Debreceni Egyetem
Informatikai Kar

SZOFTVERFEJLESZTÉS DELPHI-BEN

Témavezető:
Dr. Bölcskei András
Egyetemi docens

Készítette:
Tarsoly Zoltán
Programozó matematikus

Debrecen
2008

Tartalomjegyzék

1. Bevezetés

Szakdolgozatom témájaként nem véletlenül választottam a Szoftverfejlesztés Delphi-ben címet. A Delphi az általam egyik legkedveltebb fejlesztőkörnyezet. Fejlesztettem már CodeGear, NetBeans és Visual Studio rendszerekben is, sőt a régebbi környezeteket nem is említem. Ebből adódóan programoztam már C, C++, C# ás Java nyelveken is, természetesen az egyetemi gyakorlatokon felül, saját problémáim megoldása végett. Ám mind közül a Delphi és az objektumorientált Pascal áll hozzám a legközelebb, hiszen tizenévesen a Pascal nyelvvel kezdtem el a programozással való megismerkedést. Ezen kívül a Delphi-nek rendkívül nagy támogatottsága van a mai napig, ami nem meglepő, hiszen az egyik leginkább felhasználó barát fejlesztőkörnyezet. Az Interneten számtalan ingyenes komponenst lehet fellelni, így egy folyamatosan bővülő és fejlődő eszköz a Delphi. Habár a Pascal nyelv manapság már nem igazán felel meg a kor követelményeinek, de még a C# térhódítása sem tudta kiszorítani a piacról a Delphi-t. Sőt a Delphi készítője a Borland cég folyamatosan fejleszti ezt a környezetet, ahogy az újabbnál-újabb technológiák napvilágot látnak, így most már a Delphi-t el lehet érni, például a Microsoft által fejlesztet .NET támogatással is, ezzel is felvéve a versenyt a konkurenciával. Mivel eddig a legtöbb alkalmazásomat Delphi-ben fejlesztettem, ezért a témaválasztás számomra evidens volt.

Szakdolgozatomban egy egyszerű alkalmazás fejlesztésének lépésein keresztül fogom bemutatni a Delphi fejlesztőkörnyezet hatékonyságát, kitérve a leggyakrabban használt komponensek megismertetésére. Maga a szoftver csupán egy alap problémát hivatott megoldani, viszont a benne felhasznált komponensekkel széles körben lehet egyéb alkalmazásokat fejleszteni. Dolgozatommal azt a célt akarom elérni, hogy az olvasó (akár laikus, akár gyakorló programozó, aki még nem ismeri ezt a környezetet) a Delphi-ben való szoftverfejlesztés alapjait maradéktalanul képes legyen elsajátítani.

Bevezetés gyanánt még általában az objektumorientáltságról néhány szó. Az objektumorientált nyelvek egyik alap egysége az objektum. Az objektumoknak vannak tulajdonságaik, amiket attribútumoknak nevezünk és vannak módszereik, melyekkel megváltoztathatjuk az objektum állapotát. Objektumokat az osztályok példányosításával tudunk létrehozni. Ezért a kódot újrahasznosíthatjuk bármikor és bármennyiszer. Az

objektumok memóriában elfoglalt helyét ismerjük, és címük segítségével hivatkozhatunk is rájuk. Az éppen a memóriában lévő objektum adatait az objektum állapotának nevezzük. Minden objektumnak van kezdőállapota, mely a példányosításkor alakul ki. Egy objektum egy másik objektummal az interfészükön keresztül léphet kapcsolatba, ami meghatározza, hogy mely attribútumokat és módszereket lehessen használni. Az objektumok üzenetek formájában kommunikálnak egymással, melynek következtében lehet, hogy megváltozik az állapotuk. Minden objektum egyedi és rendelkezik objektumazonosítóval (Object Identifier – OID).

Az osztályok között lehetséges öröklődést definiálni. Ekkor az alosztály örökli a szülőosztály minden attribútumát és módszerét, melyeket megváltoztathat vagy akár újakat is létrehozhat. Az osztályok eszközeit is korlátozhatjuk a bezárás segítségével. Általában három szintet különböztetnek meg a nyelvek: publikus, védett és privát. A publikus (public) szint minden osztály számára elérhető. A védett (protected) szint csak a leszármazott osztályok által látható. A privát (private) szint, pedig csak az adott definiáló osztály által teszi elérhetővé az eszközöket. A nyelvek általában megengedik, hogy több azonos nevű, de különböző paraméter listájú módszereket definiálhassunk, ezt nevezzük túlterhelésnek.

A Delphi-ben a nélkül is lehet szoftvert fejleszteni, hogy akár egyetlen osztályt is létrehoz a programozó. A fejlesztőkörnyezet automatikusan létrehoz egy alapértelmezett osztályt. Mindenegyes komponens, amit felhasználásra kerül a fejlesztés alatt ennek az osztálynak lesz az objektuma. Így tehát az objektumok használata elemi része a Delphi-ben való alkalmazásfejlesztésnek. Ezért fontos, hogy a fenti információkat legalább alap szinten ismerje a felhasználó.

2. A Delphi-ről

A Delphi az objektumorientált Pascal nyelvet használja a kódoláshoz, ami nem tisztán OO nyelv, hanem úgynevezett eljárásalapú OO nyelv. Azaz a felhasználónak lehetősége van az osztályoktól és objektumoktól független eljárásokat, változókat és rekordokat írnia, ugyanakkor a legtöbb OO eszköz a rendelkezésére áll a fejlesztéshez.

