amelynek köszönhetően már a dobozból kivett változattal is lehetőség nyílik az AJAX technológiát használó webes alkalmazások készítésére. Ez utóbbit támogatja a Delphi másik új kiadása, a Delphi for PHP is, amely a gyors alkalmazásfejlesztés eszközeit hozza el az eddig azt nélkülözni kényszerült PHP platformra. Az új rendszer az eredeti Delphi - hez hasonló komponens-alapú fejlesztőkörnyezetet biztosít a programozók számára, akik alkalmazásaikat a VCL PHP-hez igazított - egyébként nyílt forrású - változatának alapelemeiből építhetik fel. Utóbbi az adatbázisok kezelését egyszerűsítő komponenseket is kínál, amelyek többek között MySQL, Oracle, Microsoft SQL Server és InterBase adatbázisok elérését teszik lehetővé.

A projekt állományok felépítése

Ha új alkalmazást szeretnénk készíteni Delphiben, a File menü New Application menüpontjával tehetjük meg. Ekkor a Delphi létrehoz egy új űrlapot, és az ehhez tartozó programegységet, valamint egy projekt állományt. A megjelent Form1 űrlap mögött voltaképp egy, már megírt program van. Ahogy korábban már láttuk, a formot egy unit (egység) írja le. Egy Delphi program tehát a főprogramon kívül legalább egy unitot is tartalmaz. A unitok zárt, önálló modulok, adott céllal. Írhatunk ilyet mi is, de a Delphinek - mint a Tubo Pascalnak is - vannak olyan saját belső egységei, amelyekben a Delphi eljárásait, függvényeit, objektumait stb. helyezték el. Az általunk írt unitok vagy formhoz kötöttek, vagy nem. Elsősorban nagyobb projektek esetén célszerű a logikailag, működés szempontjából együvé tartozó programrészeket egy-egységbe összevonni. Így - túl azon, hogy a programunk áttekinthetőbb - a fordítás is sokkal gyorsabb lesz.

Az ablak forráskódja (.pas): Ezt F12 funkcióbillentyűvel hívható elő

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

implementation

{$R *.dfm}

end.

A unit unit1; a modulunk nevét adja meg. Ezt követően észrevehetjük, hogy a unit két részre van bontva. Az első része az interface kulcsszóval kezdődik (csatlakozási vagy publikus felület), a második az implementation (kivitelezési vagy implementációs rész). Az interface részben fel vannak sorolva azok a típusok, változók, melyeket a unitban használunk, és amelyeket szeretnénk hogy más unitból, programból is elérhet9k legyenek, ha ott használjuk a mi unit- unkkat (ha a másik programban megadjuk a uses unit1; sort). Az implementation részben egyrészt a feljebb felsorolt eljárások, függvények megvalósítását írjuk le – tehát azt, mit is tegyen az adott eljárás vagy függvény. Másrészt ide írhatjuk azokat a további változókat, eljárásokat, függvényeket is, melyeket csak a mi unit- unkon belül szeretnénk használni. Nézzük meg részletesebben, mi van a programunk interface részében. A uses parancs után fel vannak sorolva azok a modulok, melyek szükségesek a mi modulunk futtatásához. A type parancs után a TForm1 típusú osztály definícióját látjuk. Ez valójában a mi főablakunknak a típusa. Láthatjuk, hogy a TForm típusú osztályból lett létrehozva. (Osztály = olyan adattípus, melyet valamiféle sablonnak képzelhetünk el bizonyos objektumok – például ablak – létrehozásához. Az osztály tartalmazhat adatokat, eljárásokat és függvényeket. A Delphi - ben szokás az osztályok neveit mindig T betűvel kezdeni.) Továbbá észrevehetjük, hogy a TForm1 tartalmaz egy nyomógombot (Button1) és egy címkét (Label1), stb. A var kulcsszó után egyetlen változó van deklarálva, jelen esetben ez a Form1 objektum, ami valójában a mi alkalmazásunk főablaka. Az implementation részben találunk egy {$R *.dfm} sort. A $R egy külső resource fájl beolvasását jelzi. A *.dfm most nem azt jelzi, hogy az összes .dfm végződésű állományt olvassa be, hanem itt a * csak a mi unitunk nevét helyettesíti, tehát csak a unit1.dfm állomány beolvasására kerül sor. Ez a fájl tartalmazza a főablakunk és a rajta található komponensek beállításait.

