Diplomamunka

Tóth Miklós

Debrecen
2007

Debreceni Egyetem
Informatikai Kar

Alkalmazásfejlesztés 4GL eszközökkel
Segélyek kezelése/nyilvántartása

Témavezető:
Márton Ágnes

Készítette:
Tóth Miklós
Programtervező matematikus

Debrecen
2007

Tartalomjegyzék

1. Bevezetés

Az első számítógépek megjelenésükkor még csak szűk körben és csak feladat specifikusan voltak használatosak. Nagy méretük, magas áruk és nem utolsósorban költséges fenntartásuk miatt csak nagy intézmények, illetve nagyobb vállalatok tulajdonában voltak. A számítógép gyártási technológiájának gyors fejlődése miatt egyre olcsóbbak lettek, méretük hamarosan lecsökkent az épület méretről a mai asztali változatra. A méretek csökkenésével az üzemeltetési költségük is drasztikus csökkent és a megbízhatóságuk is jelentősen javult; pl. már nem kellett minden órában kiégett tranzisztorokat cserélni. A gyártási technológiának köszönhetően lassan kisebb-nagyobb cégek számára is elérhetővé váltak a számítógépek, napjainkban már bárki számára beszerezhetőek. Manapság már személyi számítógép birtoklása nem kiváltságos helyzet. Ennek következtében manapság az informatikai rendszerek a hétköznapi élet szerves részét alkotják. Nyugodt szívvel állíthatjuk, hogy nincs olyan tevékenység, amelyben valaki valahol ne használna valamilyen számítástechnikai rendszert. A tevékenységek a legkülönbözőbbek lehetnek: pl. pénzfelvétel bankautomatából, felsőoktatási intézményekbe a tantárgyak elektronikus felvétele a Neptun rendszeren keresztül vagy boltokban az áru beolvasása vonalkód beolvasó rendszer segítségével. Mivel a régi elvű adattárolás (pl. az egészségügyben a kartonos rendszer) elavult és több hiányossággal is rendelkezik ezért szükség van az adatok számítógépes tárolására ill. a számítógépen tárolt adatok rendszeres archiválására. A tevékenységek közül kiemelkedő szerepet játszanak azok, amelyek adatok tárolásával, feldolgozásával kapcsolatosak. Így minden olyan vállalat, amely versenyképes akar maradni, kénytelen folyamatosan karbantartani, modernizálni, esetlegesen kijavítani az adatok kezelésére, feldolgozására szolgáló rendszereit. Ezzel a folyamattal párhuzamosan megnőtt az olyan informatikusok iránti igény, akik az adatbázis témakörre specializálódtak.

Az egyetemen hallgatott adatbázis-kezeléssel kapcsolatos kurzusok és a kor igényeinek megfelelő adatbázisrendszerek létrehozása és karbantartása olyan szakmai kihívás, amely arra ösztönzött, hogy ezt a témakört válasszam a diplomamunkám témájául. Mivel az adatbázis-kezelés olyan területe az informatikának, amelyre mindig szükség lesz, és így állandó munkalehetőséget kínál, ezért nem csak a diplomamunkámban, szeretnék ezzel foglalkozni, hanem a későbbiekben is. Már korábban is érdekelt az adatbázis-kezelés és mivel

szeretnék a későbbiekben, Debrecenben munkát vállalni ezért azon vagyok, hogy egyre jobbá képezzem magam. Debrecenben a National Instruments (NI) vállalat került az érdeklődési körömbe és mivel az interjú jól sikerült, ezt követte az angol nyelvű szakmai teszt megírása, amit sikeresen teljesítettem. Jelenleg az NI vállalat alkalmazásában állok, mint Service Desk Programmer Analyst.

Ahhoz, hogy érdekes feladatot találjak, megvizsgáltam különböző lehetőségeket, amelyekhez alkalmazás készíthető és igyekeztem ezek közül olyat választani, amely érdekes kihívások elé állít és ösztönöz, hogy még jobban elmélyüljek az adatbázis-kezelésben.

A szociális osztályok informatikai ellátottsága jónak mondható, és szerencsére további javulás várható. Az informatikai megoldások nagyban segítik az állampolgárok szociális igényeinek gyors és hatékony kiszolgálását, az egyes részlegek belső kommunikációját. A mai gazdasági megszorításokkal tarkított világban egyre több család kényszerül arra, hogy segélyt igényeljen az önkormányzati hivataloktól, hogy biztosítsa a megélhetését. A megnövekedett igénylések miatt egyre nagyobb teher nehezedik a szociális osztályokra. Az elérendő cél, hogy a beérkezett igények minél hamarabb elbírálásra kerüljenek így lecsökkenjen az idő mennyiség, amit az igénylőnek ki kell várnia, valamint hogy az ügyintézés minél gyorsabbá és hatékonyabbá váljon. A polgármesteri hivatalok elektronikus segítséget nyújtanak ahhoz, hogy a jelenlegi és leendő ügyfelek web böngészők segítségével információkhoz jussanak. Ennek segítségével az igénylő megtudhatja, hogy milyen hivatalos okiratok szükségesek a segélyigényléshez, illetve a különböző segélyfajtákhoz tartozó hivatalos nyomtatványok is elérhetőek nyomtatható formában. A gyorsaság növekedésével az ellátás emberközpontúvá válik. Mivel gyorsabb az ügyintézés, az emberek is azt érezhetik, hogy fontosak, mert nem kell hosszú napokat várnia abban a tudatban, hogy a „rendszernek” ők nem fontosak.

A diplomamunkám célja, hogy egy olyan alkalmazást készítsek, amely segítséget nyújt a polgármesteri hivatalok szociális osztályának a munkájuk gyors, gördülékeny, kényelmes elvégzéséhez, illetve hogy megkönnyítse a feladatok elvégzését. A rendszer több segélyfajta kezelésére képes, illetve tetszőlegesen bővíthető újabb segélyfajtákkal. A segélyfajták: Átmeneti segély, Étkezési térítési díjtámogatás, Temetési segély, Lakásfenntartási támogatás, Rendszeres szociális támogatás, Beiskolázási segély, Ápolási díj.

A rendszer képes különböző kimutatások nyomtatására: összes segélyigénylés az adott hónapban, adott elbíráló által kiutalt segélyek, a rendszerben nyilvántartott ügyfelek adatai. Az alkalmazás képes az adatbázisban tárolt adatok lementésére és visszatöltésére. A diplomamunkához készítendő alkalmazáshoz az Oracle cég adatbázis-kezelő alkalmazásait használtam név szerint Oracle Express Edition 10G, illetve a szintén Oracle cég által fejlesztett Oracle Form és Report Builder.

E rövid bevezető után had térjek rá az általam készített rendszer tárgyalására: milyen funkciókkal rendelkezik, illetve hogyan képzeltem el a rendszer működését.

2. A rendszer rövid ismertetése

A rendszer a következő felhasználói igények alapján került megtervezésre és elkészítésre. A célom az volt , hogy a felhasználói felületek minél kényelmesebbek legyenek, és hogy a mögöttes funkciók logikusak és jól felépítettek legyenek.

2.1 A rendszer által nyújtott funkciók:

Az osztályvezető:

- Kapjon lehetőséget arra, hogy ha szükséges, tudjon új ügyintézőt felvinni, illetve eltávolítani a rendszerből.
- Meg tudja tekinteni, hogy az egyes ügyintézők kiknek és mekkora összegű segélyt folyósítottak.
- Kapjon lehetőséget arra, hogy havi kimutatásokat készítsen arról, hogy az adott hónapban kinek mekkora összegű segély került folyósításra az adott időszakban.
- A segélyfajták adatai is ő változtathatja meg, illetve új segélyfajtát vihet be a rendszerbe.

A rögzítő:

- Az alkalmazás tegye lehetővé számára az ügyfelekkel kapcsolatos adatok felvételét és esetleges módosítását.
- Biztosítsa az ügyfél számára, hogy megtekinthesse a saját adatait, azt, hogy milyen segélyek kerültek neki kiutalásra és mikor, és ha van még neki kiutalandó segély, akkor mi a folyósítás várható dátuma.

Az elbírálónak:

- Meg tudja tekinteni, hogy az adott ügyfél mikor és mekkora összegű segélyben részesült már korábban.
- Tudja megmondani, hogy ha az ügyfél a korábbiakban már részesült szociális ellátásban, akkor kaphat-e újra segélyt, és ha igen akkor milyen fajtát.

- Kapjon lehetőséget legyen olyan dokumentum nyomtatására, amelyben benne vannak az ügyfél adatai és, hogy mikor, milyen és mekkora összegű segélyt folyósított neki a Polgármesteri Hivatal.
- Legyen jogosult arra, hogy eldöntse, jogos-e a segély igénylés és az ő jóváhagyásával kaphassa meg az ügyfél a kért támogatást.

Az ügyfél:

- Ha igénye van, rá megtekinthesse a személyes adatait, ehhez azonosításra lesz szükség. (A nevére és az adó vagy TAJ számára). A megfelelő azonosítás után elérhetővé váljanak a személyes adatai és hogy milyen segélyek kerültek illetve kerülnek folyósításra számára.
- Az ügyfél értesítést kap a segélyigénylésének az elbírálásáról, akár támogatva akár elutasítva lett.