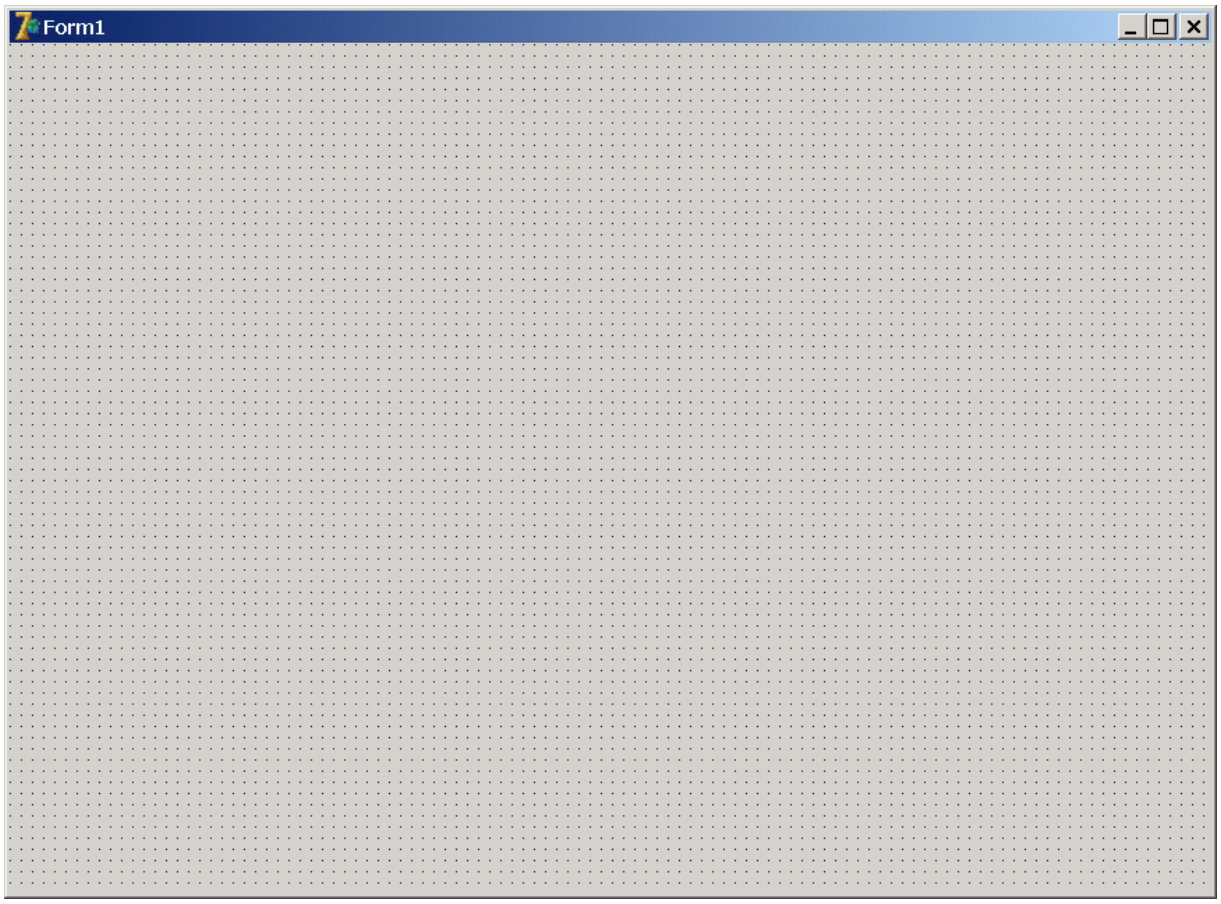
Mindeközben a Delphi eseményvezérelt környezet is, melynek lényege, hogy olyan függvényhívásokat készíthetünk, amelyek egy esemény bekövetkeztéig a program futását blokkolják, majd erre az eseményre a program a függvény hatására reagál valamit. Kétféle eseményt különböztet meg a Delphi. A felhasználói események a kommunikációt biztosítják. Például: egy billentyű lenyomásának hatására a program futása leáll. A rendszeresemények a program állapotáról nyújtanak információt. A programozó mindkét fajtájú eseményhez létrehozhat függvényeket. Az eseménykezelő (Event Handler) gondoskodik az események bekövetkeztekor a megfelelő kódrésznek a végrehajtásáról. Vannak eseményfigyelők, melyek értesítik az eseménykezelőt, ha egy adott esemény bekövetkezik.

2.1 A Delphi integrált fejlesztőkörnyezete

Amikor elindul a Delphi, egyből az integrált fejlesztőkörnyezetét (Integrated Development Environment – IDE) tölti be. Ez az IDE minden eszközt meg ad a gyors alkalmazástervezés, -fejlesztés és -teszteléshez. Éppen ezért újabban RAD (Rapid Application Development – gyors alkalmazásfejlesztés) technológiának is nevezik az ilyen környezeteket.

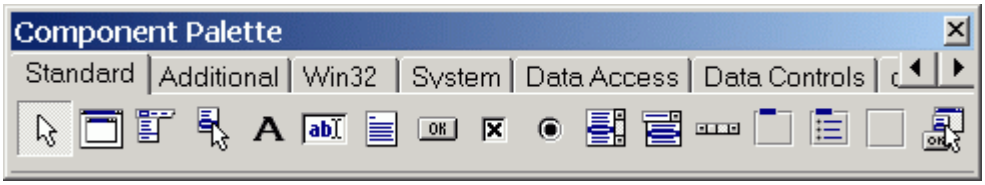
Az IDE a következő eszközöket nyújtja:

- Formatervező (Form Designer): egy üres ablak a felhasználói interfész (User Interface – UI) tervezéséhez



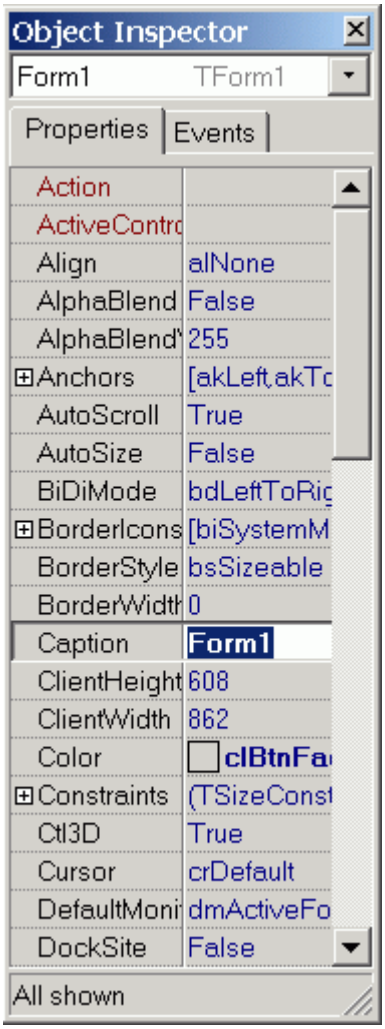
1. ábra

- Komponenspaletta (Component Palette): vizuális és nem vizuális komponensek kiválasztásához



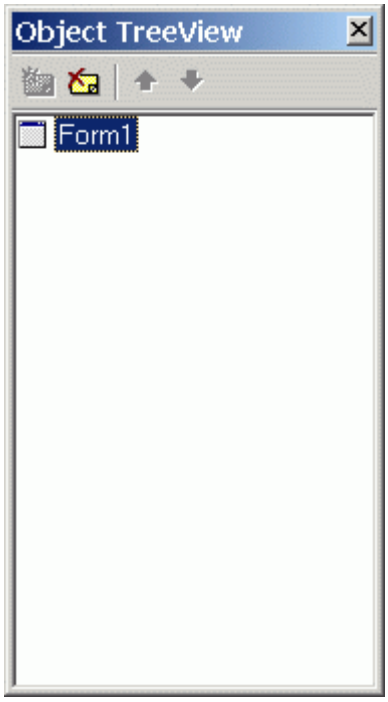
2. ábra

- Objektum felügyelő (Object Inspector): az objektumok tulajdonságainak és eseményvezérlőinek vizsgálatához és szerkesztéséhez