A főprogram kódja:

```
program Project1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.res}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

A projektállomány 3 fő részből áll:

***Programfej*:** Teljesen azonos a Pascalban lévővel

Hivatkozási rész: Tartalmazza az alkalmazás és beépített egységek neveit. A {$R .RES} egy fordítási direktíva, mely a szerkesztőnek szól, hogy a RES kiterjesztésű állományokat szerkessze be a futtatható állományba.

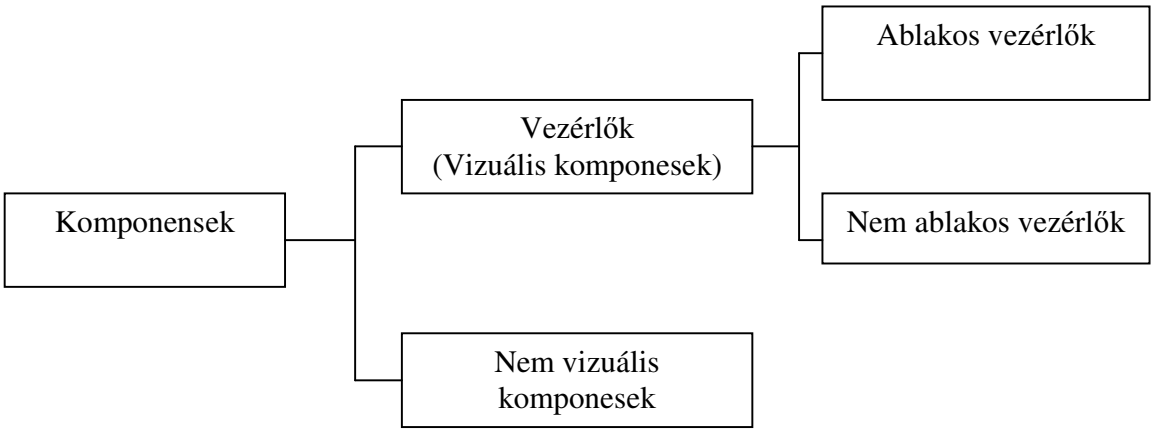
***Végrehajtható rész*:** Mint az objektumorientált alkalmazások főprogramja. Inicializálja az alkalmazást, futtatja, majd befejezi. Az Application a TApplication osztály egy példánya. Minden windowsos alkalmazás főprogramjának tárolnia kell bizonyos információkat (futtatható állomány neve, ikonja stb.), tartalmaznia kell egy inicializációs részt, valamint egy üzenetkezelő ciklust. A TApplication osztály ere van felkészítve, valamint metódusaival pedig az alkalmazást inicializálja, futtatja, majd befejezi.

A vizuális komponenskönyvtár (VCL)

A Delphi 5 –öt megelőzően az osztálykövtár neve VCL (Visual Componenent Libraly, vizuális komponensek könyvtára) volt. Ez a komponenskönyvtár a Windows programozási felületekhez kapcsolódott. A Delphi Linux változatával, a Kylixszal azonban megjelent egy új komponens könyvtár, melynek neve CLX (Componenet Libraly for Cross Platform, különféle rendszerekhez használható komponenskönyvtár). A vizuális komponensek esetében a két osztálykönyvtár különböző megoldásokat kínál a két rendszer számára, a legfontosabb osztályok, illetve a könyvtáraknak az adatbázis kezeléshez és az Internethez kapcsolódó részei szinte teljesen azonosak.

1. A VCL hagyományos részei:

A Delphi programozók, a Borland dokumentációjában javasolt nevekkel hivatkozhatunk a VCL különböző részeire, így ezek a nevek lassan a különféle komponenscsoportok azonosítóivá váltak. A komponensek a TComponent osztályból származnak, amelyek az osztályhierarchiájával legfelső elemeinek egyike. A TComponent osztály a TPresisten osztályból öröklődik. A könyvtár a komponensek mellett olyan osztályokat is tartalmaznak, amelyek közvetlenül a TObject, illetve a TPresisten osztályból származnak. Ezek a nem komponens osztályoktól származnak. Ezeket a nem komponens osztályokat a dokumentáció együttesen objektuoknak nevezi



Nem vizuális komponesek

Minden kompones, ami nem vezérlő, minden olyan osztály, amely a TComponet leszármazottja, de ősei között nem szerepel a TControl osztály, Tervezési időben a nem vizuális komponesek a formon ikonok formájában (esetleg feliratozva) jelennek meg. Néhány kompones, futásidőben is megjeleníthető marad (például a szabványos párbeszéd ablakok), de a többi azonban láthatatlan marad (adatbázis táblák komponesei).

Az adatbázis-kezelés és a Delphi

1. A Borland Delhi fejlesztőeszközzel nagyon gyorsan és hatékonyan tudunk adatbázis-kezelőalkalmazásokat fejleszteni. A Delphi kiemelkedően támogatja az adatok kezelését, és mivel RAD eszközről van szó, nagyon gyorsan fejleszthetjük ki az alkalmazás párbeszédpaneleit is. Az adatszolgáltatások rétege (Data Processing). Ez a réteg felelős az adatok fizikai eléréséért, feldolgozásáért. E réteg feladata az adatbázis állományok nyitása, zárása, újadat felvitele, törlése, módosítása, indexek kezelése, zárolási konfliktushelyzetek feloldása, stb.

2. Az alkalmazáslogika rétege (Bussines Logic). Az alkalmazáslogika rétege az adatbázisra vonatkozó szabályok összességét tartalmazza. Gyakorlatilag ebbe a rétegbe tartoznak azok a funkciók, műveletek, amelyek meghatározzák egy adatbázis működését. Ilyen szabályok a mező illetve rekordszintű ellenőrzések (mezőszintű ellenőrzés pl. ha egy tanuló érdemjegyeinek felvitelekor a program csak egy és öt közötti értéket enged felvinni), a hivatkozási függőségek ellenőrzése (pl. egy könyvet csak akkor lehessen eladni ha az szerepel a könyvesbolt árukészletén) stb.

3. Megjelenítési réteg (User Interface)

Ezek után nézzük meg milyen adatbázis architektúrák, vannak, és azokban az előbb tárgyalt rétegek hogyan vannak implementálva.

Adatbázis architektúrák

- Egygépes megvalósítás (Local Databases). Az adatbázisoknak ez a lehető legegyszerűbb megvalósítási módja. Az alkalmazás egyetlen gépre íródott, az adatbázis és az azt feldolgozó program ugyanazon a gépen helyezkedik el, az adatbázist csak egyetlen program használja egy időben. Ebben az esetben mindhárom réteg egyazon gépen helyezkedik el.

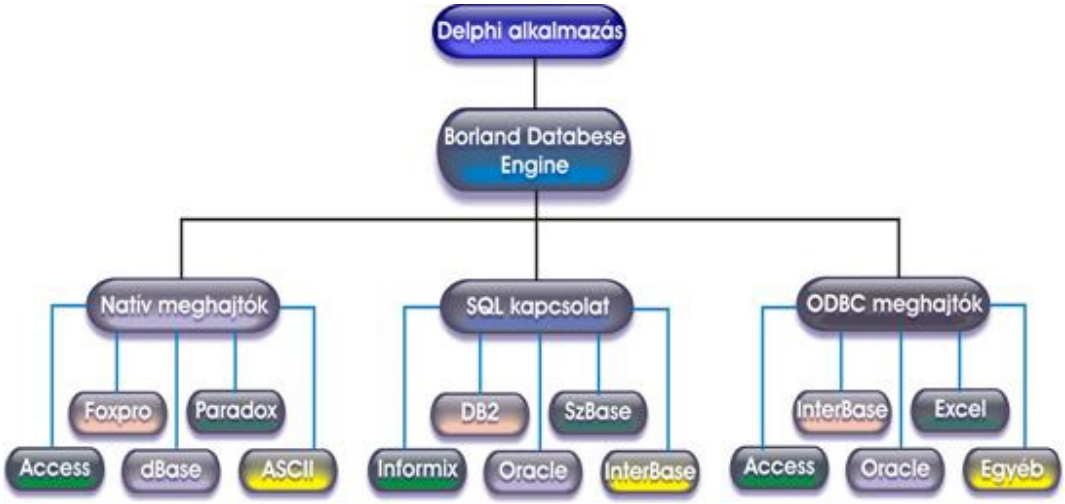
- File - kiszolgáló (File - Server) architektúra. Ebben az esetben az adatbázis állományok átkerülnek egy központi szerverre és egy időben több program is használhatja őket hálózaton keresztül. A szerver csak az adatok tárolására szolgál. Ezen megoldás esetén, ha a felhasználó akármilyen egyszerű adatműveletet akar is végrehajtani, az adatrekordoknak el kellett jutniuk a felhasználóhoz a hálózaton. Ez nagy adatforgalommal jár, ami a hálózat túlterheléséhez vezethet. Ezt a megoldást leginkább az xBase alapú (Dbase, FoxPro, Clipper stb.) adatbázis - kezelőkkel használják. Delphiben is írhatunk ilyen alkalmazásokat, de csak akkor érdemes, ha nincs szükség nagy teljesítményre, aránylag kevés felhasználója van a programnak, és olcsón meg akarjuk úszni a dolgot, (mivel egy adatbázisszerver nem olcsó mulatság). Szintén mindhárom fentebb tárgyalt réteg egyazon gépen helyezkedik el.

- Ügyfél-kiszolgáló (Client / Server) Architektúra. Az adatbázisok implementálásának e formájában az alkalmazás két részre bomlik. Az adatok közvetlen kezeléséért egy adatbázis-szervernek nevezett software a felelős, (pl. MsSql Server, Oracle, Informix, Sybase, InterBase stb.), míg a felhasználóval való kapcsolattartás az ügyfél program feladata. Az adatbázis-servert készen vásárolhatjuk meg, míg a kliens programot mi magunk írhatjuk meg valamilyen programozási nyelven, Az alkalmazás logikának egy részét magába az adatbázisba, a többit a kliens programba tudjuk beépíteni. Hogy ez hogyan is történik arról majd később lesz szó. A Client / Server technológiában az ügyfél utasítja a szervert, pl. adatokat, kér le, és erre a szerver visszaküldi az eredményt. Tehát nem kell a hálózaton a feldolgozandó adatoknak rekordról-rekordra átmenni a klienshez,

hanem egy rövid parancs hatására, csak a ténylegesen kért, hasznos adatok fognak a szervertől a kliensig utazni, ezáltal jelentősen csökkentve a hálózati forgalmat. Igy az adatfeldolgozást a szerver végzi a kliens parancsainak hatására. E parancsok számára kidolgoztak egy szabványos nyelvet, ez az SQL. Tehát az adatbázis-szervereket ilyen SQL parancsokkal tudjuk munkára bírni. Az adatbázis szervereknek a hálózati forgalom csökkentésén kívül számos más előnyük is van, biztosítják az egyidejű adatelérést (egyszerre nagyszámú felhasználó kiszolgálására képesek), az adatbiztonságot, központilag kezeli a felhasználói jogosultságokat, stb.

- Több rétegű (Multi-Tier) adatbázis architektúra. Ebben az esetben a kliens nem közvetlenül az adatbázis-szerverhez, hanem egy vagy több köztes ún. applikációs szerverhez kapcsolódik, és végül az applikációs szerver kapcsolódik az adatbázis-szerverhez. Tehát a kliensnek a középen elhelyezkedő applikációs szervertől kapják az adatokat, ezért ezt a réteget adatszolgáltatónak (Data Broker) is nevezik. Így az adatbázis logikát el lehet helyezni a középső rétegben, és a kliens feladata csak a felhasználóval való kapcsolattartás lesz. Az ilyen kliens-t "sovány" (thin) kliensnek nevezzük, hiszen a munka nagy részét az applikációs szerver végzi. Tehát ebben az esetben a három réteg, fizikailag is három különböző helyen helyezkedhet el.