A rendszerbe olyan funkciókat építettem bele, ami kényelmesebbé teszi a mindennapi használatot értem, ez alatt a teljesség igénye nélkül pl. a beviteli mezőknél legördülő menük használatát, az olyan mezőknél, ahol többet kell gépelni, ahhoz szövegszerkesztőt rendeltem, illetve az aktuális dátum és idő kijelzése. Továbbá egyes mezők kitöltését a rendszer automatikusan végzi el. Próbáltam olyan színeket használni, ami hosszú munka után sem fárasztja a felhasználó szemét így használata komfortosabb. A felhasználói felületeknél cél volt az ergonómia és a „a legkisebb meglepetés elve” ezt egy kicsit részletesebben ki is fejtem. A felületeken elhelyezett funkciógombok következetesek, így nem fordulhat elő, hogy ha a felhasználó megszokta a gombok elhelyezését, akkor nem fog olyan felülettel találkozni, ami eltérne ettől pl. a rögzítés, kilépés gombok mindig adott helyen szerepelnek.

3. A szociális segély igénylésének az ügymenete és a különböző segélyfajták:

3.1 Az igénylés ügymenete:

Röviden ismertetni szeretném a különböző segélyfajtákat, és hogy milyen feltételekkel vehetők igénybe, illetve milyen hivatalos okiratok szükségesek, ahhoz hogy elkezdődhessen a segélyigénylési folyamat. Ha hiányoznak okiratok, illetve, ha nem megfelelően lettek kitöltve a nyomtatványok, nem kezdődhet el a segélyigénylés folyamata. Ilyenkor szükség van az okiratok beszerzésére és/vagy a nyomtatványok újbóli kitöltésére. Ekkor élőszóban segítenek az alkalmazottnak az ügyintézők. Ha megvan minden szükséges okirat, akkor a rögzítő felveszi az igénylő adatait és elmenti az adatbázisba. Csak olyan személy részesülhet segélyben, aki már előzetesen regisztrálva van a rendszerben. Ez után kezdődik a segélyigénylés. Itt kiválasztják, az elérhető segélyfajták közül, hogy melyikre lenne szüksége az ügyfélnek. A rögzítő itt be is fejezi a munkát. A következő lépést az elbíráló teszi meg aki eldönti az alkalmazás segítségével, hogy az igénylő megkaphatja-e az általa kiválasztott segélyfajtát. Az elbíráló indokolja a döntését és rögzíti a megfelelő adatokat pl. a kifizetés időpontját és a segély összegét. Az ügyfél értesítve lesz levélben a segélyigénylésének az elbírálásáról, akár támogatott, akár elutasított az.

3.2 Segélyfajták:

3.2.1 Átmeneti segély

Annak az egyedülálló személynek, vagy családnak állapítható meg, aki létfenntartását veszélyeztető rendkívüli élethelyzetbe került, vagy időszakosan, illetve tartósan létfenntartási gondokkal küzd és jövedelme, illetve családjában az egy főre jutó havi nettó jövedelem nem haladja meg

- egyedül élő személy esetében a nyugdíjminimum 200 %-át,
- 2 fős család esetében a nyugdíjminimum 160 %-át,

- 3 fős család esetében a nyugdíjminimum 130 %-át,
- 4 fős család esetében a nyugdíjminimum 110 %-át,
- 5 vagy ennél több fős család esetében a nyugdíjminimum 100 %-át.

Ha a családban fogyatékos személy él, a jövedelemhatár 20 %-kal emelkedik.

Az Átmeneti segély iránti kérelem elbírálásához szükséges:

- A kérelmező és vele közös háztartásban élő családtagjai rendszeres és nem rendszeres, a kérelem benyújtását megelőző hónapban folyósított nettó jövedelméről kiállított igazolások
- Nyugdíjjal, vagy nyugdíjszerű ellátással rendelkező személy esetén a jogosult nyugdíjfolyósítási törzsszámának egyeztetése
- 16. életévét betöltött gyermek esetében iskolalátogatási igazolás
- Családi állapot igazolása
- Gyermek elhelyezésének igazolása
- A kérelmező vagy családtagja fogyatékosságának, vagy munkaképesség-csökkenésének igazolása

3.2.2 Étkezési térítési díjtámogatás bölcsődések számára

Étkezési térítési díjtámogatás állapítható meg – a jogszabályban meghatározott alanyi jogon járó normatív kedvezményeken túlmenően – a gyermekintézmény javaslata alapján annak a törvényes képviselőnek, akinek a családjában az egy főre jutó havi nettó jövedelem nem haladja meg a nyugdíjminimum 100%-át, és családjában az Önkormányzat által fenntartott bölcsődébe járó gyermeket nevel. Étkezési térítési díjtámogatás csak bölcsődés gyermek után állapítható meg.

Étkezési térítési díjtámogatásra **alanyi jogon** jogosult (bölcsődések vonatkozásában):

- fogyatékos gyermek

- 3 vagy több gyermeket nevelő család
- rendszeres gyermekvédelmi támogatásban részesülő gyermek.

Az étkezési térítési díjtámogatás legfeljebb az adott nevelési év végéig állapítható meg. A nevelési év minden év szeptember 1-től a következő év augusztus 31-ig tart.

Az étkezési térítési díjtámogatás iránti kérelmet az igénybevétel előtt legalább 60 nappal megelőzően kell benyújtani.

Étkezési térítési díjtámogatás iránti kérelem elbírálásához szükséges:

- A kérelmező és vele közös háztartásban élő családtagjai rendszeres és nem rendszeres,**a kérelem benyújtását megelőző hónapban folyósított nettó** jövedelemről kiállított igazolások

Munkaviszonnyal, illetve munkavégzésre irányuló egyéb jogviszonnyal nem rendelkező esetén a Munkaügyi Központ igazolása, vállalkozó esetén az APEH által kiadott, a tárgyévet megelőző évre vonatkozó jövedelemigazolás. Az egyszerűsített vállalkozói adó hatálya alá tartozó személyeknek nyilatkozniuk kell a kérelem beadását megelőző 3 hónap jövedelméről.

- Nyugdíjjal, vagy nyugdíjszerű ellátással rendelkező személy esetén a jogosult nyugdíjfolyósítási törzsszámának egyeztetése
- A gyermekintézmény javaslata a napi személyi térítési díj feltüntetésével
- 16. életévét betöltött gyermek esetében iskolalátogatási igazolás
- Családi állapot igazolása
- Gyermek elhelyezésének igazolása

3.2.3 Ápolási díj

Alanyi jogon, a jövedelmi viszonyokra való tekintet nélkül ápolási díjra az a hozzátartozó jogosult, aki *állandó és tartós gondozásra szoruló súlyosan fogyatékos, vagy tartósan beteg 18 év alatti személy otthoni ápolását végzi.*

Fokozott ápolást igénylő a *súlyosan fogyatékos személy*, amennyiben mások személyes segítsége nélkül önállóan nem képes:

- étkezni, vagy
- tisztálkodni, vagy
- öltözködni, vagy
- illemhelyet használni, vagy
- lakáson belül – segédeszköz igénybevételével sem – közlekedni,

feltéve, hogy esetében a felsoroltak közül legalább három egyidejűleg fennáll.

Ápolási díjra jogosult továbbá a 18. életévét betöltött tartósan beteg hozzátartozó ápolását végző személy, ha a kérelmező családjában az egy főre jutó jövedelem a nyugdíjminimum kétszeresét nem haladja meg.

Ápolási díj iránti kérelem elbírálásához szükséges:

- A háziorvos által kiállított, jelen kérelem 6. oldalán közölt „Igazolás és szakvélemény az ápolási díj megállapításához/felülvizsgálatához" igazolás
- Óvodás, nappali ellátást nyújtó szociális intézményt igénybe vevő, felsőoktatási intézmény nappali tagozatán, valamint közoktatási intézményben tanuló ápolt esetén jelen kérelem 8. oldalán közölt igazolás
- Munkaviszonnyal, illetve munkavégzésre irányuló egyéb jogviszonnyal nem rendelkező esetén
  - a Munkaügyi Kirendeltség igazolása
- Keresőtevékenységet folytató kérelmező esetén igazolás arról, hogy
  - munkaideje – az otthon történő munkavégzés kivételével – a napi 4 órát nem haladja meg
  - amennyiben a kérelmező fizetés nélküli szabadságon van, az erről szóló igazolás

3.2.4 Lakásfenntartási támogatás

A szociálisan rászorult személyeknek, családoknak az általuk lakott lakás vagy nem lakás céljára szolgáló helyiség fenntartásával kapcsolatos rendszeres kiadásaik viseléséhez nyújtott hozzájárulás.

Lakásfenntartási támogatás ugyanazon lakásra csak egy jogosultnak állapítható meg, függetlenül a lakásban élő személyek és háztartások számától. Külön lakásnak kell tekinteni a társbérletet, az albérletet, és a jogerős bírói határozattal megosztott lakás lakrészeit. Bérlőtársak, illetve tulajdonostársak esetében a lakásfenntartási támogatás csak az egyik bérlő, illetve tulajdonos jogosultsága alapján állapítható meg.