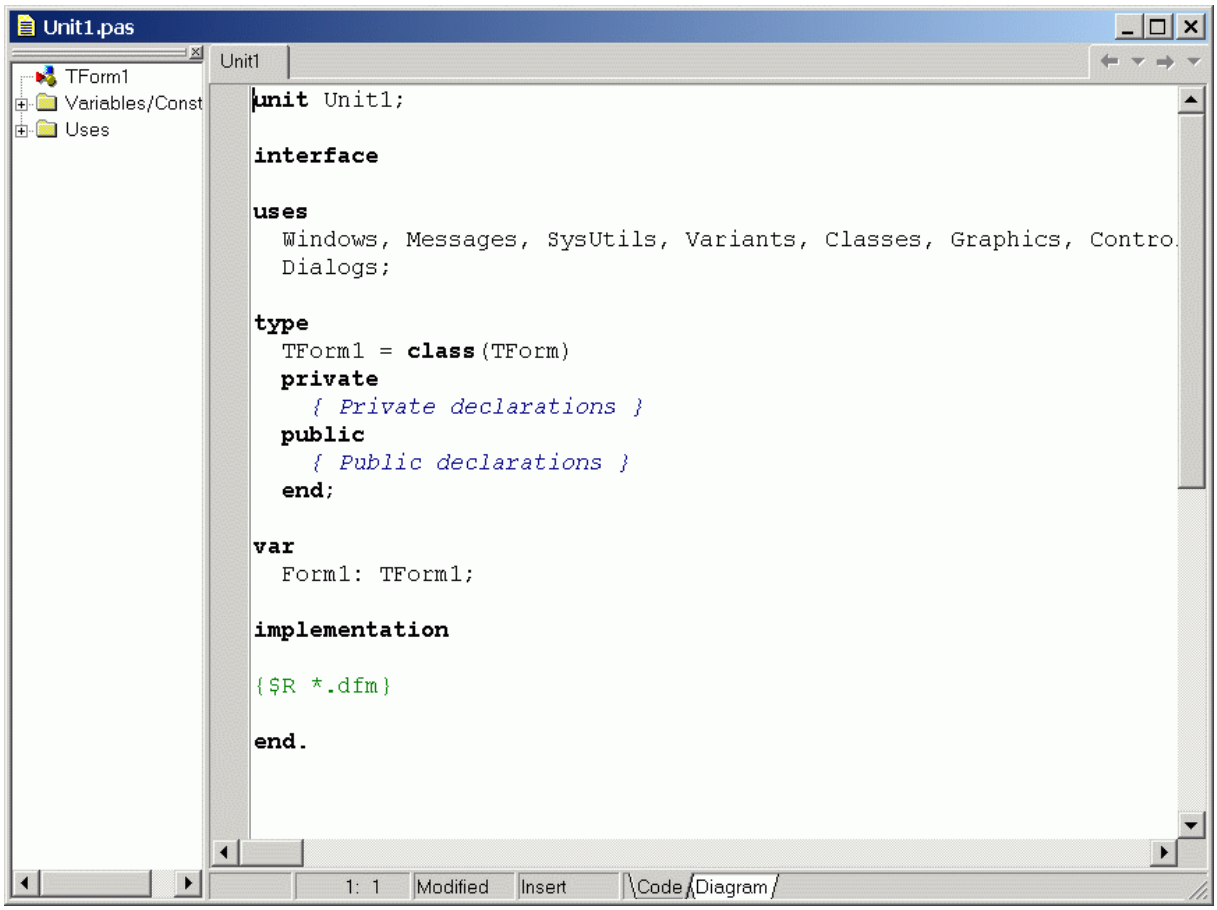
3. ábra

- Objektum fanézet (Object TreeView): a komponensek logikai kapcsolatának megjelenítéséhez és szerkesztéséhez



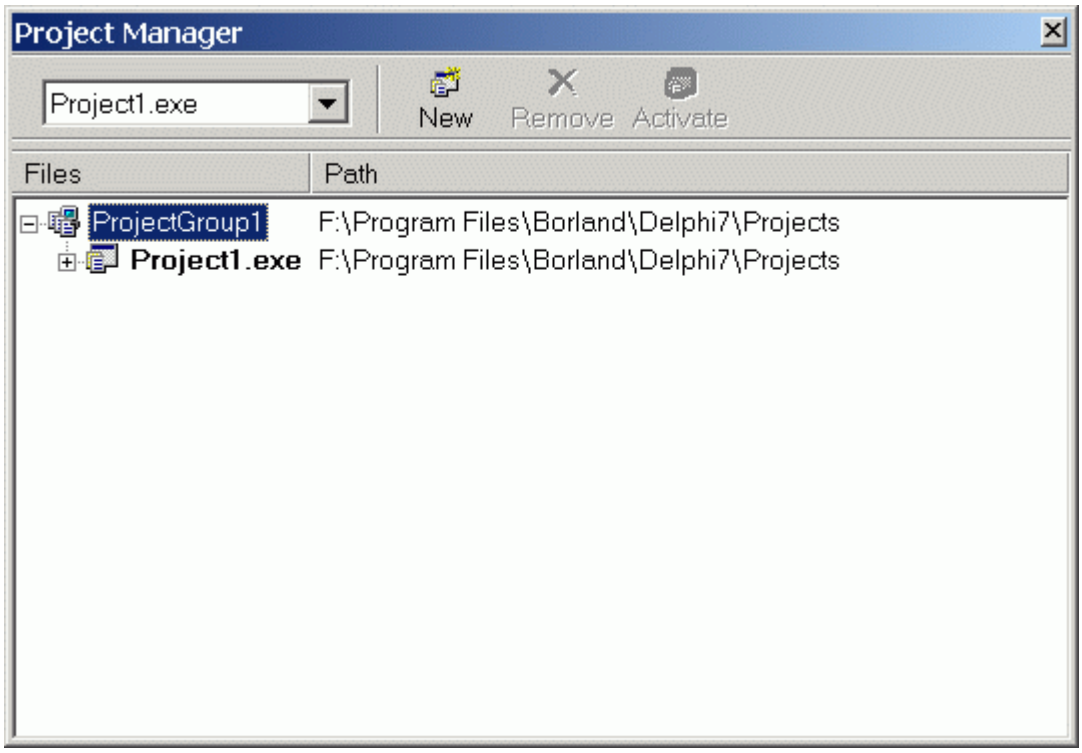
4. ábra

- Kódszerkesztő (Code Editor): a program forráskódjának írásához és szerkesztéséhez



5. ábra

- Projektmenedzser (Project Manager): a projektek fájljainak menedzseléséhez



6. ábra

- Integrált nyomkövető (Integrated Debugger): a kód hibáinak megtalálásához és kijavításához
- Parancssori eszközök (Command-line Tools): fordítók, összekapcsolók és más felhasználói programok

3. A példa szoftver

A Delphi-ben való alkalmazásfejlesztés bemutatására egy Névjegy Katalógus szoftvert használok. Ennek a programnak célja, hogy névjegyeket lehessen eltárolni, és azokat bármikor elérhessük, szerkeszthessük.

A programban a következő adatokra lesz szükségünk:

- Név
- Város
- Cím
- Irányítószám
- E-mail
- Telefonszámok
    - Otthoni
    - Munkahelyi
    - Mobil
- Megjegyzések
- Kép

Az adatokat a legcélszerűbb egy táblázatban tárolni, ami egy Paradox 7 formátumú tábla lesz. A programmal a tábla minden rekordjának minden mezője szerkeszthető, illetve az egész tábla megtekinthető.

A program használata evidens. Új képet a Kép címke alatti képre kattintva lehet betölteni. A mezők kitöltésére nincsenek megkötések.

4. Szoftverfejlesztés a Delphi-ben

Ebben a fejezetben a példa szoftverhez szükséges komponensek kerülnek részletesebb bemutatásra.

4.1 Alkalmazás létrehozása

Új alkalmazást a File/New/Application menüponttal lehet létrehozni. Ekkor a Delphi létrehozza az alkalmazáshoz szükséges fájlokat és a még üres form-ot. Célszerű egyből a File/Save Project As menüponttal elmenteni a még nem szerkesztett projektet. Ekkor adhatunk a projektünknek nevet. A későbbiekben elkészített .EXE fájlunk is ezt a nevet fogja kapni. Ennek a névnek Delphi legális azonosítónak kell lennie, azaz az angol abc betűjével vagy _-al kell kezdődnie és az angol abc betűivel, számokkal, vagy _-okkal folytatódhat. A példában ez a „Nevjegy_Katalogus.DPR” lesz. Unit-jainkat is átnevezhetjük a fenti szabályoknak megfelelően, a példában marad a „Unit1.PAS”.