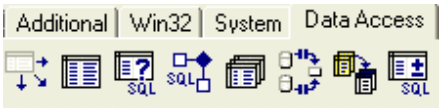
Delphiben lehetőségünk van a fentebb említett bármelyik architektúrát felhasználva adatbázisos alkalmazást készíteni. Így használja egy Delphi alkalmazás a BDE - t



BDE Aliasok (Álnevek)

A BDE álneveket (alias) használ a különböző adatbázisokra való hivatkozáskor. Az alias gyakorlatilag paraméterek halmaz, ami egyszerűbb esetben, lokális adatbázisoknál az adatbázis elhelyezkedését és típusát tartalmazza, adatbázisszerverek esetén pedig egy csomó plusz paraméter megadható, pl. a megnyitás módja, a szervernév, felhasználónév stb. Amikor elkészítjük az alkalmazásunkat, akkor abban alias- sal hivatkozunk a használt adatbázisra. Így ha később pl., megváltozik az adatok elérési útvonala, az nincs fixen belefordítva a programunkba, hanem egyszerűen megváltoztathatjuk azt a későbbiek során bármikor. Alias-t a BDE Administrator-ral, a Database Explorer-el, (A BDE Administrator az adatbázismotor konfigurációs programja, mellyel aliasokat hozhatunk létre, módosíthatunk, vagy törölhetünk. A Database Explorer egy segédprogram mellyel aliasokat kezelhetünk, adatbázisokat nézhetünk meg, módosíthatunk, SQL lekérdezéseket futtathatunk stb.) de akár saját magunk programból is létrehozhatunk. A létrehozott, alias a BDE saját konfigurációs állományában (IDAPI32. CFG) kerül elmentésre, és mindaddig megmarad, míg nem töröljük. Miután elindítottuk a BDE Administratort, már alapesetben is látható lesz egy pár, alias, ettől nem kell megijedni, ezek példák melyeket a Delphi hoz létre, amikor felinstalláljuk és a saját példa adatbázisaira, mutatnak.

Adatelérési (Data Access) komponensek



Adatelérési (Data Access) komponensek az adattáblák elérését teszik lehetővé. Tulajdonképpen a BDE megfelelő moduljaival ezek tartják a kapcsolatot. A Delphi változatától függ, hogy itt milyen elemeket láthatunk, a Standard változatban az alábbi palettával találhatjuk szemben magunkat:

Nézzük meg, hogy mire is szolgálnak az itt látható, előre elkészített komponensek, amelyek többsége nem vizuális komponens, vagyis csak tervezési időben látszanak.

Table

A Table komponens egy relációs táblával tartja a kapcsolatot a BDE - n keresztül. A paramétereit rendszerint tervezési időben határozzuk meg, néhány tulajdonságának kivételével.

DataSource

Az adatbázisok eléréséhez szükségünk van egy DataSource komponensre, amely erre szolgál. Fontos, hogy ez nem közvetlenül kapcsolódik a fizikai táblához, mivel ennek bemenete lehet akár egy tábla, akár egy lekérdezés eredménye, de akár tárolt eljárások is szolgáltathatják a bemenetét. Az adatmegjelenítési komponensek ettől kapják az adatokat. Ez a megoldás rendkívül rugalmas és hatékony eszközt ad a kezünkbe, mert így az adatok és a megjelenítési komponensek függetlenné válhatnak egymástól. Például, amikor egy táblából lekérdezéseket kell megjeleníteni a képernyőn attól függően, hogy a felhasználó mit választ ki. Borzasztó hosszadalmas lenne minden megjelenítési komponensnek megváltoztatni az adatforrását, ehelyett inkább a DataSource komponenst irányítjuk át egy másik lekérdezésre és máris a helyes eredményt, látjuk.

Query

A Query komponenssel a relációs táblákból kérdezhetünk le rekordokat az SQL nyelv használatával. Szintén tervezési időben állítjuk be a legtöbb tulajdonságát, de az SQL utasításokat az esetek döntő részében csak futási időben adjuk át.

StoredProc

A StoredProc szerver-kliens adatbázis-szerkezetnél használatos, amikor a kliens az adatbázis-szerveren eljárásokat szeretne tárolni, illetve azokhoz hozzáférni.

Database

A Database komponens rendszerint szerver-kliens architektúráknál használatos, lehetővé teszi a kapcsolatok ellenőrzését, különböző biztonsági műveletek elvégzését csakúgy, mint a kapcsolat-ellenőrzést.

Session

A Session komponens legfontosabb tulajdonsága, hogy egy eseményt biztosít az adatbázisokban történő bejelentkezések testre szabásához.

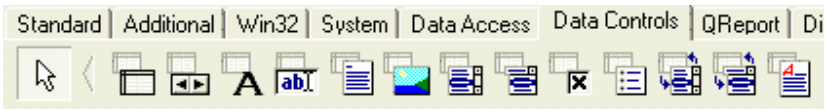
BatchMove

A BatchMove segítségével, mint ahogy a nevéből is látszik, kötegelt műveletek végrehajtásra nyílik lehetőségünk. Ilyen művelet lehet akár másolás, akár mozgatás, vagy törlés is.

UpdateSQL

Az UpdateSQL segítségével adatfrissítő műveleteket végezhetünk SQL lekérdezések segítségével egy csak olvasható Query komponensen. Rendszerint a táblák és a lekérdezések UpdateObject értékeként használjuk.

Adatmegjelenítési komponensek



Az adatok párbeszédpanelen (FORM) megjelenítésére számos komponens áll rendelkezésünkre, amelyek rengeteg előnyös tulajdonsággal rendelkeznek. Ha ezek nem lennének megfelelő, az interneten számtalan komponenshez férhetünk hozzá, amelyek egy része ingyenes, míg másokat meg kell vásárolnunk. Amikor megnyitjuk a Delphi Standard Data Controls (adat vezérlő) palettáját, akkor ehhez hasonló választékkal találjuk szemben magunkat:

DBGrid

A DBGrid egy táblázatot jelenít meg a panelen, amelyen a táblákból származó adatokat láthatjuk. A táblázat automatikusan annyi oszlopot tartalmaz, amennyi mezője van a relációnak, de ezt felül is bírálhatjuk. Áttekintő listák készítéséhez nagyon jól használható.

 **DBNavigator**

A rekordok műveleteket segíti elő a DBNavigator. Segítségével a rekordmutató léptetésén túl, felvehetünk új rekordot, törölhetjük az aktuális sort, vagy akár szerkeszthetjük is azt. A gombok tetszőlegesen ki és bekapcsolhatók, annak megfelelően, hogy melyikre van szükségünk.

 **DBLabel**

Abban az esetben, ha egy statikus, vagyis a formon nem módosítható szöveget kell megjeleníteni egy relációs táblából, akkor használjuk a DBLabel komponenst. Az éppen aktuális rekord hozzárendelt mezőjének az értékét jeleníti meg.

 **DBEdit**

Az egyik leggyakrabban használt vezérlőelem a beviteli mező adatbázisokhoz illesztett változata, a DBEdit. Nem csak a mező értékét képes megjeleníteni, hanem módosíthatjuk is az, emennyiben engedélyeztük ezt az adatbázis műveletet.

 **DBMemo**

Számos esetben előfordulhat, hogy nem elegendő egy soros beviteli mező, ilyen esetekben használhatjuk a DBMemo komponenst, amely a DBEdit többsoros változata.

 **DBImage**

Ha képet szeretnénk megjeleníteni a párbeszédpanelünkön, akkor erre a DBImage komponenst felhasználva nagyon egyszerűen lehetőséget kapunk. A képek megjelenítésénél még azt is meghatározhatjuk, hogy a nagy (vagy éppen kicsi) képekkel mi történjen, megnyújthatjuk, kicsinyíthetjük, vagy éppen levághatjuk a kilógó részeket.

 **DBListBox**

A DBListBox komponenshez nagyon hasonló vezérlőelem a DBComboBox, azonban itt a választható elemeket egy legördülő listából választhatjuk ki.

 **DBCheckBox**

Abban az esetben, ha egy mező értéke csak igaz, vagy hamis lehet, akkor használhatjuk a DBCheckBox vezérlőelemet. A logikai érték kerül eltárolásra az adattábla megfelelő mezőjében.

 **DBRadioGroup**

Egy adatbázishoz kapcsolódó választógomb-csoportot hoz létre a DBRadioGroup komponens. A gombok egy listában tárolódnak és a kiválasztott elem sorszáma kerül be a tábla mezőjébe.

 **DBLookupListBox**

Számos esetben előfordul, hogy egy listaelem sorait egy másik táblából kellene feltölteni. Ennek biztosítására készült a DBLookupListBox komponens, amely rendkívül jól használható több táblás adatbázisok esetében.

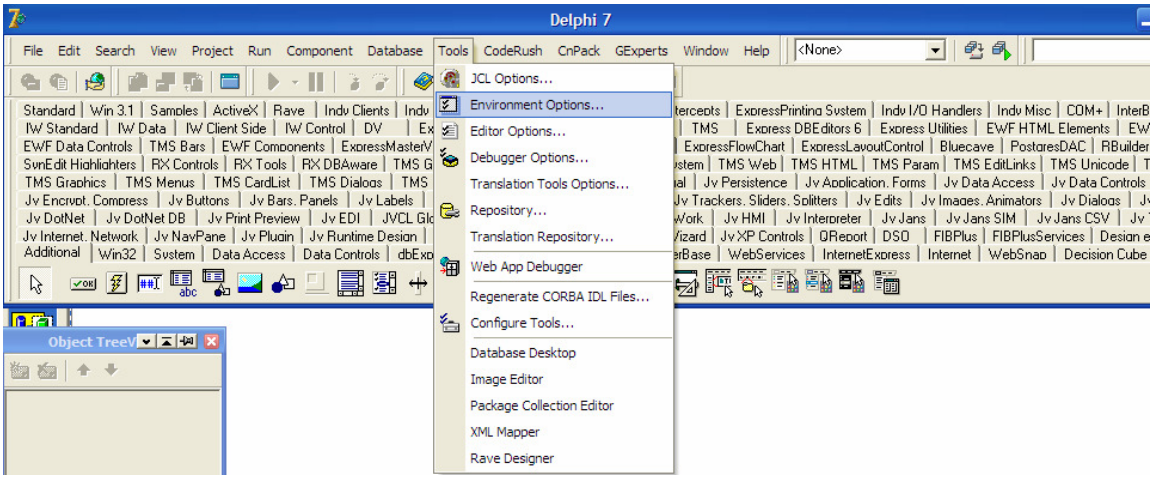
 **DBRichEdit**

A DBRichEdit komponens nagyon hasonlít a DBMemo vezérlőelemhez, mivel itt is több soros információt tárolhatunk, azonban az ebben megjelenített tartalomhoz formátumot is hozzárendelhetünk. Lehetőség van szövegek stílusát, betűtípusát, betűméretét megváltoztatni.

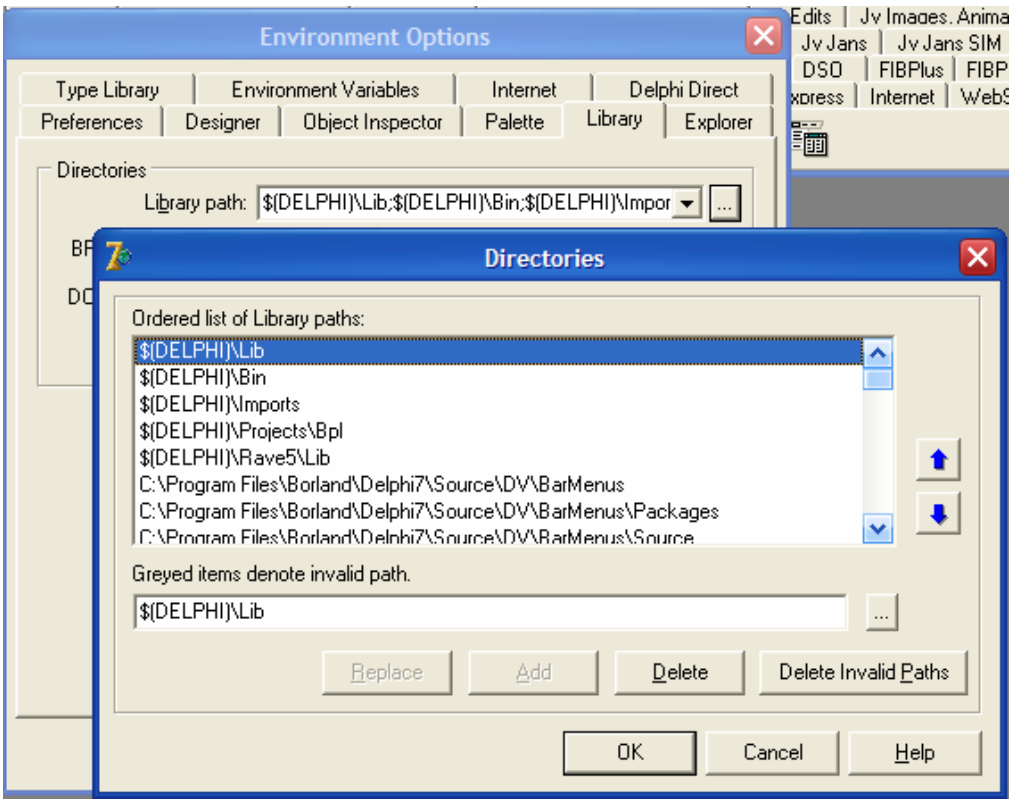
Fejlesztés során használt nem beépített komponensek:

VCLSKIN