A kérelem elbírálásához szükséges:

- A kérelmező és vele közös háztartásban élő családtagjai endszeres és nem rendszeres a kérelem benyújtását megelőző 3 hónapban folyósított nettó jövedelméről kiállított igazolások

  Munkaviszonnyal, illetve munkavégzésre irányuló egyéb jogviszonnyal nem rendelkező esetén a Munkaügyi Központ igazolása, vállalkozó esetén az APEH által kiadott, a tárgyévet megelőző évre vonatkozó jövedelemigazolás. Az egyszerűsített vállalkozói adó hatálya alá tartozó személyeknek nyilatkozniuk kell a kérelem beadását megelőző 3 hónap jövedelméről. Társadalombiztosítási és családtámogatási ellátások esetén a kérelem benyújtását megelőző hónapban folyósított ellátások összegének igazolása.

- Nyugdíjjal, vagy nyugdíjszerű ellátással rendelkező személy esetén a jogosult nyugdíjfolyósítási törzsszámának egyeztetése
- 16. életévét betöltött gyermek esetében iskolalátogatási igazolása
- A lakbérről, a közös költségről (külön szolgáltatások díjáról) és a közüzemi díjakról, továbbá a lakáscélú pénzintézeti kölcsön törlesztő részletéről szóló, a kérelem beadását megelőző egy éven belül kelt három téli és három nyári hónapra vonatkozó, bemutatott számlák, igazolások

- A lakásban lakó személyek lakásban való tartózkodásának jogcímét igazoló irat (tulajdoni lap, adásvételi szerződés, bérleti szerződés), illetve a lakás alapterületének nagyságát igazoló irat
- Családi állapot igazolása
- Gyermek elhelyezésének igazolása

3.2.5 Temetési segély

A temetési számla keltétől számított három hónapon belül annak nyújtható, aki az elhunyt személy eltemettetéséről gondoskodott, annak ellenére, hogy arra nem volt köteles, vagy tartásra köteles hozzátartozó volt ugyan, de a temetési költségek viselése a saját illetve a családja létfenntartását veszélyezteti, és akinek a családjában az egy főre jutó havi nettó jövedelme nem haladja meg a nyugdíjminimum háromszorosát.

Temetési segély iránti kérelem elbírálásához szükséges:

- A kérelmező és vele közös háztartásban élő családtagjai rendszeres és nem rendszeres, a kérelem benyújtását megelőző hónapban folyósított nettó jövedeleméről kiállított igazolások

  Munkaviszonnyal, illetve munkavégzésre irányuló egyéb jogviszonnyal nem rendelkező esetén a Munkaügyi Központ igazolása, vállalkozó esetén az APEH által kiadott, a tárgyévet megelőző évre vonatkozó jövedelemigazolás. Az egyszerűsített vállalkozói adó hatálya alá tartozó személyeknek nyilatkozniuk kell a kérelem beadását megelőző 3 hónap jövedelméről.
- Nyugdíjjal, vagy nyugdíjszerű ellátással rendelkező személy esetén a jogosult nyugdíjfolyósítási törzsszámának egyeztetése
- 16. életévét betöltött gyermek esetében iskolalátogatási igazolás
- Családi állapot igazolása
- Gyermek elhelyezésének igazolása

- Az elhunyt személy halotti anyakönyvi kivonata
- A temetés költségeiről a kérelmező vagy egy háztartásban élő családtagja nevére kiállított számlák eredeti példánya.

Amennyiben a kérelem beadását megelőző hónapban az elhunyt családtag jövedelemmel rendelkezett, annak igazolása

4. A fejlesztői környezet ismertetése

Az Oracle cég által fejlesztett szoftverek alkalmasak kliens/szerver alkalmazások elkészítésére. Képesek nemcsak a kisfelhasználók, de a nagyvállalatok igényeit is kielégíteni. A diplomamunkám elkészítéséhez az internetről letölthető Oracle 10g Express Edition verziót és a szintén Oracle cég által fejlesztett 4. generációs Oracle Form és Report Bulildert használtam. Ez a programcsomag fejlesztő eszközök integrált együttese. Támogatja a segítségével elkészíthető információs rendszerek teljes életciklusát.

A programcsomag néhány jellemzője:

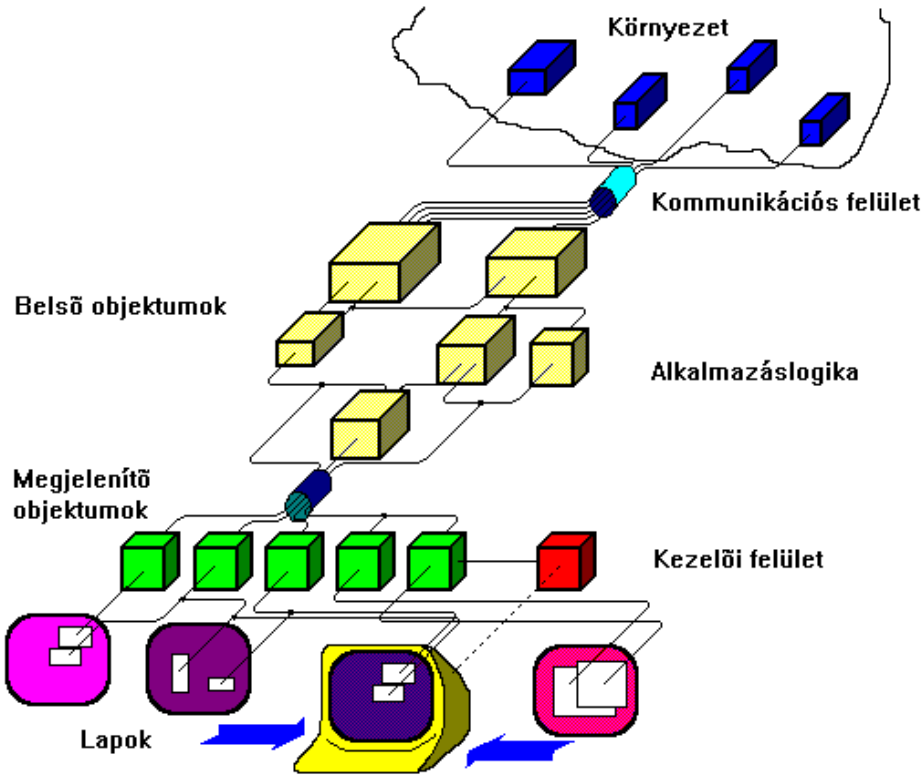
- Grafikus felhasználói interfésszel rendelkezik
- Támogatja a nemzeti nyelvek karaktereit
- Hatékony, produktív, objektumorientált fejlesztőeszközöket tartalmaz.

A fejlesztő eszközök közös belső eszközkészletet használnak, ami megkönnyíti az átjárhatóságot, és közös belső nyelvel is rendelkeznek, ez a PL/SQL. Az adatbázisok méretének és számának gyors növekedése következtében az egyedi alkalmazások létrehozása fárasztó és időrabló feladattá vált, ezért a programfejlesztők törekedtek az adatbázis-kezelés általános formában történő megfogalmazására. Ennek eredményeként jöttek létre az adatbázis-kezelő rendszerek (DBMS) és a negyedik generációs nyelvek (4 GL). Az adatbázis-kezelő rendszerek számos eszközt nyújtanak az interaktív adatbevitel, menük létrehozása terén, melyek kialakítása a harmadik generációs nyelvekben sok-sok oldal kód leírásával lenne csak lehetséges. A negyedik generációs eszközöknél már nem soronkénti programírással kell dolgoznunk, hanem használhatunk ún. űrlapokat (formokat).

4.1 4GL fejlesztőeszközök

A 4GL betűszó a 4th Generation Language (negyedik generációs nyelv) szavak rövidítése. Az elnevezés bevett és akár lépten nyomon találkozhatunk vele félrevezető lehet, ugyanis a 4GL eszközök valójában nem nyelvek, hanem egy vagy több magas szintű nyelvre épülő komplex, objektumorientált programfejlesztői környezetek. Pl. a Basic egy programozói nyelv, de a Visual Basic 4GL

alkalmazásfejlesztő eszköz. Az első 4GL alkalmazásfejlesztő eszközök a 80-as évek közepén jelentek meg, és használatuk a 90-es évek második felére tömeges méreteket öltött. A 4GL eszközök működése azon a tényen alapul, hogy a szoftverrendszerek nem elszigetelt módon működnek, hanem az alábbi ábrának megfelelően feladataik végrehajtása közben folyamatos párbeszédet folytatnak a környezetükkel.



A környezet két részre bontható:

Emberi környezet: A kezelő, akivel a rendszer egy alkalmasan kialakított *kezelői felületen* keresztül tartja a kapcsolatot.

Gépi környezet: Külső számítógépes rendszerek, amelyekből beérkező információk vagy események a rendszerünk működését befolyásolják. Ezekkel a *kommunikációs felületen* keresztül tartja a rendszer a kapcsolatot.

A szoftverrendszer harmadik komponense az *alkalmazáslogika*, mely a rendszer magjaként a feladatspecifikus műveletek végrehajtására szolgáló algoritmusokat foglalja magába.

Egy régi, statisztikai megfigyelés azt mutatja, hogy egy átlagos programozó általa jól ismert programnyelven a feladat kitűzésétől a kész program átadásáig, beleszámítva az elemzés, tervezés, implementáció (kódolás, belövés) idejét, naponta átlagosan 20 sor programot készít el. Ha alkalmazói rendszerről van szó, 10 sort, ha rendszerprogramról, 2(!) sort, ha I/O tevékenységet kell programoznia.