4.2 Beállítások

Néhány beállítás érdemes még a munka előtt elvégezni, hogy ne kelljen később vesződni vele.

4.2.1 Projekt beállítása

A Project/Options menüponttal lehet szerkeszteni a projekt beállításait. Az Application fül alatt a Title-el adható meg az alkalmazás címe. Ez egy legfeljebb 255 hosszú karakterlánc lehet. Ez a cím látszik a tálcán. Ez a példában a „Névjegy Katalógus”. Az Icon-al megadhatjuk az alkalmazás ikonját, mely reprezentálja a programot a Programmenedzserben és minimalizált állapotban.

4.2.2 Form beállítása

A Form Caption tulajdonságával a program fejlécében megjelenő szöveget lehet beállítani. A példában ez a „Névjegy Katalógus”. A Position tulajdonsággal a program elindulásakor a képernyőn elfoglalt helyét lehet szabályozni. Ez a példában a „poDesktopCenter”. Így a program a képernyő közepén fog megjelenni indításkor. A BorderStyle tulajdonsággal a program keretét lehet megadni. Ez a „bsSingle” a példában. Emiatt futásközben nem lehet átméretezni a program ablakát. A BorderIcons tulajdonságban a bsMaximaize-t „false” lesz a példában, így nem lehet majd teljes képernyő méretűre kinyitni a program ablakát.

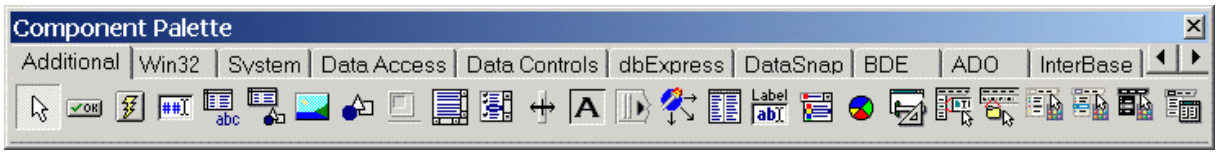
4.3 Tábla létrehozása

A készülendő alkalmazásnak szüksége van egy táblára, melyben az adat le lesznek tárolva. Ilyen táblát könnyedén létre lehet hozni a Delphi fejlesztői csomagjával együtt elérhető Database Desktop nevű programmal. A program elindítása után a File/New/Table menüponttal lehet új táblát létrehozni. Ki kell választani a tábla típusát, ami a példában a „Paradox 7”. Ez után a meg kell adni a tábla mezőinek neveit és tulajdonságait. Ezek a példában „Nev, Varos, Cím, Iranyitoszam, E-mail, Otthoni, Munkahelyi, Mobil, Megjegyzesek, Kep”. Az egyszerűség kedvéért legyen az összes mező típusa „Alpha”, méretük „255” és nincs kulcs. A Save As gombbal lehet elmenteni a táblát, majd szerkeszteni annak mezőinek tartalmát. A példában a tábla neve a „Nevjegyek.DB”. Ezt a fájlt a majdani .EXE fájlunkkal kell egy mappába helyeznünk.

4.4 Komponensek

Komponenseket a Componenet Palette segítségével választhatunk ki. A kívánt komponenst egy kattintással kell kiválasztani, majd a form-on bárhova kattintva lehet azt elhelyezni. Csak a vizuális komponenseknek fontos a pozíciója, a nem vizuális komponenseknek lényegtelen, mivel azok csak tervezéskor láthatóak, futási időben nem.

4.4.1 Additional komponensek

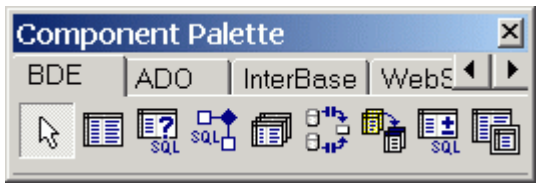


7. ábra

 A SpeedButton komponenssel egy olyan gomb jeleníthető meg, mely képes úgynevezett glyph-et megjeleníteni. Ez a glyph egy .BMP kiterjesztésű kép lehet, melyet a Glyph tulajdonságnál a Picture Editor-ral adható meg. Mivel ezeket a gombokat csak egy ikon reprezentálja a felhasználó számára, ezért érdemes a Hint tulajdonságot kitölteni a megfelelő művelet rövid leírásával. Ezután a ShowHint tulajdonságot true-ra kell állítani. A Cursor tulajdonsággal meg lehet változtatni az alapértelmezett ikont, amikor a felhasználó a gomb fölé viszi az egérkurzort. Ezekkel is több munka lesz, ezért érdemes a Name tulajdonságaikat átnevezni, hogy egyértelműen lehessen tudni, hogy melyik gomb melyik művelethez tartozik. A példában ezek a nevek a következők: „SBUj, SBSzerkesztes, SBMentes, SBMegsem, SBTorles, SBElso, SBElozo, SBKovetkezo, SBUtolso”.

 Az Image komponenssel egy bitmap, ikon vagy metafájl jeleníthető meg. A Stretch tulajdonságának „true”-ra állításával a megjelenítendő képet felnagyítja, vagy lekicsinyíti, hogy beleilleszkedjen a kívánt méretbe.

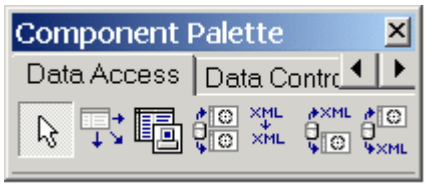
4.4.2 BDE komponens



8. ábra

 A Table komponenssel létesítünk kapcsolatot a katalógus táblájával. Be kell állítani a TableName tulajdonságát. Ez a példában a „Nevjegyek.DB”. Az Active tulajdonságát „True”-ra kell állítani.

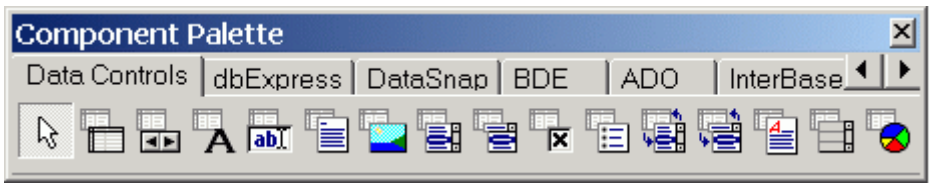
4.4.3 Data Access komponens



9. ábra

 A DataSource komponensre az adateléréshez van szükség. Ez létesít kapcsolatot az adatok, valamint a megjelenítendő objektumok között. A DataSet tulajdonságban meg kell adni az adatforrás nevét. Ez a példában a „Table1”.