Telepítése nagyon egyszerű, mert szerencsés helyzetben vagyunk, mert vagy a cég mellékel hozzá egy setup.exe –t és ő mindent elintéz helyettük, mint most vagy manuálisan kell egy komponest feltelepíteni ez sem nehéz, de egy kicsit körülményesebb, mert kézzel kell beállítani az elérési útvonalakat. Meg próbálom levezetni ezt a megoldást is mert későbbi fejlesztés során bárki belefuthat a componens telepítési problémákba (Tools -> Environment Options..).

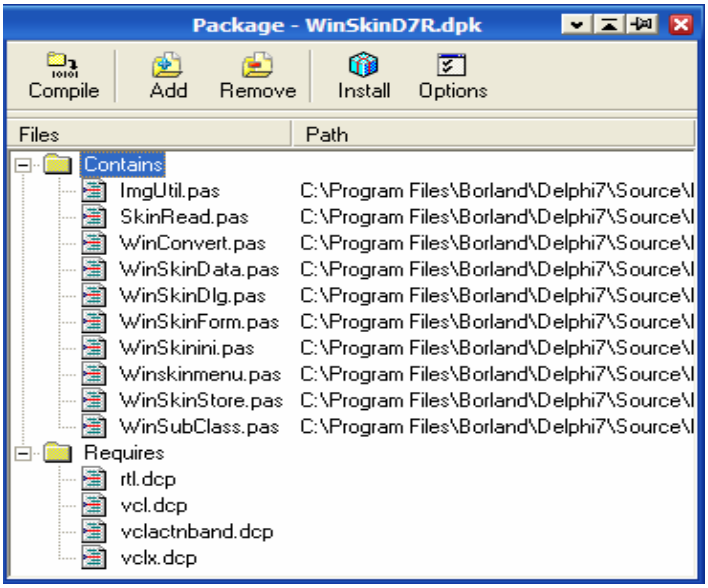


Majd kiválasztjuk a Library fület és a Libraly path mellett lévő speed butonra klikelve megkeressük az elérési utat, és az add butonra klikelünk.

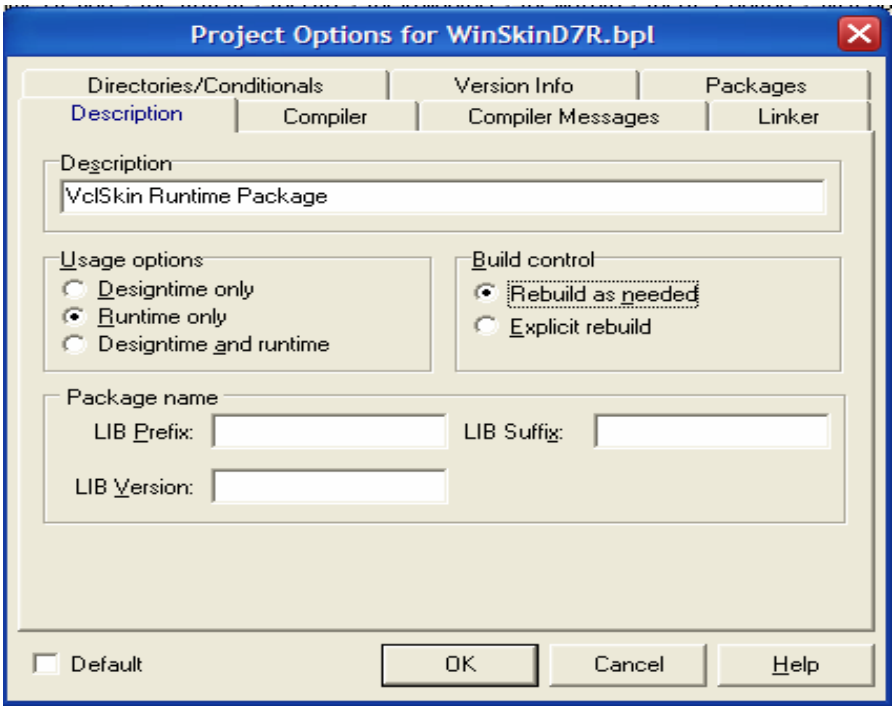


Ha eddig eljutottunk akkor ezek után File menüpont Close All menüjére klikeljünk rá (ez csak akkor szükséges, ha van megnyitott project állományuk). Majd válasszuk ki a File menü open

menüpontját és keressük meg a feltelepítendő componenst, „*megjegyzésként mondom, hogyha több komponenst használunk egy fejlesztés során, akkor azokat érdemes egy könyvtárba másolni a könnyebb kezelhetőség miatt én mindig a Delphi könyvtáron belül a Source mappába szoktam másolni*”, ha ez sikeres volt, akkor a következő képet kell, hogy lássuk.



Itt célszerű az Options gombra klikelni és a Rebuild as needed rádió gombot kell aktivá tenni

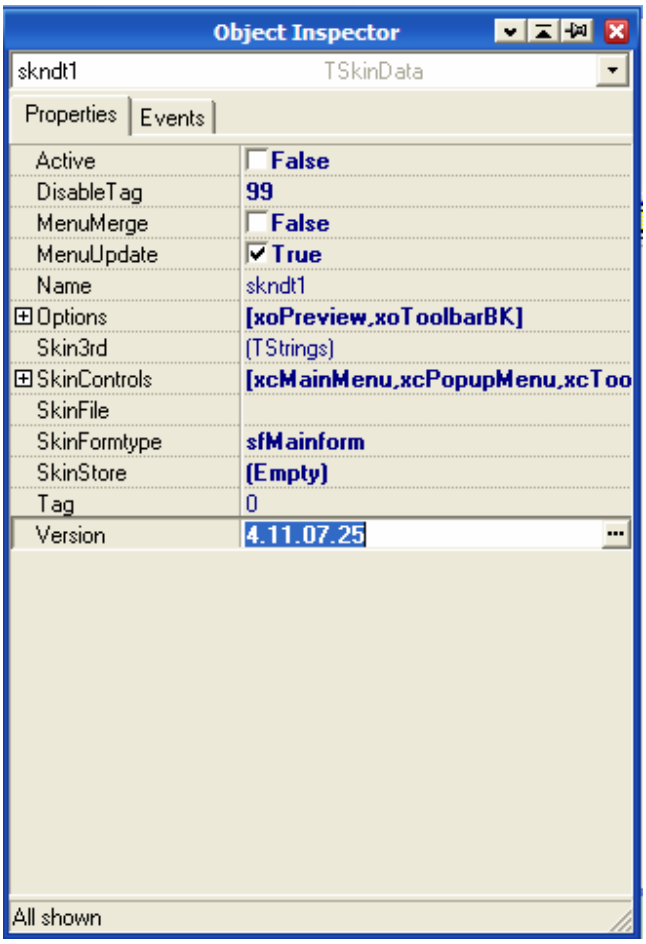


Utána Ok gomb kiválasztása, Compile és Install és használhatjuk a komponensünket.

VCLSKIN igazából csak egy Skint húz a formokra melynek csak annyi a jelentősége, hogy egy sokkal szebb felületet nyerünk és ez által tetszetősebb lesz az alkalmazásunk. Használata igen egyszerű kiválasztjuk a komponens palettán és rárakjuk a formunkra.

SkinData, SkinCaption

Feladata a kiválasztott Skin és a form - hoz tartozó caption tárolása tárolása. Minden feltelepített komponensnek vannak tulajdonságai *„Properties”*, eseményei *„Events”* és ezeket tetszőleges konfigurálhatjuk igényeink szerint az Object InsPector segítségével ezt az F11 gyorsbillentyű segítségével, érhetjük el.



6. Az SQL

Az SQL a strukturált lekérdező nyelv (Structured Query Language) rövidítése, melyet az IBM dolgozott ki a DB2 relációs adatbázis kezelőjéhez. Ma már a relációs adatbázis kezelők szabványosított nyelve, bár több dialektusa, bővítése alakult ki.

Az SQL szerepe, tulajdonságai

Az SQL egy szabványosított lekérdező nyelv, melyet több relációs adatbázis kezelő ismer, különböző operációs rendszeri környezetben. Ennek óriási jelentősége van az adatbázis alkalmazások fejlesztőinek körében, mert így az alkalmazások a különböző operációs rendszerek és adatbázis kezelők között módosítás nélkül vagy csekély módosítással átvihetők.

Az SQL nem algoritmikus nyelv, nem tartalmaz algoritmus szerkezeteket (elágazás, ciklus stb.). Az SQL halmaz orientált nyelv, mely a relációkon dolgozik. A halmaz orientáltság azt jelenti, hogy nem kell definiálni a művelet végrehajtásának lépéseit, hanem a feladat nem eljárásszerű megfogalmazását kell megadni, melyek a reláció vagy relációk kiválasztott sorain hajtódnak végre. A művelet végrehajtásához optimális megoldás megtalálása a nyelvi processzor feladata, nem a programozóé. Például annak eldöntése, hogy egy adott visszakeresésben alkalmazhatók-e indexek, vannak-e indexek vagy építsen-e fel új indexet, a nyelvi processzor feladata. Az SQL nem rekurzív nyelv.

Az SQL nyelvnek két felhasználási lehetősége van:

- önálló SQL, vagy 4. generációs eszközbe építve
- beágyazott SQL

Az SQL nyelv önálló felhasználása esetén csak a nyelv utasításai állnak rendelkezésre. Ennek alkalmazására főként akkor kerülhet sor, ha nincs megfelelő alkalmazás az adott feladat

elvégzésére, illetve az alkalmazások fejlesztői használják a negyedik generációs nyelvekbe építve. Ilyen eszközök a jelentéskészítő, az űrlapkészítő vagy menükészítő lehet.

A beágyazott SQL esetén egy harmadik generációs algoritmikus nyelvbe (C, PL/SQL, Pascal FORTRAN stb.) ágyazva alkalmazzuk az SQL nyelv elemeit. Ebben az esetben az algoritmikus feladatokat a harmadik generációs nyelvre, az adatbázissal kapcsolatos műveleteket pedig az SQL-re bízhatjuk.

SQL parancsok csoportosítása:

• ***DDL*** (adatdefiníciós parancsok): az adatbázis tábláit definiálhatjuk, törölhetjük, módosíthatjuk velük (CREATE TABLE; ALTER TABLE,; DROP TABLE, CREATE VIEW stb.)

• ***DML*** (adatmanipulációs parancsok): Az adatok módosítására használhatóak, beszúrás, módosítás, törlés (INSERT, UPDATE, DELETE)

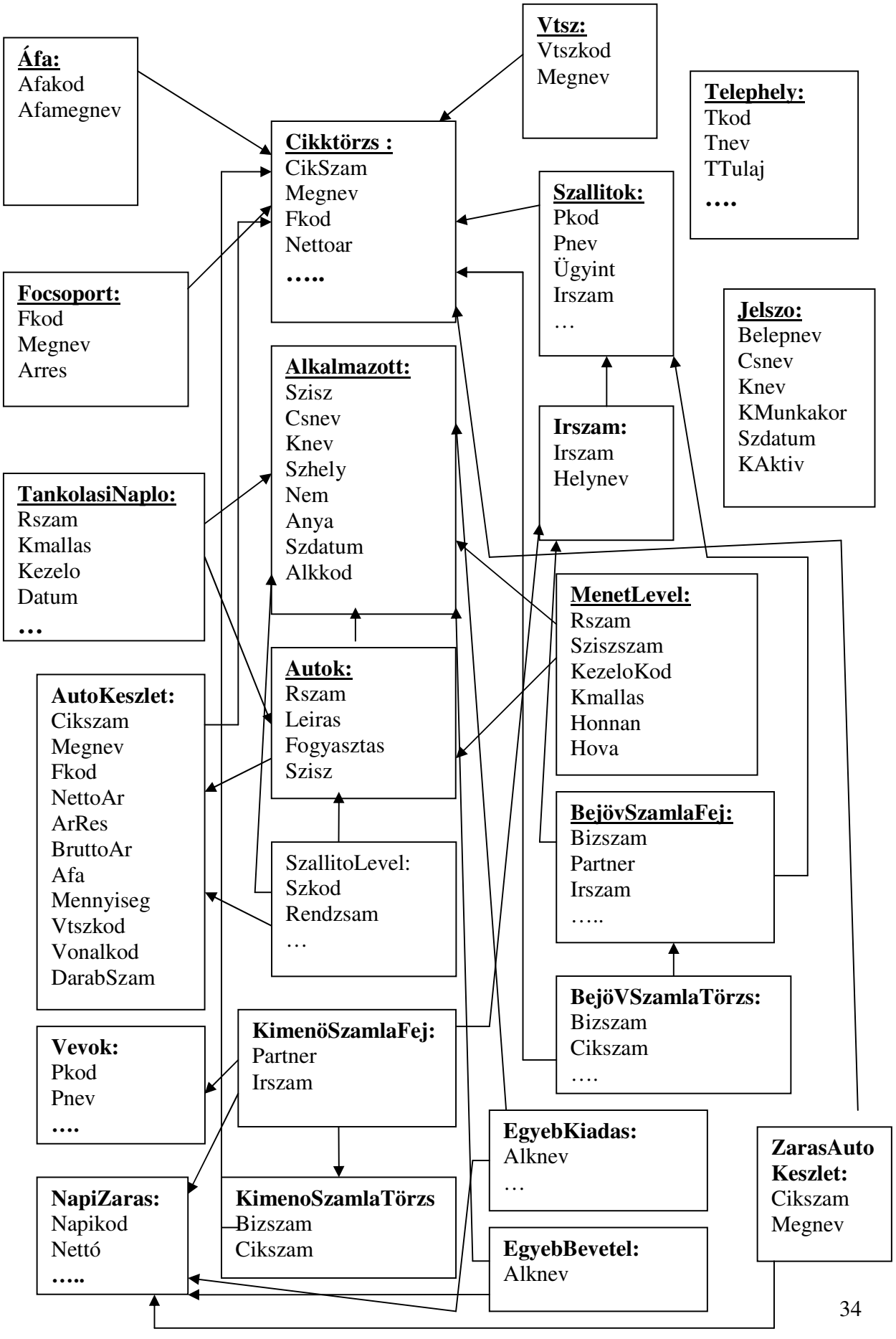
• ***DCL*** (adatvezérlő parancsok): Az adatbázis-kezelésével kapcsolatos feladatok irányíthatóak vele, mint például jogosultság kezelés, tranzakció-kezelés (COMMIT, ROLLBACK, GRANT, REVOKE).

• *SELECT*: A **lekérdező nyelv** egyetlen utasításból áll, mely számos alparancsot tartalmazhat, és a lekérdező utasítások többszörös mélységben egymásba ágyazhatók.. Néhány megközelítésben a DML csoportba tartozik, egy szűrést végez az adott táblán, ennek eredményét adja vissza, azonban a tábla tartalmára vonatkozóan semmit nem változtat.

7. A program bemutatása

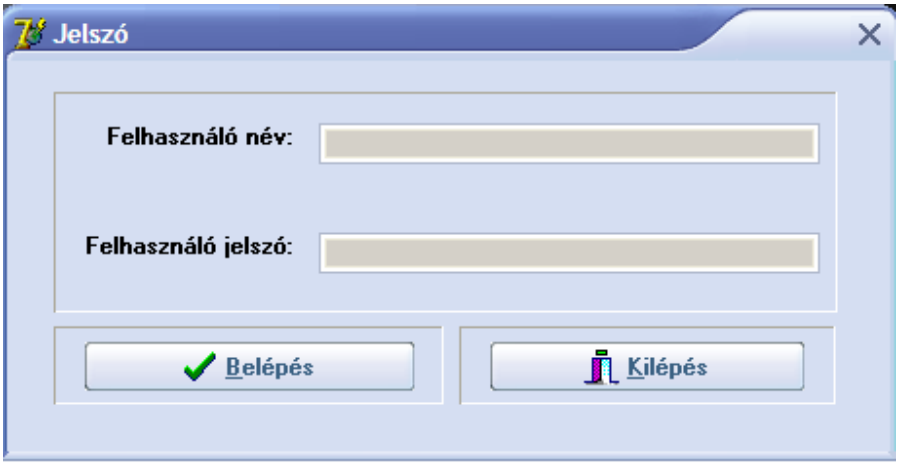
Adatmodell

Az adatbázis alapú alkalmazások fejlesztésének legkényesebb pontja, egy jól megtervezett és átgondolt adatmodellre sokkal egyszerűbb felületet húzni, mint egy olyanra, amelynél a fejlesztés során jön rá az ember, hogy nem logikus, vagy egyáltalán nem jó az. Mivel én egy fuvarozási kisvállalkozás raktárkészlet nyilvántartó rendszerét készítettem el, és a cég jelen esetben műanyag termékek forgalmazásával és reklámozásával foglalkozik. Ezért a program főbb feladata a központi raktárkészlet pontos nyilvántartása és főcsoportokba történő bekategorizálása (főcsoportok alatt jelenesetbe pl. csomagoló termékek, műanyag áruk, táskák), vevők, szállítók főbb adatainak tárolása. A cég saját maga végzi az árúk terítését a környező településeken ezért fontos volt még a gépkocsik készletének kimutatás, és fuvarozók napvégi elszámoltatása. Az adatbázis tervezésekor a dBase IV választottam a beépített adatbázis kezelők közül, viszonylag bővebb a típuskínálata a választhatóak közül. Terezés során a következő táblákat valósítottam meg mely a következő ábrán jól látható.


Áfa:
Afakod
Afamegnev
Vtsz:
Vtszkod
Megnev
Telephely:
Tkod
Tnev
TTulaj
….
Cikktörzs :
CikSzam
Megnev
Fkod
Nettoar
…..
Szallitok:
Pkod
Pnev
Ügyint
Irszam
…
Focsoport:
Fkod
Megnev
Arres
Jelszo:
Belepnev
Csnev
Knev
KMunkakor
Szdatum
KAktiv
Alkalmazott:
Szisz
Csnev
Knev
Szhely
Nem
Anya
Szdatum
Alkkod
Irszam:
Irszam
Helynev
TankolasiNaplo:
Rszam
Kmallas
Kezelo
Datum
…
MenetLevel:
Rszam
Szisszam
KezeloKod
Kmallas
Honnan
Hova
Autok:
Rszam
Leiras
Fogyasztas
Szisz
AutoKeszlet:
Cikszam
Megnev
Fkod
NettoAr
ArRes
BruttoAr
Afa
Mennyiseg
Vtszkod
Vonalkod
DarabSzam
BejövSzamlaFej:
Bizszam
Partner
Irszam
…..
SzallitoLevel:
Szkod
Rendzsam
…
BejöVSzamlaTörzs:
Bizszam
Cikszam
….
KimenöSzamlaFej:
Partner
Irszam
Vevok:
Pkod
Pnev
….
EgyebKiadas:
Alknev
…
ZarasAuto
Keszlet:
Cikszam
Megnev
NapiZaras:
Napikod
Nettó
…..
KimenoSzamlaTörzs
Bizszam
Cikszam
EgyebBevetel:
Alknev

A program működése:

A program indításakor legelőször egy jelszó bekérő formot látunk, ennek a jelentősége igazából abban rejlik, hogy illetéktelen személyek ne módosíthassák és láthassák fontos adatainkat. Fejlesztés során bevezettem egy **Admin** jelszót, ami jelen esetben „developer”,.



A belépés gombra kattintva megtörténik az ellenőrzés, hogy a kitöltött felhasználó név, jelszó helyesen lett – e megadva. Az ellenőrzés programkódja a következő:

Legelsőnek EditFelhasznaloChange eseménykor leszűkítjük a jelszó táblát erre a felhasználóra a QueryJelszo segítségével a következő módon