Az I/O tevékenységek aránytalanul nehezebb programozhatóságának fő oka egyébként kicsit tüzetesebb szemlélődés után nyilvánvalóvá válik: ezek a felületek *eseményvezéreltek*. Hogy ez általánosságban mit jelent, azt már tudjuk. A programfejlesztés szempontjából a dolog úgy fest, hogy a programozónak nem egyedi eseményeket, hanem eseménysorozatokat kell helyesen kezelnie, melyek elvileg végtelen sokfélék lehetnek, és kölcsönhatásaik tetszőlegesen bonyolultak. Ezért a kezelői- és kommunikációs felületek tesztelése és belövése rendkívül nehéz feladat.

Mindebből az következik, hogy a legnagyobb bonyodalmak általában nem az alkalmazáslogika megvalósítása körül bukkannak elő, hanem a kezelői felület és a kommunikációs felület létrehozásakor. Más szóval a magasszintű nyelvek segítségével megvalósíthatók ugyan a kezelői felülethez és a kommunikációs felülethez kapcsolódó funkciók, de lényegesen költségesebben, mint a program "belső részeinek", azaz az alkalmazáslogikának a programozása.

Ennek a gondolatmenetnek logikus folytatásaként adódik, hogy a költségek csökkentésének, a programfejlesztés sebességének és az elkészült program megbízhatóságának növelése érdekében olyan fejlesztői környezetre van szükség, amely az ábrán látható séma minden elemének létrehozását erőteljesen támogatja megfelelő céleszközök segítségével. Ilyen fejlesztői környezetek a 4GL eszközök.

A 4GL eszközök, a *kezelői felület létrehozására* speciális szerkesztőket, ún. *látványtervezőket* (*layout editor, dialog editor*) alkalmaznak, melyek segítségével a kezelői felület elemei egyszerűen megrajzolhatók, elrendezhetők, és tulajdonságaik (pl. méret, szín) könnyedén beállíthatók.

A *kommunikációs felület létrehozása* kissé más jellegű feladat, ugyanis míg a kezelő tipikusan ember (és az emberek sok szempontból eléggé hasonlítanak egymásra), a

rendszer környezete nagyon sokféle lehet. Ezért a 4GL eszközök vagy előre definiált környezetet tételeznek föl (pl. relációs adatbázis-kezelő rendszerek, ipari mérésadatgyűjtő rendszerek, kórházi betegfelügyelő rendszerek), és az adott környezettel való kapcsolattartásra alkalmas elemeket eleve tartalmazzák, vagy pedig nyílt rendszerként működnek, és képesek különböző, szabványos illesztőfelületeken keresztül kommunikálni (pl. ODBC, JDBC, különböző hálózati protokollok). Akármelyik eset áll is fönn, a 4GL rendszerek a kommunikációs felület implementálására általában nem tartalmaznak külön eszközöket, legföljebb lehetővé teszik az előre definiált kommunikációs felület viselkedésének a megváltoztatását. Ez érthető is, hiszen a kommunikációs felület létrehozása többnyire a 4GL eszközzel fejlesztett alkalmazás által megvalósítandó funkcióknál sokkal alacsonyabb szintű gondolkodásmódot igényel.

Az *alkalmazáslogika* implementálására magasszintű, gyakran objektumorientált programozási nyelv szolgál. Nagyon sokféle nyelvre alapulnak 4GL környezetek. Így például a Borland Delphi az *Object Pascal*-ra, a Visual Basic és az Oracle Power Objects a *Basic*-re, az Oracle Forms a *PL/SQL*-re, a Visual C++ a *C++*-ra, a Borland JBuilder a *Java-ra*, és még hosszan sorolhatnánk.

4.1.1 Kezelői felület

A kezelői felületen keresztül tudjuk egy működő szoftverrendszer viselkedését befolyásolni. Elméleti szinten a kezelői felületek kialakítása komoly szaktudást igényel, amely nem csak programozástechnikai, hanem ergonómiai, esztétikai és egyéb problémák kezelését is magába foglalja.

A kezelői felületek a kezelőnek a rendszerről kialakított modelljét kell tükrözzék, melyen keresztül a kezelő a géppel kapcsolatba kerülhet. A kezelő fejében a világról számos modell él. Ezek közül jelen vizsgálódásaink szempontjából hármat érdemes kiemelni:

Rendszermodell *:* A kezelő modellje az általa használt rendszer funkcióiról és belső működéséről, melynek alapján képes a rendszer számára absztrakt (azaz bizonyos értelemben véve a világtól elvonatkoztatott) parancsok megfogalmazására. Ilyenkor a

kezelő a lehető legnagyobb mértékben alkalmazkodik a rendszer képességeihez annak érdekében, hogy a kezelői felület a legegyszerűbb, és ez által végrehajtási sebesség és megvalósíthatóság szempontjából a leghatékonyabb legyen. Ezt a fajta kommunikációt *formális párbeszéd* néven is szokás emlegetni. Ebbe a csoportba tartoznak a parancsnyelvek, a menürendszerek és a forma (űrlap) alapú karakteres kezelői felületek.

Vizuális modell: A kezelői felület ez esetben olyan eszközöket nyújt, amelyek megjelenésüket és használatuk módját tekintve alkalmazkodnak, a kezelő mindennapi életében megszokott tárgyakhoz és eljárásokhoz. Ebbe a csoportba tartoznak a grafikus kezelői felületek, melyek előszeretettel alkalmaznak olyan módszereket, mint a fogd-és-vidd (*drag and drop*), és képesek elvileg bármilyen kijelző és adatbeviteli eszköz olyan módon történő megvalósítására, hogy az eszköz egy valós világbeli tárgyra emlékeztessen megjelenését és használatát tekintve egyaránt. Ez a fajta megközelítés jól alkalmazkodik a kezelő intuitív világlátásához, ezért a grafikus kezelői felületek megtanulása általában egyszerűbb, mint a formális párbeszéden alapulóké, viszont a grafikus felületek megvalósítása jóval bonyolultabb, végrehajtásuk pedig sokkal erőforrás-igényesebb, ezért használatuk csak a kilencvenes évek közepére tudott tömegessé válni.

Szemantikus modell: A rendszerrel a kezelő természetes nyelven kommunikál, így elvileg bármit kifejezhet, amire gondol. A természetes nyelvű parancsok a rendszer deklaratív kezelését teszik lehetővé, azaz összetett utasításokat fogalmazhatunk meg tömör, egyszerű módon, ráadásul a rendszer a kezelő által a mindennapi életében használt nyelvet közvetlenül megérti. Itt azonban számos, jelenleg még megoldatlan probléma vetődik föl. A természetes nyelvek elemzése, mint önálló feladat sem teljesen megoldott (már csak azért sem, mert maga a nyelvészet is nyitott tudomány, tehát vannak a természetes nyelvekben nem teljesen tisztázott jelenségek), ráadásul esetünkben az elemzés csak kiindulópontja a megoldáskeresés folyamatának. A jövő minden bizonnyal ez, de a megoldás rendkívül bonyolult, így még várat magára.

A jelenlegi 4GL eszközök a lehetséges kezelői felület modellek közül érthető módon csak az első kettőt támogatják.

A kezelői felület bemeneti és kimeneti hardver eszközei hosszú fejlődés után - néhány egzotikus alkalmazástól eltekintve - napjainkra egységessé váltak: bemeneti eszközként klaviatúra és egér szolgál, kimeneti eszközként pedig képernyő. A klaviatúráról parancsokat és adatokat gépelhetünk be, az egérrel pedig a képernyő elemei közül választhatunk, és rajtuk különböző műveleteket végezhetünk (pl. kurzor pozíciójának megváltoztatása, szövegrészek kiválasztása, nyomógombok aktivizálása, másolás). Az alkalmazás pillanatnyi állapotát a képernyőn követhetjük nyomon.

A kezelői felület megvalósításakor alkalmazható elemek és eljárások szempontjából lényeges a képernyőnk felbontása, mely kétfajta lehet:

Karakteres fejlesztőeszközök esetén a képernyőt $n \times m$ karakterre bontják, ahol n az egy sorban elhelyezhető karakterek maximális száma, m pedig a képernyőn megjeleníthető sorok maximális száma (tipikus felbontás a 80× 25). A karakteres megjelenítés régi, jól bevált módszer, és manapság is rengeteg ilyen módon működő alkalmazás létezik. Ezt használja minden terminál, és számos operációs rendszer, így például a DOS és a UNIX.

Grafikus fejlesztőeszközök esetén a képernyőt képpontokra (*pixel*) bontják. A képpontok száma mindenképpen több százezer, de egészen hétköznapi számítógép konfigurációkon is könnyen meghaladhatja a milliót. Mindez a megjelenítést rendkívül rugalmassá teszi.