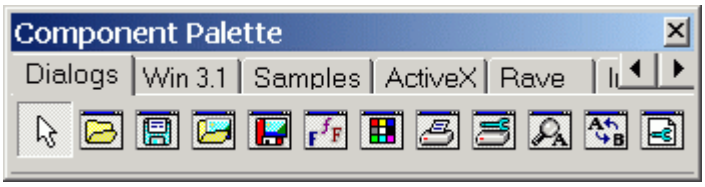
4.4.4 Data Controls komponens



10. ábra

 A DBGrid komponenssel táblázatok adatai jeleníthetőek meg vagy akár szerkeszteni is lehet azokat. A DataSource tulajdonságát mindenkép be kell állítani, ami a példában a „DataSource1”. A Columns tulajdonsággal szerkeszthető, hogy a komponens a tábla mely mezőit jelenítse meg. Új oszlop hozzáadásához az Add New gombra kell kattintani. A példa a következő oszlopokat használja: „Nev, Varos, Cim, Irányitoszam, E-mail, Otthoni, Munkahelyi, Mobil, Megjegyzesek, Kep”. Az oszlopok Title/Caption tulajdonságával a felhasználó által látható oszlop neve szerkeszthető.

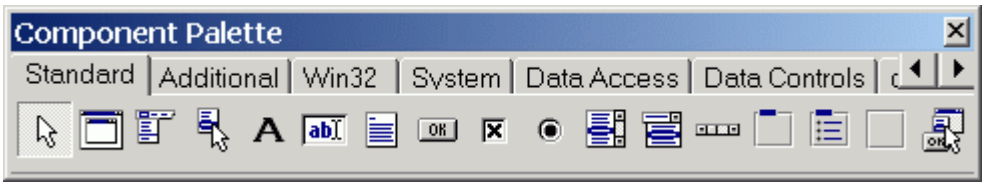
4.4.5 Dialogs komponens



11. ábra

 Az OpenPictureDialog komponens egy Windows-os dialógus ablakot jelenít meg, mellyel grafikus fájlokat választhat ki és nyithat meg a felhasználó. Ez a komponens megegyezik az OpenDialog komponenssel, annyi különbséggel, hogy tartalmaz egy előnézeti ablakot is. A Title tulajdonsággal megadhatjuk a dialógus ablaknak a felhasználó által látható nevét.

4.4.6 Standard komponensek



12. ábra

 A Label komponenssel egy tetszőleges szövege helyezhető el a form-on, melyet a felhasználó nem tud kijelölni vagy szerkeszteni. A szöveg módosításához a komponens Caption tulajdonságát kell szerkeszteni. A példa nyolc ilyen címkét használ, melyek Caption tulajdonságai a következők: „Név:, Cím:, Város:, Irányítószám:, E-mail:, Otthoni:, Munkahelyi:, Mobil:”.

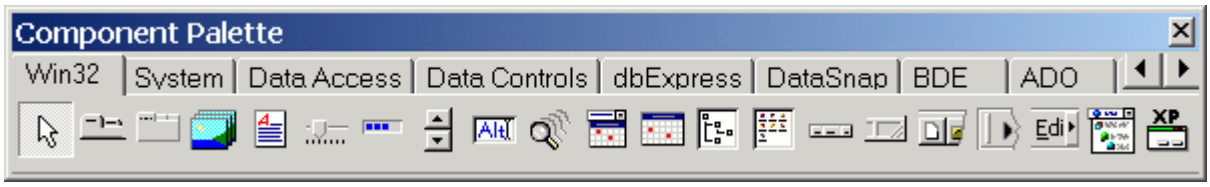
 Az Edit komponens egy szerkesztőterületet jelenít meg, melyben a felhasználó beírhat, vagy módosíthat egy egysoros szöveget. Ezt a szöveget a Text tulajdonsággal lehet elérni vagy szerkeszteni. A példa nyolc ilyen komponenst használ, melyek Name tulajdonságai a következők: „EditNev, EditCim, EditVaros, EditIranyitoszam, EditEmail,

EditOtthoni, EditMunkahelyi, EditMobil”. Mivel ezekkel a komponensekkel a későbbiekben sok munka lesz, ezért érdemes a Delphi által generált Edit1, Edit2, stb. neveket átnevezni úgy, hogy lehessen tudni, hogy melyik mezőhöz melyik Edit tartozik.

 A Memo komponens szintén egy szerkesztőterületet jelenít meg, ám ebben a felhasználó többsoros szöveget is írhat, vagy szerkeszthet. A komponens teljes szövegét a Text tulajdonsággal lehet elérni vagy szerkeszteni. A Lines tulajdonsággal törölhető a „Memo1” felirat a szövegdobozból. A ScrollBar tulajdonságát ssVertical-ra kell állítani, hogy nagyobb szövegek esetén könnyen lehessen navigálni. A példa egy ilyen komponenst használ.

 A GroupBox komponenssel az egy csoportba tartozó komponenseket gyűjthetjük össze. A Caption tulajdonság szerkesztésével adható meg a csoport megjelenített neve. Új komponenseket a GroupBox-ra kattintva adhatunk a csoporthoz, a már meglévő komponenseket, pedig az Object TreeView-ban drag & drop módszerrel ráhúzhatjuk a GroupBox-ra és ez által a komponens a tagja lesz a csoportnak. A példa két ilyen komponenst használ, ezek Caption tulajdonságai a következők: „Telefonszámok, Kép”.