```
DataModuleQuery.QueryJelszo.Close;
DataModuleQuery.QueryJelszo.SQL.Clear;
DataModuleQuery.QueryJelszo.SQL.Add('select *from jelszo where BELEPNEV
LIKE('''+EditFelhasznalo.Text+''')');
DataModuleQuery.QueryJelszo.Open;
```

```
procedure Tjelszo_form.btnBelepClick(Sender: TObject);
Var UserName : String;
begin
UserName := 'developer';
```

//Itt történik a fentiekben említett definiált felhasználó ellenőrzése és a fejlesztő menüpont engedélyezése

```
if EditFelhasznalo.Text = UserName then
Begin
 EditFelhasznalo.Text:='';
 FormEgyebbKiadas.BitBtnEgyebKTorol.Visible := True;
 FormEgyebbBevetel.BitBtnBevTorol.Visible:= True;
 with main_form do
 Begin
  Developer1.Visible := True;
  LabelKezelo.Caption:='Programozó';
  LabelMunkakor.Caption:='developer';
  ShowModal;
 End;
End;
```

//Ellenkező esetben megvizsgáljuk, hogy az előzőleg leszűrt User jelszója megegyezik a jelszó mezőbe begépelttel és aktív – e a felhasználó státusza

```
if ((DataModuleQuery.QueryJelszo.FieldByName('KJELSZO').AsString =  EditJelszo.Text) and
(DataModuleQuery.QueryJelszo.FieldByName('KAKTIV').AsInteger = 1) ) then
 Begin
 //Letiltva
 main_form.ranzakcik1.Visible := False;
 main_form.Egybbkltsgek1.Visible := False;
 main_form.Szmlk1.Visible := False;
```

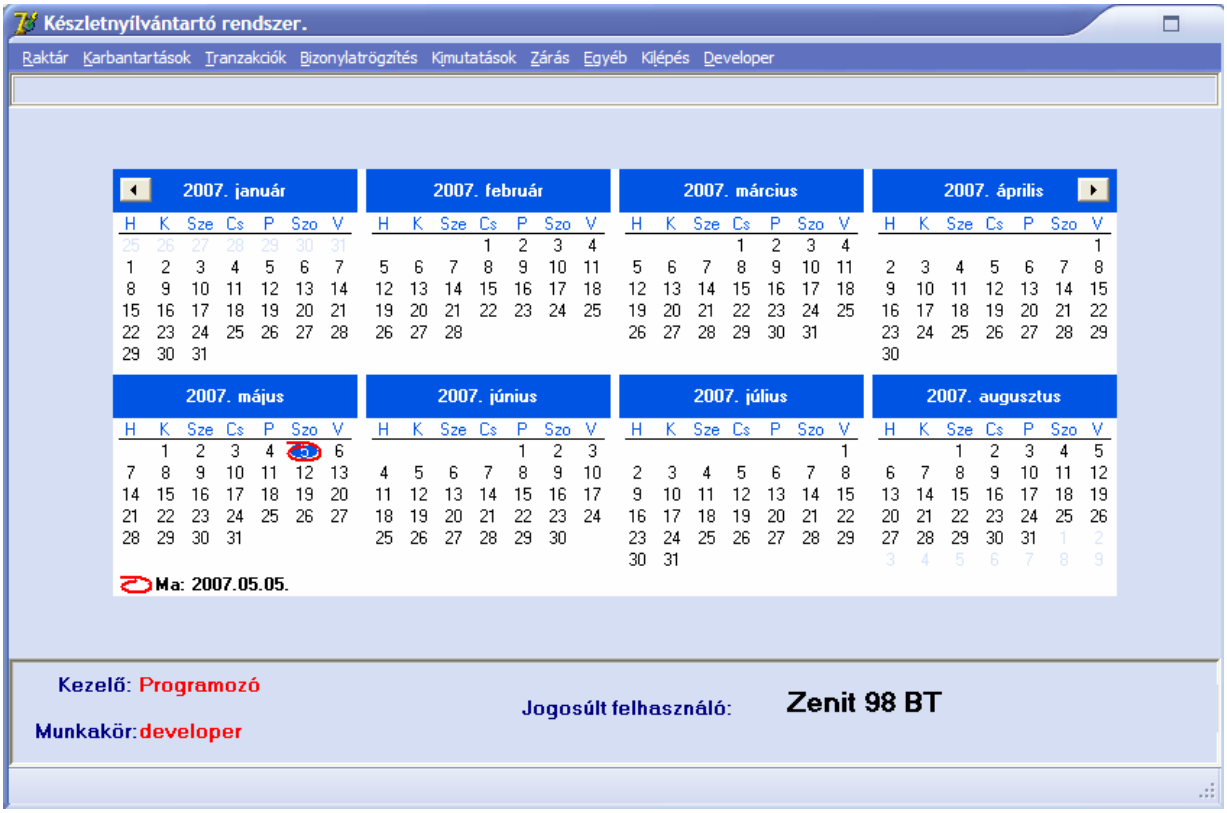
//// Ez csak legelső indításkor fordulhat elő és jelezzük a felhasználónak, hogy töltse ki a Cégre vonatkozó paramétereket, mert általában minden lista fejlécében a Cégre vonatkozó információ jelenik meg

```
 if DataModule1.TableTulajdonos.RecordCount = 0 then
  Begin
     MessageDlg('Kérem Elsőnek rögzítse a cégadatokat!!',mtWarning,[mbOk],0);
     FormCeginfo.ShowModal;
  End;
 main_form.LabelKezelo.Caption:=DataModuleQuery.QueryJelszo.FieldByNaMe('BELEPNEV').AsString;
 main_form.LabelMunkakor.Caption:=DataModuleQuery.QueryJelszo.FieldByName('KMUNKAKOR').AsString;
 main_form.Developer1.Visible := False;
 EditFelhasznalo.Text:='';
 EditJelszo.Text:='';
 FormEgyebbKiadas.BitBtnEgyebKTorol.Visible := False;
 FormEgyebbBevetel.BitBtnBevTorol.Visible:= False;
 main_form.ShowModal;
 End
```

// Egyébként figyelmeztetjük a felhasználót, hogy tévesen töltötte ki valamely mezőt.