Míg a karakteres fejlesztőeszközök csak a formális párbeszéden alapuló kezelői felületek létrehozását támogatják, addig a grafikus fejlesztőeszközök emellett a grafikus kezelői felületek megvalósítását is lehetővé teszik. A csak karakteres felületet használó 4GL eszközök már elavultnak tekinthetők, és bár léteznek olyanok is (pl. Oracle Forms), melyek mindkét képernyő-kezelési módszert lehetővé teszik, ezekről is elmondható, hogy az alkalmazásfejlesztés grafikus felületen keresztül történik. Így a továbbiakban a karakteres felületek kezelését a grafikus felületek részhalmazának tekintjük, és nem foglalkozunk külön a karakteres felületekkel. A teljesség kedvéért még meg kell említenünk, hogy ez a megközelítésmódunk igaz az alkalmazásfejlesztés fázisára, nem igaz azonban az alkalmazás végrehajtására, ahol az eltérő hardver

eszközökből kifolyólag az alkalmazás futtatható kódjának természetesen eltérőnek kell lennie a két esetben. Ez azonban csak a kódgenerálást módosítja, ami jelenlegi vizsgálódásainkat nem érinti.

A kezelői felület megjelenítő objektumokból és parancsokból áll. A kezelő a megjelenítő objektumokon keresztül kaphat információkat a rendszertől, és rajtuk keresztül be is avatkozhat a rendszer működésébe. A parancsok parancsbillentyű kombinációk (pl. Ctrl+C) vagy menük. A szöveges parancsfelület létrehozását a 4GL eszközök közvetlenül nem támogatják, azonban az alapnyelvük segítségével természetesen ez is lehetséges.

Először a megjelenítő objektumok kezelésével foglalkozunk.

Első lépésként a megjelenítő objektumokat erre a célra kialakított speciális rajzolóprogramokkal, ún. *látványtervezőkkel* (*dialog editor, layout editor*) felrajzoljuk alkalmas felületekre, melyeket a továbbiakban *lapoknak* nevezünk. A látványtervező egyszerű módon biztosítja a megjelenítő objektumok mozgatását, átméretezését, másolását, törlését, és ezen kívül még számos hasznos művelet elvégzését.

A *lap* előre meghatározott méretű terület, melyet a látványtervező segítségével létrehozunk, és ez tartalmazza a kezelői felület alapelemeit. Egy alkalmazáson belül a kezelői felület elvileg tetszőleges számú lapot tartalmazhat. Grafikus környezetben a lapok általában egy-egy operációs rendszer szintű ablakban jelennek meg, karakteres környezetben azonban, ahol az operációs rendszer nem rendelkezik ablakozó felülettel, maguk a lapok viselkednek ablakokként. Ilyenkor az ablakozó felületet a 4GL eszköz futtató rendszere biztosíthatja.

A 4GL eszközök a kezelői felületnek a lapokon megjelenő objektumait osztálykönyvtárakban tárolják, melyek a fejlesztő számára eszköztárak (*toolbox, toolbar*) formájában jelennek meg a 4GL eszköz felületén. Az eszköztár ikonos nyomógombok csoportja, melyen belül minden nyomógomb egy-egy adott megjelenítő objektumféleség létrehozására szolgál. Maga a létrehozás nem igényel többet, mint az egér segítségével az eszköztárból a megfelelő nyomógomb kiválasztása, majd az új elem (objektum) helyének kijelölése az adott lapon.

Az objektumok - ahogy ezt tőlük el is várjuk - tulajdonságokkal és módszerekkel rendelkeznek. Minden létrehozott megjelenítő elemhez tartozik egy adatlap (*property sheet*), melynek segítségével az objektum tulajdonságai beállíthatók, ezáltal pedig az egyes objektumok (ezen keresztül az alkalmazás) viselkedése tág határok között befolyásolható. Az adatlapok formája a jelenleg forgalomban lévő 4GL eszközökben meglehetősen egységes. Az adatlap külön ablakban jelenik meg, melynek fő komponense az adott objektum tulajdonságaira vonatkozó, (tulajdonság, érték) párokból álló lista. A tulajdonságok értéke - többnyire közvetlenül, átírásssal - magán az adatlapon belül megváltoztatható.

A megjelenítő objektumokat az alkalmazás információfeldolgozási sémájához való kapcsolatuk alapján két csoportba sorolhatjuk:

Passzív objektumok: Tartalmuk, megjelenésük, és általában állapotuk is állandó. Kezelői eseményekre nem reagálnak, és adatokat nem jelenítenek meg. A passzív objektumok a kezelői felület olyan nélkülözhetetlen részei, melyek az egyes lapok áttekinthetőségét szolgálják, vagy emelik az alkalmazás fényét. Ebbe a csoportba tartoznak az egyszerű geometriai alakzatok (egyenes, kör, ellipszis, négyszög, sokszög, szabadkézi vonal), a feliratok és a statikus ábrák (pl. cégjelzés, háttér képek).

Aktív objektumok: Az alkalmazás információfeldolgozási folyamatának szerves részét képezik. Tartalmuk időben változó, és az általuk megjelenített adatokat az alkalmazáslogika képzi, vagy közvetlenül a környezetből származnak. Ha egy aktív objektum e mellett kezelői beavatkozásokra is reagál, *kezelőszervnek* (*control*) nevezik.

4.1.2 Szabványos aktív kezelőszervek

A következőkben összefoglalom a 4GL eszközökben szokásos aktív objektumok alapvető osztályait, és bemutatjuk a legelterjedtebben használt MS Windows ablakozó rendszerben szokásos megjelenésüket.

***Nyomógombok** (button)*

A nyomógomb olyan kezelőszerv, amely meghatározott műveletet vált ki, ha a felhasználó rákattint. Ilyen a gyakran előforduló **OK** nyomógomb, amely lezár egy párbeszédablakot, amikor a nyomógombra rákattintunk. Ha valaki nem látott volna még ilyet, így néz ki:

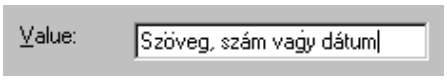
***Kapcsolók** (checkbox)*

A kapcsoló kétállapotú kezelőszerv: kikapcsolt vagy bekapcsolt állapotban lehet. A kapcsoló egyes állapotaihoz értékek rendelhetők, melyek a kapcsolón keresztül beállíthatók, és a kapcsoló az adott értéket megjelenítéskor az adott állapotával jelzi.

***Választógombok** (radio button)*

A választógombok egymást kölcsönösen kizáró választási lehetőségek halmazát jelenítik meg. A választógombok általában egy választógomb kerethez tartoznak, amely csoportokba fogja őket, és tárolja az aktuálisan kiválasztott lehetőségnek megfelelő értéket. A mellékelt ábra egy választógomb csoportra mutat példát:

Szövegmezők



A szövegmező olyan vezérlő, amely adatok megjelenítésére, közvetlen bevitelére és módosításra alkalmas. A szövegmező a környezetből származó, számított vagy a felhasználó által begépelt adatokat tartalmazhat. Minden olyan adattípus megjelenítésére alkalmas, mely szöveges formában ábrázolható, így karaktersorozatok, számok és dátumok. A következő ábra egy szövegmezőt mutat:

Listák

A listák adatok megjelenítésére képes mezők. A listákhoz adott értékek sorozata kapcsolódik, melyek közül választást tesznek lehetővé.

Alapvetően három fajtájuk szokásos: az állandó listák (T list), beugró listák (popup list) és a vegyes listák (combo box).

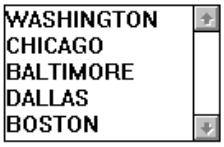
A listákhoz tartozó értéksorozatokat a listaobjektumok tulajdonságaként statikus módon, vagy programozottan lehet megadni. Az utóbbi esetben a felkínált értékek mindig a környezet aktuális állapotát tükrözik, más szóval a lista dinamikus.

A választó listáknak és a beugró listáknak két értékük van:

***Belső*:** A listamező értéke. Az alkalmazáslogika belső objektumai ezt az értéket látják, amikor a listamezővel műveleteket végeznek.

***Kijelzett*:** A listamezőben megjelenő érték.

Az egyes listamezők különleges tulajdonságai a következők:
Állandó listák



Az állandó lista segítségével a felhasználó adott értékek egy görgethető listájából választhat, de ezeken kívül más értékeket nem adhat meg. Az állandó lista folyamatosan mutat a felkínált elemek közül néhányat. Gyakori alkalmazása, mikor a felhasználónak meg akarjuk mutatni a választási lehetőségeket, miközben egy másik vezérlővel dolgozik. A mellékelt ábra egy állandó listát mutat:

Beugró listák

A beugró lista által felkínált értékek csak akkor jelennek meg, mikor a felhasználó a beugró lista vezérlőre kattint. Miután a lista megjelent, a felhasználó a vezérlőhöz

tartozó görgetősávval nézheti végig a felkínált elemeket.

A beugró lista az állandó listához hasonlóan a megadott elemekre korlátozza a felhasználó választási szabadságát. Állandó lista helyett gyakran beugró listákat alkalmazunk, ha az adott lapon kevés a hely. A mellékelt ábra beugró listát mutat (az értéklista éppen lezárt állapotban van).