4.4.7 Win32 komponensek



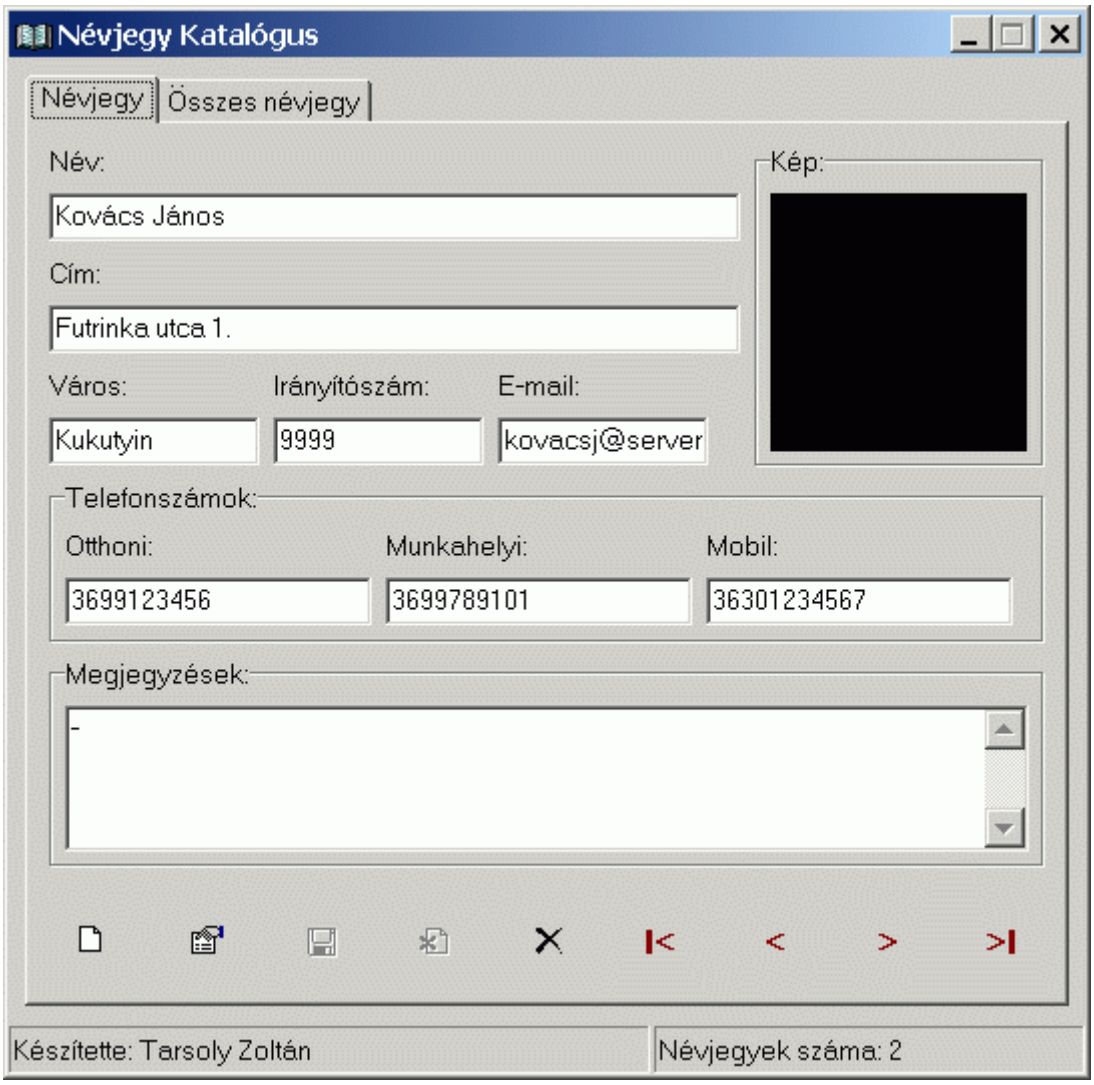
13. ábra

 A PageControl komponenssel egy több lapos dialógust lehet készíteni. A komponensen jobb kattintással lehet új lapot hozzá adni. A példa két lapot használ, ezek a „TabSheet1, TabSheet2”. A TabSheet1 és TabSheet2 objektumok Caption tulajdonságában lehet beállítani a felhasználók által látható lapneveket. A példában ezek „Névjegy, Összes névjegy”. Új komponenseket a kiválasztott TabSheet-re való kattintással helyezhetünk a lapra, vagy az Object TreeView-ban drag & drop módszerrel ráhúzhatjuk a már meglévő komponenst a kívánt TabSheet-re, hogy így az a lap része legyen.

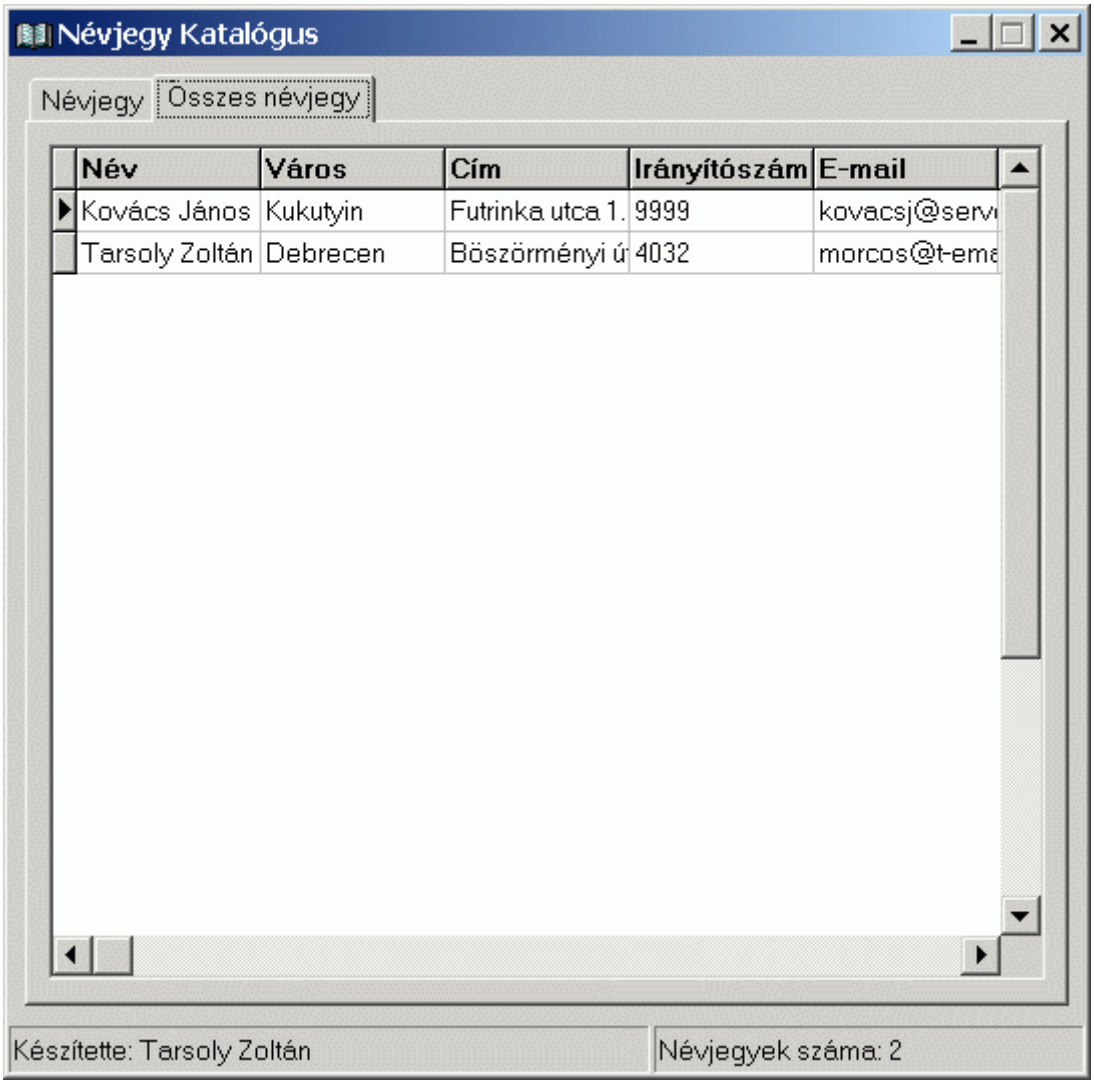
 A StatusBar komponenssel egy terület alakítható ki a form alján, melyben információkat lehet megjeleníteni. A Panels tulajadonsággal szerkeszthető a komponens paneljei. Az Add New gombbal lehet új panelt hozzáadni. A paneleken megjelenítendő szöveget a Text tulajdonsággal lehet megadni. A Width tulajdonsággal a panel szélességét lehet beállítani. A példa egy ilyen komponenst használ két panellel.

4.4.8 Komponensek elhelyezése

Ha minden komponens a form-on van, akkor azokat az alábbi ábrák segítségével a megfelelő helyre lehet helyezni.



14. ábra


Névjegy Katalógus
Névjegy
Összes névjegy
Név
Város
Cím
Irányítószám
E-mail
Kovács János
Kukutyin
Futrinka utca 1.
9999
kovacsj@serv
Tarsoly Zoltán
Debrecen
Böszörményi ú
4032
morcos@t-ema
Készítette: Tarsoly Zoltán
Névjegyek száma: 2

15. ábra

4.5. Eseményvezérlés

Ebben a részben a példa szoftver komponenseinek az eseményekre való reakcióinak a kódja kerül részletes bemutatásra.