```
else
  Begin
  MessageDlg('Sajnos a jelszót vagy a felhasználó nevet elgépelte',mtWarning,[mbOk],0);
  EditFelhasznalo.Text:='';
  EditJelszo.Text:='';
  End;
end;
```

Ha mindezen túljutottunk, akkor találkozunk a következő felülettel a main form - al ahol minden funkciót elérhetünk.



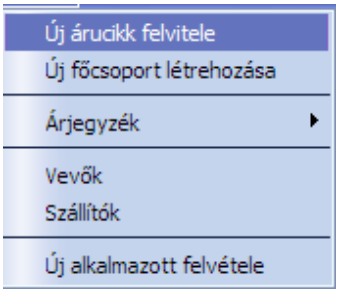
A program szerkezete a következőképpen tagolódik:

- Raktár,
- Karbantartások,
- Tranzakciók
- Bizonylatrögzítés
- Kimutatások
- Zárás
- Egyéb
- Kilépés
- Fejlesztői funkció (Developer)

Legelső a raktárkészlet menüpont melyen belül az új cikket, új főcsoportot, új alkalmazottat, vevőt és szállítót vehetünk fel az adatbázisba ide került az árjegyzékkészítés melyet, készíthetünk kifejezetten egy partner részére vagy csak általánosan. A karbantartások menüpont alatt a raktárkészlet menüpont alatt felvett adatokat törölhetjük, módosíthatjuk. Ide történt még az áfa kulcsok karbantartása, az alkalmazottak gépjárművek összerendelése, a tankolási napló, menetlevél karbantartása. A tranzakciós menüpont alatt a gépjárművek feltöltését és a szállítólevél készítését tudjuk elvégezni. A bizonylatrögzítés alatt a bejövő kimenő számlakezelés és az egyéb bevétel és kiadásokat tarthatjuk nyílván. A kimutatások menüpont alatt a vevők szállítók, bejövő kimenő számlákat, és egyéb bevétel kiadás kimutatásait tekinthetjük meg. A napi zárás alatt a napvégi összegzéseket és a gépjárművek ellenőrzéseit végezhetjük el. Az egyéb menüpont alá olyan funkciók kerültek, ami a mindennapi használatban kevésbé játszanak szerepet például cégadatok rögzítése, ami csak a rendszer indításakor kell meghatározni, a program névjegye, szövegszerkesztő és a felhasználók karbantartása. A kilépés menüpont alá a program bezárása és a felhasználó kijelentkeztetése került. A developer menüpontot csak a rendszergazda érheti el itt olyan adatok törléséhez is joga van amit a felhasználói joggal esetleg nem tehetne meg.

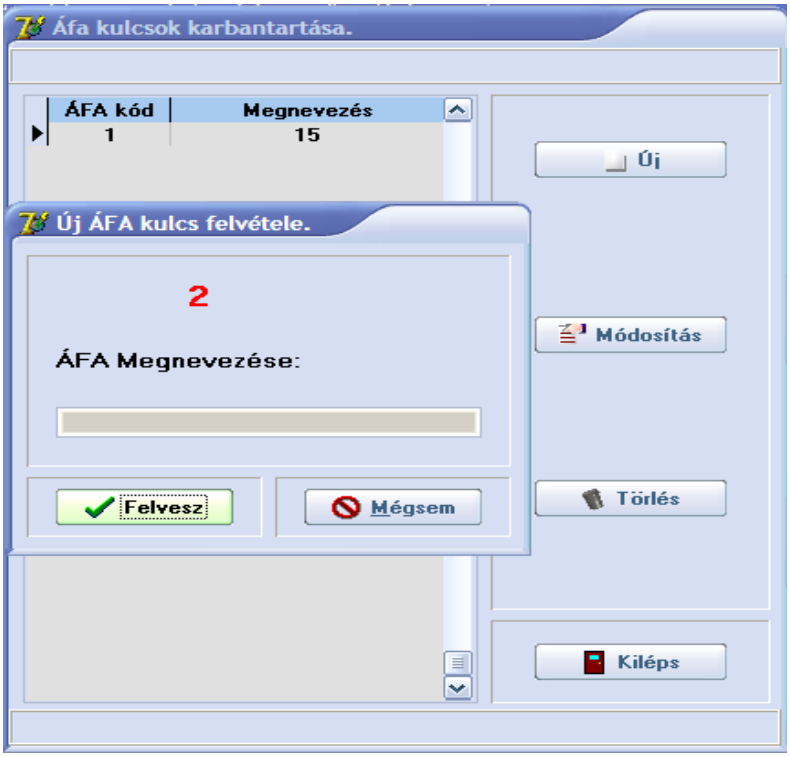
A program fontosabb menüpontjainak részletes elemzése:

1: ***A raktár menüpont***

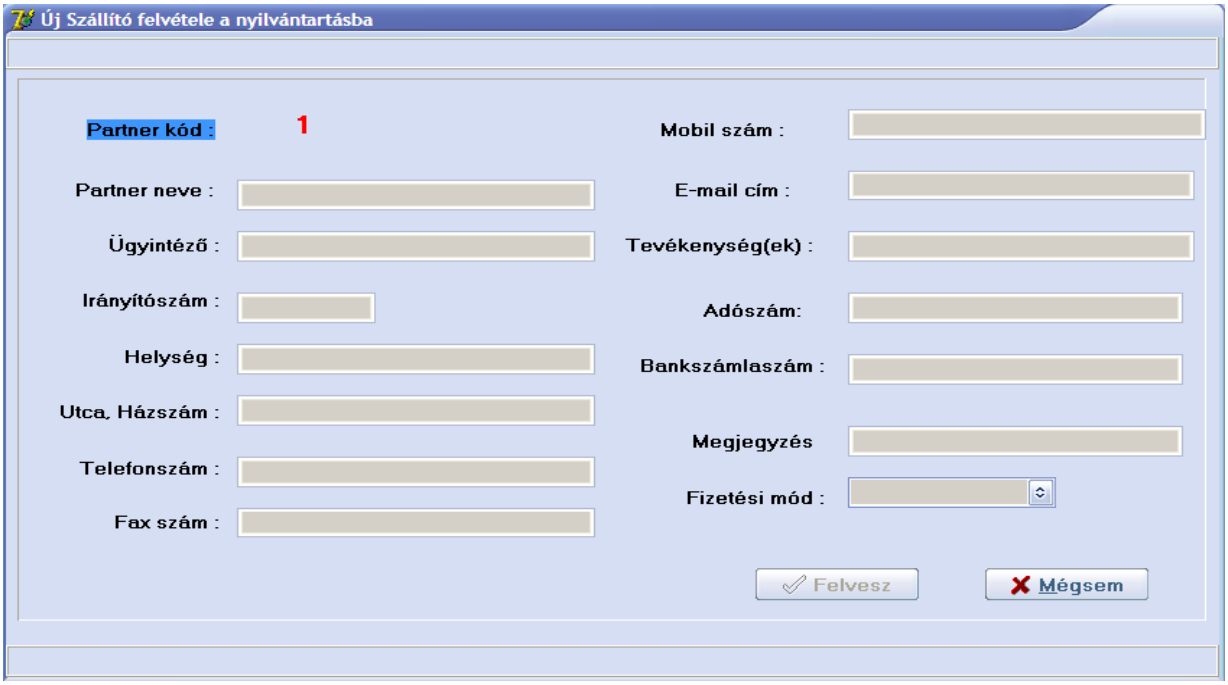


1.1 ***Az új árucikk felvitele***: Mint a nevében benne is van itt a cikktörzset, tudjuk bővíteni az új tételek megadott paramétereinek rögzítésével, itt érdekesség csak akkor, rendszer indításakor következhet be, mert mint a fentiekben látott adatbázis séma alapján a cikkhez elég sok paraméter kötődik, mégpedig a követezőek. Legelsőnek is a Főcsoport a főcsoportok kategorizálásán azt értem, hogy egy cikktörzset szétosztunk szeletekre, mégpedig azáltal, hogy az új cikket besoroljuk például a műanyag vagy papír árú termékcsoport alá, ezáltal növeljük a satisztikai kimutatások lehetőségeit a termékcsoportok kelendősége végett, a főcsoportok rögzítésekor megadunk egy árrés százalékot, hogy mennyit szeretnénk rárakni a beszerzői árra és értékesítés során már ezzel nem is kell foglakoznunk, vagy ezt a kategorizálást igényeink szerint alakítgathatjuk.

Következő paraméter, ami szükséges még a cikk rögzítéséhez az Áfa kulcsok karbantartása ”Általános forgalmi adó” A magyar adórendszerben az általa hozzáadott érték, vagyis az értéknövekedés után az eladónak minden kereskedelmi szinten meg kell fizetnie. Mivel a hozzáadott és nem a teljes érték után fizetendő, az áfa végső összege egy termékre nézve független attól, hogy hány vételen és eladáson megy keresztül. Az áfa indirekt adó, hiszen nem az fizeti be az államnak, akitől a forrás összege származik. A kettős adóztatás elkerülése érdekében az exportált termékek után általában nem kell általános forgalmi adót fizetni, vagy pedig az adóvisszetérítés eszközét alkalmazzák. Az áfa kulcsokat a karbantartások menüpont alatt tudjuk elvégezni.



A következő paraméter, ami szükséges még egy árucikk rögzítéséhez a beszállító meghatározása szintén statisztikai jelentősége van.



A képnek megfelelően meghatározzuk a szállítóra vonatkozó paraméterek, mint a képen is látszik, hogy a felvesz gomb inaktív állapotban ez egy minimális hiba ellenőrzés is én a Vevők és a szállítók felvételére ugyanazt a formot használom a következő módon.

Megnézem a vevő táblát, hogy van e felvéve vevő, ha igen akkor már biztosan van irányítószám rögzítve, ellenkező esetben megvizsgálom az irányítószám táblát, ha nincs, figyelmeztettem a felhasználót, hogy és egyből fel is hozom az irányítószámok karbantartását.

```
if DataModule1.TableVevok.RecordCount = 0 then
 Begin
  if DataModule1.TableIrszam.RecordCount = 0 then
   Begin
    MessageDlg('Sajnos még nincs rögzítve Irányítószám, kérem pótolja a pontos nyílvántartás miatt!!',mtWarning,[mbOk],0);
      FormIrszamok.ShowModal;
   End;
 azonosito :=1
 End
```

Itt egy partner kód generálása folyik, mert sajnos a dBase –ben nincs lehetőség auto incrementálásra, vagy tárolt eljárások használatára.