Vegyes listák

A vegyes lista a beugró lista alternatívája, mely a listán felkínált értékek választásán kívül megengedi azt is, hogy a felhasználó a listamezőbe adatokat gépeljen. A vegyes lista olyankor használatos, mikor a felhasználónak javasolni szeretnénk néhány lehetséges értéket, de biztosítani akarjuk a felsorolásban nem szereplő értékek bevitelének lehetőségét is. A vegyes listák csak belső értékkel rendelkeznek, amely megegyezik a kijelzett értékkel, ugyanis a felhasználó által begépelt értékek esetén a kijelzett érték nem különíthető el a belső értéktől. Megjelenésében csak abban különbözik a beugró listától, hogy a lista megjelenítésére szolgáló nyilacska kissé külön válik a listamezőtől. A mellékelt ábra vegyes listát mutat.

Görgetősávok

A görgetősáv segítségével beállíthatjuk más alkalmazás objektumok értékét. Vízszintes és függőleges görgetősávokat is használhatunk. A mellékelt ábrán vízszintes görgetősáv látható:

Egyedi kezelőszervek

Bizonyos célrendszerek kialakítására szolgáló 4GL eszközök a fenti kezelőszerveken kívül számos egyéb megjelenítő objektumot és készen kínálhatnak. Ez különösen az ipari felügyelő rendszerek kialakítására szolgáló 4GL eszközökre

jellemző, ahol a legváltozatosabb technológiai folyamatok elemeit kell megfelelő módon kezelni. Ilyen környezetben szabványosnak mondható a toló- és tekerő potenciométerek, a különböző analóg, mutatós kijelzők, és számos egyéb elem (pl. tartályok, hőcserélők, reaktorok) megjelenítése.

Beviteli eszközök kezelése

A felhasználó a rendszer működésébe a két szabványos beviteli eszköz, a klaviatúra és az egér segítségével avatkozhat be. A beavatkozás meghatározott kezelőszervekhez kapcsolódó eseményeket vált ki, melyeket aztán az alkalmazáslogika kezel le. Ebben a pontban azokat a fogalmakat és eljárásokat ismertetem, melyeket a 4GL eszközök a klaviatúra és egér kezelésére általában beépített módon, készen kínálnak.

Mivel klaviatúrából és egérből csak egy van, kezelőszervből viszont tetszőlegesen sok lehet, az első probléma annak meghatározása, hogy az egér- vagy klaviatúraesemény a kezelői felület melyik objektumára vonatkozik. Ezt a problémát a 4GL eszközök (és általában minden grafikus kezelői felülettel rendelkező szoftverrendszer) a ***fókusz*** (*focus*) és az ***egérmutató*** (*mouse pointer*) fogalmának bevezetésével és használatával oldják meg:

*A **fókusz*** határozza meg, hogy (a parancsbillentyűket kivéve) a klaviatúra billentyűinek leütéséből származó események a kezelői felület melyik objektumára vonatkozzanak. A fókusz az alkalmazás használata során a kezelő igényeinek megfelelően vándorol objektumról objektumra. Mivel az egész rendszerben csak egy fókusz létezik, a fókuszt mindig egy és csak egy elem birtokolja, melynek megjelenését a fókusz kis mértékben módosítja. A fókuszt szövegmezőkben és vegyes listákban egy kurzor (általában villogó, függőleges vonal) mutatja, míg gombokon, választógombokon vagy kapcsolókon vékony, szaggatott keret jelzi.

*Az **egérmutató*** határozza meg, hogy az egér gombjainak megnyomásából származó események a kezelői felület melyik objektumára vonatkozzanak.

A kezelői felület lényeges eleme a fókusz és az egérmutató mozgatása. Az egérmutató az egér mozgatásával szinkronban változtatja a helyét a képernyőn. A fókusz mozgása ennél jóval bonyolultabb mechanizmus szerint történik, melyet *navigáció* néven szokás emlegetni. A továbbiakban a navigáció kérdésével részletesebben foglakozunk.

A navigáció eszköze lehet az egér vagy a klaviatúra is.

A navigáció legtermészetesebb módon egérrel végezhető. Ehhez az egérmutatót a kívánt elem fölé visszük, majd az egyik (többnyire a bal) egérgombra kattintunk, melynek hatására (szerencsés esetben) a fókusz a régi helyéről az új elemre vándorol.

Ahhoz, hogy a klaviatúráról is tudjunk navigálni, külön e célra szolgáló, speciális parancsbillentyűket vagy parancsbillentyű-kombinációkat kell definiálnunk. Ily módon, a klaviatúránkon lesznek majd olyan billentyűk, melyek a fókuszt birtokló elem alapvető funkcióinak kiváltására valók (pl. nyomógomb megnyomása általában az `Enter` billentyűvel, listaelem kiválasztása általában a le-föl nyilakkal, szövegmező tartalmának változtatása pedig az alfanumerikus karakterekkel, valamint a törlés és jobbra-balra nyilakkal lehetséges). E mellett lesznek olyan billentyűink, amelyek a fókusz változtatására szolgálnak (pl. általában ilyenek a `Tab`, a `Shift+Tab` és a `Ctrl+Tab` billentyűk)

4.1.3 Kommunikációs felület

A kommunikációs felületen keresztül tartja a rendszerünk a kapcsolatot a gépi környezettel. A kommunikációs felület tehát egyfajta adatcserét és szinkronizációt kell, biztosítson külső alkalmazásokkal.

A külső alkalmazás mindig olyan komplex, gyakran általános részfeladatot old meg, melynek megvalósítása nem a mi alkalmazásunk feladata. Ebből a szempontból - a teljesség igénye nélkül - a következő öt fontos alapesetet különböztetjük meg:

1. Operációs rendszer szintű szolgáltatásokhoz akarunk hozzáférni, mint pl. állományok kezelése, nyomtatás, hálózati kommunikáció.

2. Egy (vagy több) adatbázishoz szeretnénk hozzáférni, és a bennük tárolt adatokat lekérdezni, módosítani, és az adatbázisokba újabb sorokat bevinni.

3. Létező, komplex alkalmazások (pl. hálózati böngészők, szövegszerkesztők, táblázatkezelők) szolgáltatásainak igénybevételével nézegetnénk, szerkesztenénk és tárolnánk dokumentumokat.

4. Méréseket szeretnénk elvégezni, a mérések eredményeit megjeleníteni, feldolgozni, és tárolni.

5. Komplex ipari rendszert akarnánk az alkalmazásunkon keresztül felügyelni és/vagy irányítani.

Bár komplexitásukat és szemléletmódjukat tekintve ezek az esetek rendkívül eltérőek, azért mégis van egy közös vonásuk: **nem jöhet szóba, hogy a külső rendszer funkcióit az alkalmazásunkon belül, mi magunk valósítsuk meg.** Egyrészt nyilvánvalóan nem állhatunk neki olyan - esetenként több száz emberévnyi munkát igénylő - szoftverrendszerek megvalósításának, mint amilyenek az operációs rendszerek, adatbázis-kezelők, szövegszerkesztők vagy Web böngészők, nem beszélve a komplex mérőrendszerek és ipari technológiák kialakításáról, ahol már csak a dolog hardver vonzatai és a kialakításukhoz szükséges rendkívül összetett speciális tudás hiánya miatt sem vehetjük föl a kesztyűt. Másrészt a működő, kipróbált rendszerek reprodukciója kifejezetten káros. A fejlesztők számára azért, mert feleslegesen dolgoznak, a felhasználók számára pedig azért, mert a megszokott, szabványos felületük (pl.Word) helyett biztos, hogy kicsit (vagy éppenséggel nagyon) mást kapnak, amitől kényelmetlenül érzik magukat, és bennünket szidnak majd.

4.1.4 Alkalmazáslogika

Az alkalmazáslogika valósítja meg az alkalmazásunk procedurális belső működéseit. A végrehajtandó tevékenységeket két általános csoportba sorolhatjuk.

Eseménykezelés

Ezek, a kezelői felületről és a kommunikációs felületről érkező események kiszolgálásával kapcsolatos tevékenységek. A felületen megjelenő előredefiniált elemek száma bármely 4GL környezetben adott, és így definiálható azon események köre, melyek ezekhez, a felületi elemekhez kapcsolódnak.

Alkalmazásspecifikus tevékenységek

Olyan egyedi, általunk megírt kódrészletek ezek, melyek az alkalmazás igényeinknek megfelelő működését eredményezik.

A kétfajta tevékenység a következő módon kapcsolódik egymáshoz. A 4GL eszközök az általuk ismert események körét előre definiálják, és kezelésükre egy (remélhetőleg) jól dokumentált, előre gyártott sémát, lényegében egy kész alkalmazásvázat kínálnak. Az események kezeléséhez az alkalmazások egyes objektumainak megfelelő módszerei kapcsolódnak, melyeket a 4GL eszköz futtatórendszere a megfelelő esemény bekövetkeztekor automatikusan végrehajt. Mivel, ahogy azt jól tudjuk, a módszerek osztályszintű kódrészletek, így a 4GL eszközökben az egyes objektumokhoz a módszereken keresztül számos előredefiniált tevékenység tartozik (pl. ellenőrzések, navigációs tevékenységek, egérkezelés, tranzakciókezelés). Ennek következményeként egyszerűbb, tipikus feladatokat megoldó alkalmazások akár egyetlen programsor megírása nélkül, pusztán a megfelelő objektumok létrehozásával, és a hozzájuk kapcsolódó beépített tevékenységek kihasználásával megvalósíthatók.