4.5.1 A SpeedButton komponensek eseménykezelői

Ezeknek a komponenseknek az OnClick eseményére van szükség. Az esemény neve melletti dupla kattintással a Delphi elkészíti az esemény bekövetkeztekor lefutó eljárás vázát. Ezt az SBUj nevű SpeedButton esetében a következő kép kell módosítani:

```
procedure TForm1.SBUjClick(Sender: TObject);
begin
  SzerkesztokTorles;
  Muvelet:='Letrehozas';
  SBMentes.Enabled:=true;
  SBMegsem.Enabled:=true;
  SBUj.Enabled:=false;
  SBSzerkesztes.Enabled:=false;
  Image1.Picture.LoadFromFile('Nincs_Kep.bmp');
end;
```

Lépésről lépésre a következők történnek. Először meghívódik egy „SzerkesztokTorlese” nevű eljárás. Ez az eljárás az „Egyéb kódok” fejezetben található. Miután ez lefutott egy string típusú Muvelet nevű változó a „Letrhozas” értéket kapja. Ennek egy másik esemény bekövetkeztekor lesz szerepe. Az SBMentes és SBMegsem nevű SpeedButton komponensek Enabled tulajdonságai a „true” értéket kapják, míg az SBUj es SBSzerkesztes komponensek hasonló tulajdonságai a „false” értéket. Meghívódik az Image1 nevű Image komponens Picture tulajdonságának a LoadFromFile nevű metódusa a „’Nincs_Kep.bmp’” paraméterrel. A Nincs_Kep.bmp fájl bármilyen bitmap típusú kép lehet. A program ezt a képet fogja mindig megjeleníteni, hogy ha a felhasználó még nem választott ki saját képet. A példában ez egy 1*1 pixeles fekete kép, ami mivel be lett állítva az Image1 Stretch tulajdonsága „true”-ra, ezért egy 128*128 pixeles teljesen fekete képnek fog látszani a futáskor. A többi komponens kódja a továbbiakban nem lesz ilyen részletességgel levezetve.

Az összes többi SpeedButton OnClick eseménykor lefutó eljárásának a kódja a következő:

```
procedure TForm1.SBSzerkesztesClick(Sender: TObject);
begin
  Muvelet:='Szerkesztes';
  SBMentes.Enabled:=true;
  SBMegsem.Enabled:=true;
  SBUj.Enabled:=false;
  SBSzerkesztes.Enabled:=false;
  EditNev.SetFocus;
end;
```

Ebben az eljárásban a SetFocus metódus az újdonság, mellyel az eljárás végén a beviteli fókusz az EditNev nevű komponensé lesz, így azonnal fogadhat adatokat a billentyűzetről.

```
procedure TForm1.SBMentesClick(Sender: TObject);
begin
  if Muvelet='Letrehozas' then Table1.Insert;
  if Muvelet='Szerkesztes' then Table1.Edit;
  Table1.FieldByName('Nev').AsString:=EditNev.Text;
  Table1.FieldByName('Varos').AsString:=EditVaros.Text;
  Table1.FieldByName('Cim').AsString:=EditCim.Text;
  Table1.FieldByName('Iranyitoszam').AsString:=EditIranyitoszam.Text;
  Table1.FieldByName('E-mail').AsString:=EditEmail.Text;
  Table1.FieldByName('Otthoni').AsString:=EditOtthoni.Text;
  Table1.FieldByName('Munkahelyi').AsString:=EditMunkahelyi.Text;
  Table1.FieldByName('Mobil').AsString:=EditMobil.Text;
  Table1.FieldByName('Megjegyzesek').AsString:=Memo1.Text;
  Table1.FieldByName('Kep').AsString:=OpenPictureDialog1.FileName;
  Table1.Post;
  SBMegsem.Enabled:=false;
  SBMentes.Enabled:=false;
  SBUj.Enabled:=true;
  SBSzerkesztes.Enabled:=true;
  RekordokSzamolasa;
```

```
  Statusbar1.Panels[1].Text:=('Névjegyek száma: ')+IntToStr(Rekordok);
  if Table1.FieldByName('Kep').AsString<>'' then
    Image1.Picture.LoadFromFile(Table1.FieldByName('Kep').AsString)
  else Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)
    +'Nincs_Kep.bmp');
end;
```

Az újdonságok ebben az eljárásban a Table1 nevű komponens metódusai. Az Insert nevű metódusa egy új üres bejegyzést hoz létre a táblában, míg az Edit nevű metódussal az aktuális bejegyzést lehet szerkeszteni. A FieldByName('Nev').AsString metódussal a Table1 aktuális bejegyzésének a Nev nevű mezőjének adható érték, ami a hozzátartozó EditNev komponens Text tulajdonságával lesz egyenlő. Ugyanígy a többi mező is megkapja az értékét. A Megjegyzesek mező a Memo1 komponens Text tulajdonságának az értékét kapja meg, míg a Kep mező az OpenPictureDialog1 komponens FileName tulajdonságának az értékét. A Table1 Post metódusa a szerkesztett bejegyzést elmenti a táblába. Meghívódik a RekordokSzamolasa eljárás. A StatusBar1 1-es indexű paneljének Text tulajdonsága a „'Névjegyek száma: '+IntToStr(Rekordok)” értéket kapja. Az IntToStr függvény egy egészszámú típust karakterlánc típusúvá alakít. Ha az aktuális bejegyzés Kep mezője nem üres, akkor betölti az adott képet, ha igen, akkor a program könyvtárában található Nincs_Kep.bmp fájlt tölti be. A program könyvtárát az ExtractFilePath(Application.ExeName) paraméterű függvény alkalmazásával lehet lekérdezni.

```
procedure TForm1.SBMegsemClick(Sender: TObject);
begin
  RekordokSzamolasa;
  Table1.Cancel;
  if Rekordok>0 then SzerkesztokKitoltese;
  SBMegsem.Enabled:=false;
  SBMentes.Enabled:=false;
  SBUj.Enabled:=true;
  SBSzerkesztes.Enabled:=true;
end;
```

A Table1 Cancel metódusa megakadályozza a bejegyzés módosítását, ha azok még nem voltak elmentve. Ha nem üres a tábla, akkor meghívódik a SzerkesztokKitoltese eljárás.

```
procedure TForm1.SBTorlesClick(Sender: TObject);
begin
  if Rekordok>0 then
  begin
    Table1.Delete;
    SBElozo.Click;
  end;
  RekordokSzamolasa;
  Statusbar1.Panels[1].Text:=('Névjegyek száma: ')+IntToStr(Rekordok);
end;
```

Ha nem üres a tábla, akkor meghívódik a Table1 Delete és az SBElozo Click metódusa. A Delete metódus törli a táblából az aktuális bejegyzést, majd a Click metódussal visszalépünk az előző bejegyzésre.

```
procedure TForm1.SBElsoClick(Sender: TObject);
begin
  Table1.FindFirst;
  if Rekordok>0 then SzerkesztokKitoltese;
end;
```

A FindFirst metódussal a tábla első bejegyzése lesz az aktuális bejegyzés.

```
procedure TForm1.SBElozoClick(Sender: TObject);
begin
  Table1.FindPrior;
  if Rekordok>0 then SzerkesztokKitoltese;
end;
```

A FindPrior metódus az aktuális bejegyzés előtti bejegyzést teszi aktívvá.

```
procedure TForm1.SBKovetkezoClick(Sender: TObject);
begin
  Table1.FindNext;
  if Rekordok>0 then SzerkesztokKitoltese;
end;
```

A FindNext metódus az aktuális bejegyzés utáni bejegyzést teszi aktívvá.

```
procedure TForm1.SBUtolsoClick(Sender: TObject);
begin
  Table1.FindLast;
  if Rekordok>0 then SzerkesztokKitoltese;
end;
```

A FindLast metódussal a tábla utolsó bejegyzése lesz az aktuális bejegyzés.