```
 else
 Begin
    DataModule1.TableVevok.Last;
    azonosito:=DataModule1.TableVevok['PKOD']+1;
 End;
```

*A **wit … do** minősítésre szolgál, majd az Edit mezők text tulajdonságát kinullozom*

```
with VevokSzalitokFelvitele do
Begin
 EditNeve.Text:='';
 EditIntezo.Text:='';
 EditIrszam.Text:='';
```

```
 EditLakcim.Text:='';
 EditTelefon.Text:='';
 EditFax.Text:='';
 EditMobil.Text:='';
 EditEmail.Text:='';
 EditEmail.Text:='';
 EditTev.Text:='';
 EditAdoszam.Text:='';
 EditBankszamla.Text:='';
 EditMegjegyzes.Text:='';
 DBEditHelyseg.Text:='';
 LabelPartner.Caption:=IntToStr(azonosito);
End;
 VevokSzalitokFelvitele.Caption:='Új vevő felvétele a nyilvántartásba';
 VevokSzalitokFelvitele.BitBtnFelvesz.Caption:='Felvesz';
```

A formok kezelését mindenhol hasonló elvvel valósítom meg, ezért a számottevő kód a main.form forrásában található meg. Megnézem mindig a form ShowModal visszatérési értéke mrYes ha igen, egy minimális hiba ellenőrzést végzek a kötelező mezőkre.

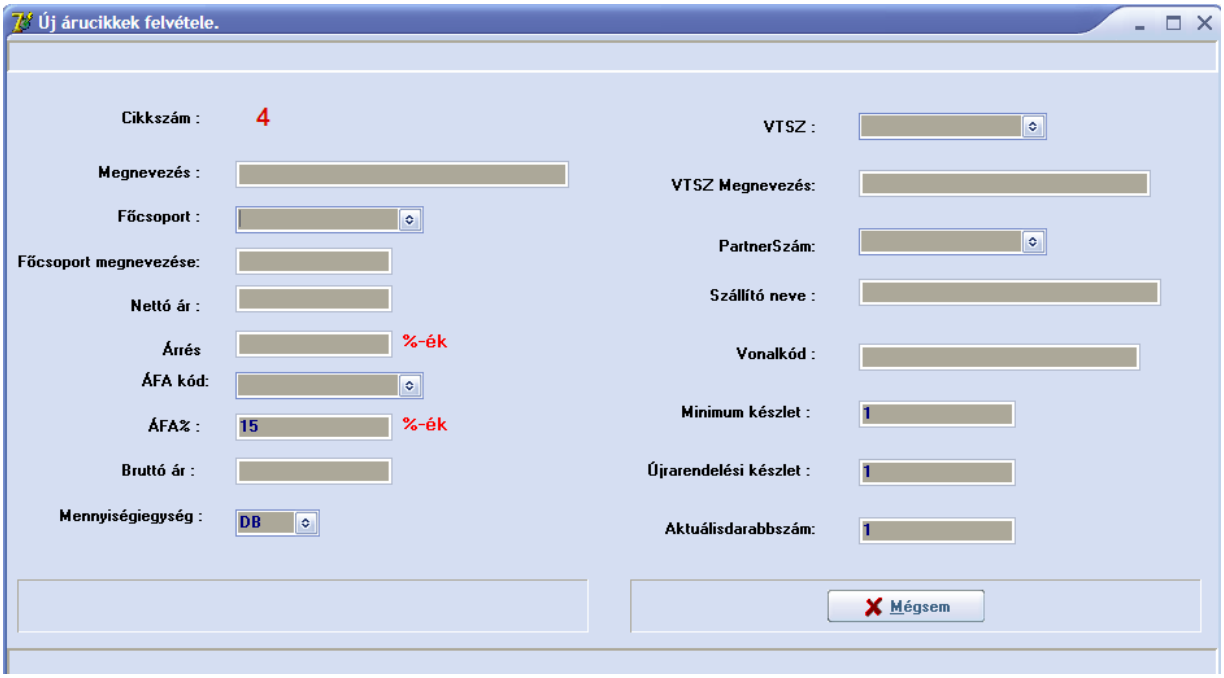
```
  if (VevokSzalitokFelvitele.ShowModal = mrYes) then
   Begin
    With VevokSzalitokFelvitele do
     Begin
      if (EditNeve.GetTextLen = 0)then
       Begin
        ShowMessage('Kérem töltse ki a Partner neve mezőt a pontos nyílvántartás miatt!');
        VevokSzalitokFelvitele.ShowModal;
       End;
      if (EditIrszam.GetTextLen = 0)then
       Begin
        ShowMessage('Kérem Töltse ki az Irányítószám mezőt a pontos nyílvántartás miatt!');
        VevokSzalitokFelvitele.ShowModal;
       End;
      if (EditLakcim.GetTextLen = 0)then
```

```
    Begin
      ShowMessage('Kérem Töltse ki a Házszám mezőt a pontos nyílvántartás miatt!');
      VevokSzalitokFelvitele.ShowModal;
    End;
  End;
```

Ha mindezeken túljutott a program következhet az adatbázisba történő rögzítés, szintén láthatjuk, hogy egy minősítést következik a DataModule1 –re ez egy speciális tároló egység, a fejlesztés során én itt tárolom a táblákat és a hozzátartozó DataSurce – kat. Insert kulcsszó az adatbázist Insert módba állítja, majd egyenként megadjuk a rögzítendő mezőket hasonló módon „FieldByName('PKOD').AsInteger:= azonosito;" mezőnév, típus, mivel legyen egyenlő. A Post pedig véglegesítésre szolgál.

```
  with DataModule1.TableVevok, VevokSzalitokFelvitele do
   Begin
   Insert;
    FieldByName('PKOD').AsInteger:=azonosito;
    FieldByName('PNEV').AsString:=EditNeve.Text;
    FieldByName('UGYINT').AsString:=EditIntezo.Text;
    if Length(EditIrszam.Text) <> 0 then
    FieldByName('IRSZAM').AsInteger:=StrToInt(EditIrszam.Text);
    FieldByName('HAZSZAM').AsString:=EditLakcim.Text;
    FieldByName('TELSZAM').AsString:=EditTelefon.Text;
    FieldByName('FAX').AsString:=EditFax.Text;
    FieldByName('MOBIL').AsString:=EditMobil.Text;
    FieldByName('EMAIL').AsString:=EditEmail.Text;
    FieldByName('TEV').AsString:=EditTev.Text;
    FieldByName('ADSZAM').AsString:=EditAdoszam.Text;
    FieldByName('BANKSZAM').AsString:=EditBankszamla.Text;
    FieldByName('MEGJEGY').AsString:=EditMegjegyzes.Text;
    FieldByName('FIZMOD').AsString:=ComboBoxFizmod.Text;
    Post;
   End;
 End;
```

Ha sikerült rögzítenünk a fentiekben felsorolt adatokat következhet a cikktörzs bővítése, mely a képen látható paraméterekkel rendelkezik



1.2 **Főcsoportok rögzítése**. A fentiekben bemutatásra került

1.3 **Árjegyzék készítése**. Általános árucikkjegyzék, árjegyzék egy partnernek ez a funkció végül egy kimutatás, ami csak annyiban tér el egymástól, hogy a partnernek történő kimutatáskor szerkeszthetjük a fejlécet a személyes megszólítás miatt. Paraméterezési lehetőségei a

1.4 következők Árucikk számtól - Árucikk számig, Szállító kiválasztása, készlet, rendezettség.

Szűrés eredményeként a következő listát kapjuk.

Zenit 98 BT 2007.05.06.

Józsa

Telefon: 0630456799

Fax 0652111111

Email: zenit@chello.hu

Zoga Abc

Tisztelt Vevő

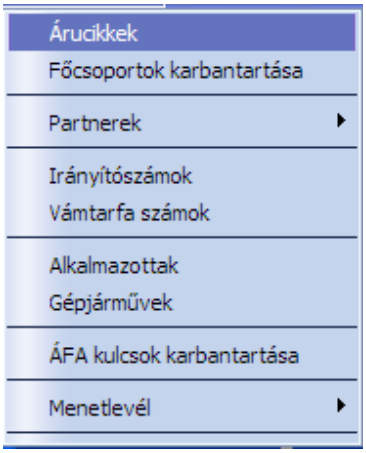
Cikszám	Megnevezés	Mennyiségi egység	Bruttó ár
1	Műanyagtáska kicsi	DB	19
2	Papír Csomagoló	DB	71
3	Papírtáska	DB	33

1.5 **Vevők:** fentiekben bemutatásra került

1.6 **Szállítók**: fentiekben bemutatásra került

1.7 **Új alkalmazott felvétele**: Menüpont feladata a cég alkalmazottjainak fontosabb adatainak rögzítése.

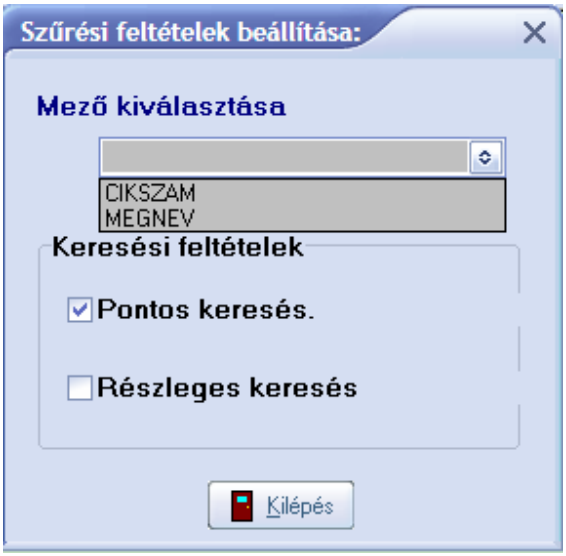
Karbantartások menüpont:



A karbantartás menüpont annyival bővebb, mint a raktár menüpont, hogy itt már lehetőségünk van módosításra, törlésre, keresésre bizonyos feltételek alapján. Röviden bemutatnám azokat a funkciókat, amelyekkel még nem találkozhattunk az eddigiek során.

Alkalmazottak / Autók karbantartása:

Itt tudjuk összekötni alkalmazottainkat és a gépjármű parkunkban lévő autókat. Itt egy Master Detail kapcsol táblát, látunk, ennek a kapcsolatnak a lényege, hogy csak az adott alkalmazotthoz tartozó gépjárműveket látjuk. Árucikkek és a vevők, szállítók karbantartásánál találkozhatunk eddig még nem használt funkcióval a szűréssel.



Kiválasztjuk a szűrendő mezőt és a keresési feltételt, beállítjuk a nekünk megfelelően, vigyázat csak egyszerre egy feltétel bejelölésnek van értelme.

ComboBox feltöltés kódja:

```
procedure TCikktorzs_karbantart.BitBtnSzurClick(Sender: TObject);
Var i:Integer;
Const Lista : array[1..2]of string=('CIKSZAM','MEGNEV');
begin
FormSzur.Mezo.Items.Clear;
  For i:=1 to 2 do
    FormSzur.Mezo.Items.Add(Lista[i]);
FormSzur.ShowModal;
Labelmezo.Caption:= FormSzur.Mezo.Text;
end;
```

Pontos keresés kódja