Az alkalmazásspecifikus tevékenységeket ezek után a megfelelő objektumok megfelelő módszereinek törzsében elhelyezett - gyakran mindössze néhány soros -

kódrészletekkel valósítjuk meg. Ez a módszer rendkívül hatékony programfejlesztést tesz lehetővé, de a hagyományos programnyelvekhez képest eltérő, és sok programfejlesztő által kevéssé preferált szemléletmódot igényel. A hagyományos szemléletű fejlesztés során a programozó egy virtuális gépen futó, egybefüggő programkódot ír, így a történésekkel teljes egészében tisztában van. Ezzel szemben a 4GL eszközök esetében a saját kód elhelyezésére csak belépési pontokat kapunk, miközben a 4GL ütemező működése sok esetben homályba vész, és apró részleteiről csak a dokumentáció meglehetősen fáradságos és aprólékos tanulmányozása (lényegében folyamatábrák visszafejtése) során kaphatunk precíz információkat, holott az alkalmazás működése szempontjából az egyes események feldolgozásának és a feldolgozó módszerek aktivizálásának sorrendje és feltételei alapvetően fontosak. Ezért a legkomolyabb problémát a 4GL fejlesztés során általában nem is annyira a megfelelő kódrészletek megírása, hanem a kódrészletek végrehajtatására alkalmas módszerek, megkeresése jelenti.

Az egyedi tevékenységek megvalósítása során a 4GL eszközök lehetővé teszik a programozó számára, hogy a rendszer beépített működéseit kiegészítse vagy egyszerűen, felüldefiniálja. Az utóbbi esetben a 4GL eszköz által nyújtott alapviselkedés teljes egészében elfedhető, és igényeinkhez alakítható.

Érdemes megfigyelni, hogy a 4GL eszközök az objektum fogalmát az objektumorientált nyelvekhez képest jelentős eltéréssel kezelik. Az objektumorientált nyelvekben az objektum végrehajtható programegység saját munkaterülettel, mely a rajta végrehajtható műveleteket osztályának módszereiként örökli. Más szóval nemigen szokásos egy objektumhoz saját módszereket rendelni. Mindez egy 4GL eszköz esetében teljesen megszokott dolog, ugyanis az egyedi objektumok (pl. kezelőszervek) logikus módon egyedi bánásmódot igényelnek. Ez egyébként az életünket egyszerűsíti, ugyanis egyébként minden egyedi tevékenység megvalósításához egy-egy újabb osztályt kellene létrehoznunk, melyhez aztán többnyire egyetlen egyedet generálnánk. Ennek a filozófiának a következménye, hogy osztályok létrehozására 4GL eszközökben gyakran alkalmazott módszer az ún. ***delegálás***, melynek lényege, hogy egy objektum mintájára hozunk létre egy osztályt, felhasználva az objektum tulajdonságait és a hozzá kapcsolódó módszereket.

5. A megvalósítandó feladat leírása

A diplomamunka témájául választott alkalmazás a Püspökladányi Polgármesteri Hivatal szociális munkájával kapcsolatos dolgokat igyekszik megoldani. Az alkalmazást próbáltam úgy elkészíteni, hogy a munka elvégzése minél egyszerűbb legyen. A szociális osztályon dolgozókkal történő többszöri megbeszélés és egyeztetés után egyre pontosabb képet kaptam arról, hogy mire is lenne igazán szükségük a munkamenet hatékonyságának a növeléséhez.
Az alábbiakban következnek az elvárások, amiknek meg kell felelnie az általam készített alkalmazásnak:

5. 1 A felhasználói elvárások:

Az osztályvezetőnek:

- Kapjon lehetőséget arra, hogy ha szükséges, tudjon új ügyintézőt felvinni, illetve eltávolítani a rendszerből.
- Módosíthatja a segélye adatait, illetve új segélyfajtát vihet fel a rendszerbe.
- Meg tudja tekinteni, hogy az egyes ügyintézők kiknek és mekkora összegű segélyt folyósítottak.
- Kapjon lehetőséget arra, hogy havi kimutatásokat készítsen arról, hogy az adott hónapban kinek mekkora összegű segély került folyósításra az adott időszakban.

A rögzítőnek:

- Az alkalmazás tegye lehetővé az ügyfelekkel kapcsolatos adatok bevitelét és módosítását
- Biztosítsa az ügyfél számára, hogy megtekinthesse a saját adatait, azt, hogy milyen segélyek kerültek neki kiutalásra és mikor, és ha van még neki kiutalandó segély, akkor mi a folyósítás várható dátuma.

Az elbírálónak

- Az alkalmazás tegye lehetővé, hogy megtekintse, hogy az adott ügyfél mikor és mekkora összegű segélyben részesült.
- Ha az ügyfél már kapott szociális juttatást, akkor az elbíráló tudja megmondani, hogy kaphat-e újra segélyt, és ha igen akkor milyen fajtát.
- Képes legyen nyomtani, egy olyan dokumentumot, amelyben benne vannak az ügyfél adatai és, hogy mikor és mekkora összegű segélyt folyósít neki a Polgármesteri Hivatal.
- Legyen jogosult arra, hogy eldöntse, jogos-e a segély igénylés és az ő jóváhagyásával kaphassa meg az ügyfél a kért támogatást.

Az ügyfélnek:

- Az ügyfél, ha igénye van, rá megtekintheti, a személyes adatait ehhez azonosításra lesz szükség. (A nevére és az adó vagy TAJ számára). A megfelelő azonosítás után elérhetővé válnak a személyes adatai és, hogy milyen segélyek kerültek, illetve kerülnek folyósításra számára.
- Az ügyfél értesítést kap a segélyigénylésének az elbírálásáról, akár támogatva akár elutasítva lett.

Export/import

Az alkalmazás biztosítson lehetőséget az adatbázisban tárolt adatok időnkénti lementésére és visszatöltésére. Ezért a tevékenységért a rendszergazda a felelős. Ő bizonyos időközönként rendszeresen mentést készít az adatbázisban tárolt adatokról így esetleges hiba esetén lehetséges az adatbázisban tárolt adatok visszaállítása egy korábbi állapotra.

6. A logikai tervezés

A programot a következő felhasználók indíthatják el: rögzítő, elbíráló, osztályvezető. A regisztrált ügyfelek adatai mentve vannak, így azokat meg lehet tekinteni. Az alkalmazást igyekeztem úgy elkészíteni, hogy mindenki számára hatékony és egyszerű legyen.

A rögzítők számára az igénylők és segélyfajták egyszerű adminisztrációját szeretném biztosítani. A igénylők felvitelét és a segélyfajták adatainak a módosítását ők végzik. Az adatokat könnyen kezelhető grafikus felületen tudják felvinni. Lehetőségük van az igénylő adatainak a megtekintésére és módosítására. Ők nyújtanak segítséget az igénylőknek a saját adatok nyújtásában.

Az elbírálók számára lehetőséget nyújt a segély igénylések feldolgozására.

Az osztályvezetőnek információkat nyújt a segélyigénylés különböző állapotairól.

Az alkalmazás a következő funkciókat biztosítja miután bejelentkezett a felhasználó:

6. 1 Rögzítő lehetőségei:

- Igénylő adatainak a megtekintése és módosítása
- Alkalmazottak felvitele a rendszerbe

6.2 Elbíráló lehetőségei:

- Megtekinteni a segélyigényléseket
- Eldönteni, hogy a segélyigénylés támogatott vagy elutasított.

6.3 Osztályvezető lehetőségei:

- Neki joga van, hogy az összes tevékenységet ellássa.
- Segély adatainak a módosítása, új segélyfajták felvitele.
- Kimutatások megtekintése az igénylőkről, segélyigénylésekről és elbírált igénylésekről.

6.4 Ügyfél lehetőségei:

- A megfelelő azonosítás után lehetősége van a saját adatainak a megtekintésére.

6.5 Az adatbázis ismertetése

6.5.1 Az adatbázis táblái

- Igénylő
- Gyermekek
- Segélyfajták
- Segélyigénylések
- Elbírálva
- Folyósított juttatások
- Ügyintézők

Az adatbázis alapjául szolgáló adatbázis táblák és a közöttük lévő kapcsolatok az alábbiakban láthatóak. Ahhoz, hogy könnyen érthetőek legyenek a táblák néhány attribútum magyarázatát megadom, hogy miért szükséges a táblában és milyen szerepet tölt be.

6.5.2 Alapfogalmak:

Igénylő táblában:

- IGENYLO_AZON: ez a tábla elsődleges kulcsa így egyértelműen azonosítja az igénylőt.
- ALLAMPOLGARSAGA: az az állam, ahol letelepedési engedélyt kapott, megtalálható az igénylő személyi azonosságát igazoló igazolványban.
- TELEPULES: az igénylő állandó lakcímének városa.
- KOZTERULET: az utca, ahol az igénylő lakik
- TAJ_SZAM: az Országos Egészségbiztosítási Pénztár által minden társadalombiztosítással személy részére kiadott biztosítási kártya 9 számjegyből álló egyedi kódja. (Társadalombiztosítási azonosító jel) Az egészségügyi vizsgálatok során ez szolgál a betegek azonosítására.
- IGENYLO_BANKJANAK_NEVE
- IGENYLO_BANKJANAK_SZAMLASZAMA
- IGENYLO_SZAMLASZAMA

Ezek az attribútumok tárolják a banki átutaláshoz szükséges információkat.