4.5.2 Az Image komponens eseménykezelője

```
procedure TForm1.Image1Click(Sender: TObject);
begin
  if OpenPictureDialog1.Execute then
    Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;
```

Az Image1 OnClick eseménykor bekövetkező eljárásában, ha az OpenPictureDialog1 sikeresen lefut, akkor az Image1 betölti a FileName tulajdonság által megadott fájlt. A sikeres futást az Execute metódussal lehet ellenőrizni. Így biztosan létező fájl kerül betöltésre.

4.5.3 A Form eseménykezelője

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  PageControl1.ActivePageIndex:=0;
  Table1.Open;
  Table1.FindFirst;
  RekordokSzamolasa;
  if Rekordok>0 then SzerkesztokKitoltese;
  Statusbar1.Panels[1].Text:=('Névjegyek száma: ')+IntToStr(Rekordok);
end;
```

A Form1-nek is vannak lehetséges eseményei. Ilyen például az, amikor létrejön. Ez az OnCreate esemény. Ezzel több olyan beállítást is el lehet végezni, amit tervezőidőben nem

lehet megtenni, viszont a program futása szempontjában fontos. Ilyen beállítás a PageControl1-nek az ActivePageIndex tulajdonsága, mely a 0 lesz. Így mindig a megadott lap lesz az aktív, míg ha ez nincs megadva, akkor a tervezőidőben kijelölt lap lesz az aktív.

4.6 Egyéb kódok

A program működéséhez szükséges egyéb kódok kerülnek bemutatásra ebben a részben.

```
var
  Muvelet: string;
  Rekordok: integer;
```

Két változó definiálása, melyekre szükség van a program kódjának további részében.

```
procedure TForm1.FelfedezoEngedelyezese;
begin
  SBElso.Enabled:=true;
  SBElozo.Enabled:=true;
  SBKovetkezo.Enabled:=true;
  SBUtolso.Enabled:=true;
end;

procedure TForm1.FelfedezoTiltasa;
begin
  SBElso.Enabled:=false;
  SBElozo.Enabled:=false;
  SBKovetkezo.Enabled:=false;
  SBUtolso.Enabled:=false;
end;

procedure TForm1.RekordokSzamolasa;
begin
  Rekordok:=Table1.RecordCount;
  if Rekordok>0 then
  begin
    FelfedezoEngedelyezese;
```

```
    SBSzerkesztes.Enabled:=true;
    SBTorles.Enabled:=true;
  end
  else if Rekordok=0 then
  begin
    FelfedezoTiltasa;
    SBSzerkesztes.Enabled:=false;
    SBTorles.Enabled:=false;
  end;
end;
```

A Rekordok nevű egészszám típusú változó a Table1 komponens RecordCount tulajdonságának értékét kapja meg, ami egy szintén egészszám típusban a bejegyzések számát tartalmazza. Ha a tábla nem üres, akkor meghívódik a FelfedezoEngedelyezese, ha üres, akkor a FelfedezoTiltasa eljárás.

```
procedure TForm1.SzerkesztokKitoltese;
begin
  EditNev.Text:=Table1.FieldByName('Nev').AsString;
  EditVaros.Text:=Table1.FieldByName('Varos').AsString;
  EditCim.Text:=Table1.FieldByName ('Cim').AsString;
  EditIranyitoszam.Text:=Table1.FieldByName('Iranyitoszam').AsString;
  EditEmail.Text:=Table1.FieldByName('E-mail').AsString;
  EditOtthoni.Text:=Table1.FieldByName('Otthoni').AsString;
  EditMunkahelyi.Text:=Table1.FieldByName('Munkahelyi').AsString;
  EditMobil.Text:=Table1.FieldByName('Mobil').AsString;
  Memo1.Text:=Table1.FieldByName('Megjegyzesek').AsString;
  if Table1.FieldByName('Kep').AsString<>'' then
    Image1.Picture.LoadFromFile(Table1.FieldByName('Kep').AsString)
  else Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)
    +'Nincs_Kep.bmp');
end;
```

A FieldByName('Nev').AsString metódussal értéket kap a mezőhöz tartozó EditNev komponens Text tulajdonsága. Ugyanígy a többi szerkesztő Text tulajdonsága is megkapja az értékét.

```
procedure TForm1.SzerkesztokTorlese;
begin
  EditNev.Text:='';
  EditVaros.Text:='';
  EditCim.Text:='';
  EditIranyitoszam.Text:='';
  EditEmail.Text:='';
  EditOtthoni.Text:='';
  EditMunkahelyi.Text:='';
  EditMobil.Text:='';
  Memo1.Text:='';
  EditNev.SetFocus;
end;
```

Az összes Edit komponens Text tulajdonságának a tartalma törlődik, ahogyan a Memo1 Text tulajdonságának tartalma is, majd a beviteli fókusz az EditNev nevű komponensre áll.

4.7 Fordítás és futtatás

Minden készen áll ahhoz, hogy a programot le lehessen fordítani. Ezt a Run/Run menüponttal vagy az F9 billentyű lenyomásával lehet megtenni. A Delphi ekkor elkészíti a futtatható .EXE állományt. A program megfelelő működéséhez szükség van a Nevjegyek.DB és a Nincs_Kep.BMP fájlokra is.

5. Összefoglalás

Az olvasó tehát láthatta lépésről-lépésre, sorról-sorra egy Delphi alkalmazás elkészítését. Bemutatásra került tizenkét komponens működése. Több eseményhez is készült eseménykezelő eljárás. A cél elsősorban a Delphi fejlesztőkörnyezet bemutatása volt. A példa szoftver éppen ezért nem is kapott olyan nagy hangsúlyt. Természetesen a példa szoftver is könnyűszerrel tovább fejleszthető igen komoly alkalmazássá.

Ám a Delphi összes lehetőségét rendkívül nehéz lenne egyetlen dolgozat keretein belül bemutatni. Sőt, szinte lehetetlen! Csak az alap Delphi 7 komponenspaletta több mint 400 komponenst tartalmaz. Az Interneten pedig ezrével érhetők el az ingyenes vagy fizetős komponensek. Mindenesetre remélem, hogy érzékelhető volt, hogy mennyire egyszerű programot készíteni a Delphi-ben. Csupán pár kattintással és néhány sornyi kódolással komplett alkalmazásokat lehet készítni. Sőt! Az Internetről letöltött ingyenes komponenssel és két kattintással komplett Excel táblázatkezelőt lehet készíteni, ami még többet is tud Microsoft-os társánál. Akkor képzeljük el, hogy egy pénzért kapható komponenssel és komoly munkával milyen alkalmazásokat lehet fejleszteni a Delphi segítségével.

6. Irodalomjegyzék

- Kuzmina Jekatyerina, Dr. Tamás Péter, Tóth Bertalan: Programozzunk Delphi 7 rendszerben! ComputerBooks, Budapest, 2005
- Kende Mária, Kotsis Domokos, Nagy István: Adatbázis-kezelés az Oracle rendszerben. Panem, Budapest, 2002
- Juhász István: Programozás 2. MobiDIÁK könyvtár, http://mobidiak.inf.unideb.hu, 2004
- http://www.delphibasics.co.uk