```
if FormSzur.PontosKeres.Checked then
  Begin
    DataModule1.TableCikktorzs.IndexName:=FormSzur.Mezo.Text;
   if not DataModule1.TableCikktorzs.FindKey([EditMitKeres.Text])then
    MessageDlg('Sajnos nincs Találat',mtError,[mbOk],0);
  End;
```

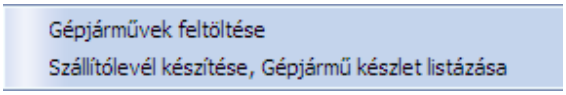
Részleges keresés kódja

```
if FormSzur.ReszlegesKeres.Checked then
  Begin
   DataModule1.TableCikktorzs.IndexName:=FormSzur.Mezo.Text;
   DataModule1.TableCikktorzs.FindNearest([EditMitKeres.Text]);
  End;
```

Tankolási napló kitöltése, tankolási adatok kalkulálása:

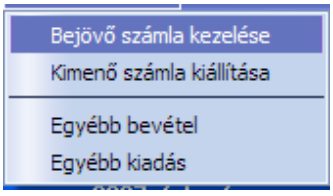
A menetlevél almenüpont alatt tudjuk a tankolási adatokat rögzíteni, kiválasztjuk a rendszámot, majd megadjuk a kilométeróra állását, kitöltjük a tankolt mennyiséget és az árat. Ha több mint egy tankolási adat van rögzítve egy autóhoz, akkor van értelme átlagfogyasztást számolni. Hasonlóan működik, mint az előző kiválasztjuk az aktuális rendszámot és egyből a jobboldali griden láthatjuk a hozzá tartozó adatokat. A jobboldali griden az aktuális recordra klikelve az utána lévőhöz képest megkapjuk az adott értékeket, futott Km, tankolt mennyiség átlagfogyasztás.

Tranzakciók menüpont



A menüpont alatt tudjuk a gépjárműveket az adott cikkeket, majd kinyomtatni a szállítólevelet.
A gépjárművek feltöltése nagyon egyszerű csak kiválasztjuk az aktuális rendszámot, és máris történhet a gépjármű feltöltése. A baloldali griden láthatjuk a cikktáblát, a jobboldalin pedig az utók adatait. A szállító levél nyomtatásának a lényege abban rejlik, hogy miután felpakoltuk az autóra a kívánt cikkmennyiséget akkor az alkalmazott megkapja ezt a listát és csak a napi kiszállítások során csökkenti az értékeket, ezáltal az napvégi elszámoltatás gyorsabb és könnyebben zajlik le.

Bizonylatrögzítés:

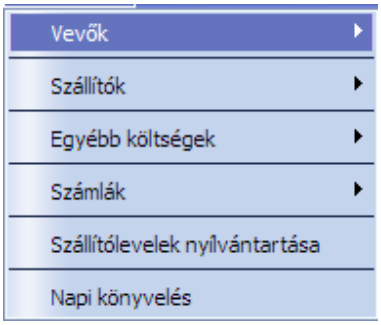


Mint a nevében is benne van bejövő, kimenő és egyéb mozgások rögzítését végezhetjük itt. A bejövő bizonylatoknak készletnövelő, a kimenő bizonylatoknak készlet csökkentő hatása van és itt egy kimenő bizonylat nyomtatása is, történik. Az egyéb bevétel és egyéb kiadásnak csak a pontos nyilvántartásban van szerepe. Érdekes kódrészlet a számlavégi összeg betűvel történő kiíratása

```
function NumToStr( Szam: Extended ): string;
  var Text, sNum: string;
      SLen, ExpTag: Integer;
      s1, s2, s3: Integer;
      Sub: string;
      Minus: Boolean;

begin
 sNum := IntToStr( Trunc( Szam ) );
 Minus := Szam < 0;
```

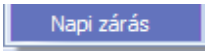
```
  if Minus then
    begin
   sNum := Copy( sNum, 2, $FF );
    end;

SLen := Length( sNum );
ExpTag := 0;
  Text   := '';
while ( sNum > '' ) and ( ExpTag < High( ExpTomb ) ) do
    begin
 s1 := StrToInt( Copy( sNum, sLen, 1 ) );
      if Length( sNum ) >= 2 then
  s2 := StrToInt( Copy( sNum, sLen - 1, 1 ) )
      else s2 := 0;
      if Length( sNum ) >= 3 then
  s3 := StrToInt( Copy( sNum, sLen - 2, 1 ) )
 else s3 := 0;
 Sub := SzamTomb[ s3, 1 ];
      if ( s3 > 0 ) then
       Sub := Sub + 'száz';
 if s1 > 0 then
       Sub := Sub + SzamTomb[ s2, 3 ]
      else
       Sub := Sub + SzamTomb[ s2, 2 ];
 Sub := Sub + SzamTomb[ s1, 1 ];
 SLen := SLen - 3;
 sNum := Copy( sNum, 0, SLen );
 Inc( ExpTag );
 if Sub <> '' then
  if ( ExpTag = 2 ) and ( Text <> '' ) and
           ( ( s3 * 100 ) + ( s2 * 10 ) + s1 > 1 ) then
    Text := Sub + ExpTomb[ ExpTag ] + '-' + Text
  else
         Text := Sub + ExpTomb[ ExpTag ] + Text;
    end;
if Text = '' then
   Result := 'nulla'
  else
 if Szam < 0 then
       Result := 'minusz ' + Text

      else

       Result := Text;

end;
```

Kimutatások menüpont

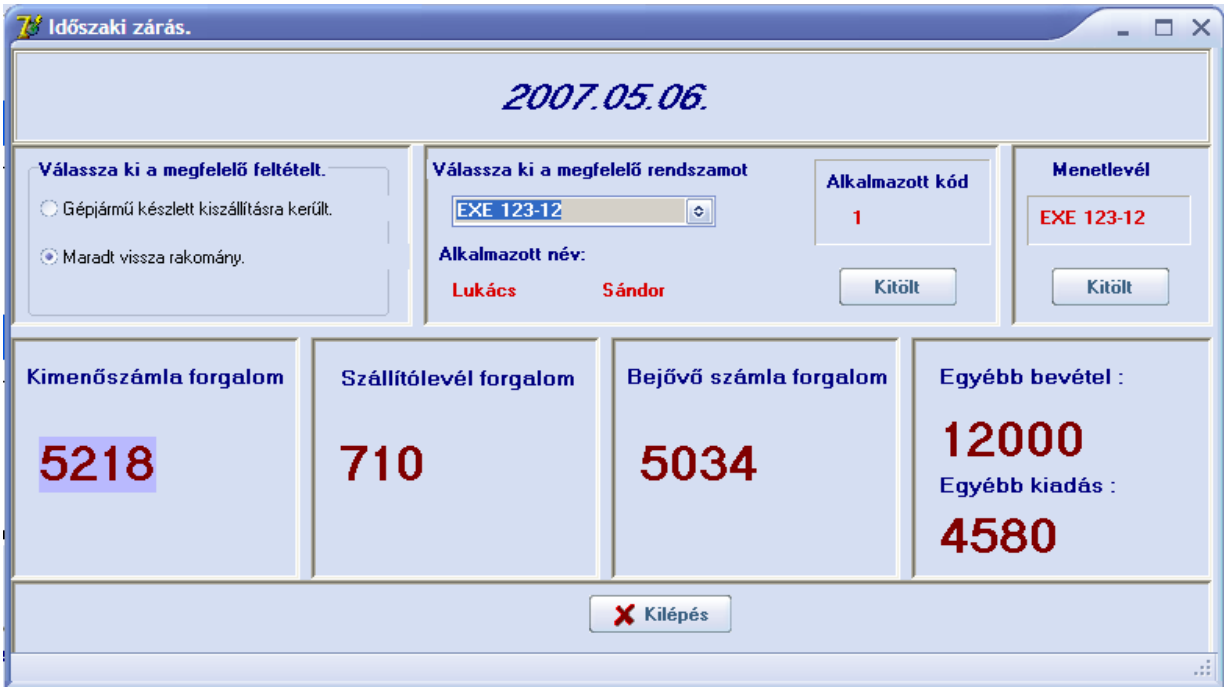


A menüpont alatt egy listát kaphatunk az eddig rögzített vevőkről, szállítókról, egyéb költségeinkről, napi könyveléseinkről.

Zárás menüpont:

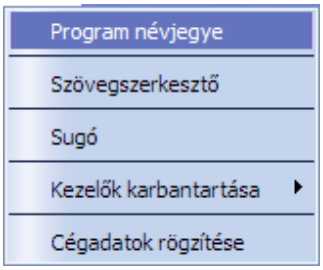


A napvégi elszámoltatások, bevételek kiadások összesítése. A menüpontra klikkelve egyből egy jelszó bekérő formal találkozunk a jelszó „START” szerepe, illetéktelen emberek ne tudják összezavarni a napi zárásokat.



Mint láthatjuk összegződnek a kimenő, bejövő, egyéb kiadás bevétel összegei. Az adott rendszám kiválasztásával módosíthatjuk a gépjármű készlet tartalmát, és szerkeszthetjük a menetlevelet is. Ha az összes tartalom kiszállításra került akkor elég kiválasztanunk az aktuális rádió buttont és program csökkenti helyettünk a gépjárművek készleteit.

Egyéb menüpont:



Nem gyakori funkciók itt, ami érdekes a kezelők karbantartása és a cégadatok rögzítése. A kezelők karbantartásán belül hozhatjuk létre a felhasználót, adhatunk jelszót vagy ideiglenesen az aktív státuszt, elvehetjük a felhasználótól. Cégadatok rögzítése a fentiekben már ki lett fejtve igazából csak reportok fejlécadatait szolgálják.

Összegzés

A szakdolgozathoz készített program fejlesztése során igyekeztem minél több Delphi komponenst használni, és az adatbázis-alapú komponensek lehető legtöbb lehetőségét kipróbálni. Azonban az idő végessége, valamint a komponensek felhasználhatósági módjai ezt csak részben engedték meg.

Az eredményül kapott alkalmazás a feladatát teljes mértékben el képes látni, ennek ellenére számos továbbfejlesztési lehetőséget rejt még magában, melyek iránya szerteágazó:

- Többnyelvűség megvalósítása
- Kimutatások számának növelése
- Mentési pontok létrehozás (napi, heti, havonta biztonsági mentés készítése)
- A megjelenítés testreszabhatósága

A Delphi fejlesztőrendszere mindezekhez támogatást nyújt, gyakorlatban a lehetőségeknek csak az emberi fantázia szab határokat, összességében megállapítható, hogy a Delphi alapú alkalmazás-fejlesztésnek nagy jövője van, mivel egy felhasználóbarát, megbízható, egyénre tervezett alkalmazás kifejlesztése gyorsan kivitelezhető a segítségével.

Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Bajalinov Eriknek, hogy tanácsaival, javaslataival segítette szakdolgozatom elkészítését

Irodalomjegyzék

Marco Cantù: Delphi 3 mesteri szinten I-II. Kötet

Marco Cantù: Delphi 7 mesteri szinten I-II. Kötet

Baga Edit: Delphi másképp

Szabó László: A Pascal programnyelv

Borland Delphi 7 help

Sybex.Mastering.Delphi 7 eBook

http://delphi.madscan.hu

http://www.prog.hu

http://www.delphizine.com

http://www.torry.net

http://www.softwareonline.hu