A táblában a telefonszámon, az e-mail címen és a banki átutalással kapcsolatos adatokon kívül az összes adat megadása kötelező.

Gyermekek táblában:

- GYERMEK_AZON: a tábla elsődleges kulcsa, így egyértelműen azonosítja a gyermeket.
- IGENYLO_AZON: külső kulcs, amivel biztosított, hogy elérhetővé váljon a szülő (igénylő).

A gyermekek táblát azért hoztam létre külön, hogy csökkentsem az adatbázisban lévő redundanciát, és így erőforrás hatékonyabban tudom letárolni az egyes szülőkhöz tartozó gyermekeket.

Ügyintézők táblában:

- UGYINTEZO_AZON: a tábla elsődleges kulcsa, így egyértelműen azonosítja az ügyintézőket.
- BEOSZTASA: az adott ügyintéző milyen beosztású. Két értéket vehet fel: rögzítő és elbíráló

Segélyfajták táblában:

- SEGELYFAJTA_AZON: a tábla elsődleges kulcsa, így egyértelműen azonosítja a segélyfajtákat.
- SEGELY_OSSZEGE: az adott segélyfajtához tartozó kiutalható összeg
- KIUTALAS_GYAKORISAGA: a segély milyen gyakorisággal kerül kiutalásra. A lehetséges értékei: egyszeri, hetenkénti, havonkénti.
- OREGSEGI_NYUGDIJ_MIN_OSSZEGE: a mindenkori öregségi nyugdíj értéke. Egyes segélyfajtákhoz tartozó értékek ebből kerülnek kiszámításra
- SZAZALEK_ERTEK: az öregségi nyugdíj hány százaléka tartozik a segélyhez

Segélyigénylések táblában:

- SEGELY_IGENYLES_AZON: a tábla elsődleges kulcsa, így egyértelműen azonosítja az adott segélyigénylést.
- IGENYLO_AZON: külső kulcs, hogy elérhetővé váljanak az igénylő adatai.
- UGYINTEZO_AZON: külső kulcs az ügyintézőhöz. Így nyomon lehet követni, hogy ki rögzítette az igénylést.
- SEGEALYFAJTA_AZON: külső kulcs, a segélyfajtákhoz. A segély paramétereit érem így el.
- IGENYELT_OSSZEG: az igénylő itt adja meg a rögzítőnek, hogy mekkora összegű segélyt szeretne kapni.
- DATUM: az igény rögzítésének az időpontja.
- IGENYLES_OKA: az igénylő itt adhatja meg, hogy milyen céllal folyamodik a segélyre. Az elbíráló ez alapján, és hogy az igénylő milyen támogatásban részesült már dönt a későbbiekben, hogy jogosult-e a segélyre.

- IGENYLES_ALLAPOTA: két értéket vehet fel ’TAMOGATVA’ és ’ELUTASITVA’. Az érték létrehozáskor folyamatban, ami akkor módosul, ha az elbíráló foglalkozott az adott segélyigényléssel.

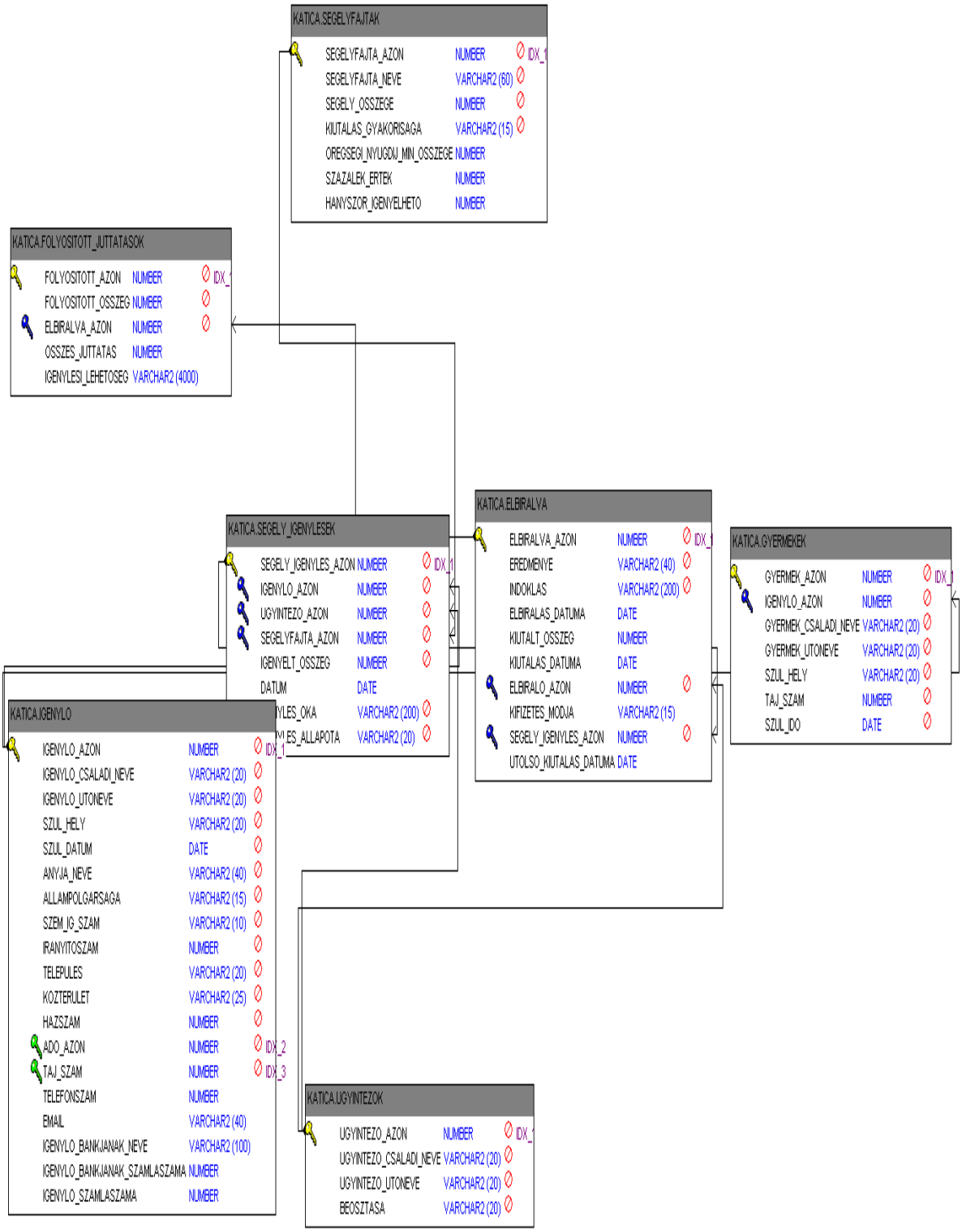
Elbírálva táblában:

- ELBIRALVA_AZON: a tábla elsődleges kulcsa, így egyértelműen azonosítja az adott elbírálandó ügyet.
- EREDMENYE: az elbíráló döntése az igénylő adatai és az igénylő indoklásának függvényében. A lehetséges értékei: ’TAMOGATVA’ és ’ELUTASITVA’.
- INDOKLAS: az elbíráló itt indokolja meg, miként jutott a döntésére.
- ELBIRALAS_DATUMA: az elbírálás készítésének az időpontja.
- KIUTALT_OSSZEG: mekkora mértékű támogatás került megállapításra.
- KIUTALAS_DATUMA: az igénylő mikortól számíthat a segély folyósítására.
- UGYINTEZO_AZON: Így nyomon lehet követni, hogy ki végezte a segélyigénylés elbírálását.
- KIFIZETES_MODJA: a megállapításra kerülő összeg milyen formában jut el az igénylőhöz. Lehetséges módjai: banki átutalás, készpénzben történő kifizetés.
- SEGELY_IGENYLES_AZON: külső kulcs a segélyigénylés táblához.
- UTOLSO_KIUTALAS_DATUMA: itt kaphatunk információt, hogy mikor volt az utolsó kiutalás az ügyfélnek.

Folyósított juttatások:

- FOLYOSITOTT_AZON: a tábla elsődleges kulcsa, így egyértelműen azonosítja az egyes folyósított juttatásokat.
- FOLYOSITOTT_OSSZEG: az elbíráló által meghatározott összeg. Ennek az értékét az Elbírálva táblából veszem.

Ez a tábla adatrögzítő funkciót lát. A célja, hogy a kimutatások elkészítéséhez nyújtson segítséget.


KATICA.SEGELYFAJTAK
SEGELYFAJTA_AZON NUMBER IDX_1
SEGELYFAJTA_NEVE VARCHAR2 (60)
SEGELY_OSSZEGE NUMBER
KIUTALAS_GYAKORISAGA VARCHAR2 (15)
OREGSEGI_NYUGDIJ_MIN_OSSZEGE NUMBER
SZAZALEK_ERTEK NUMBER
HANYSZOR_IGENYELHETO NUMBER
KATICA.FOLYOSITOTT_JUTTATASOK
FOLYOSITOTT_AZON NUMBER IDX_1
FOLYOSITOTT_OSSZEG NUMBER
ELBIRALVA_AZON NUMBER
OSSZES_JUTTATAS NUMBER
IGENYLESI_LEHETOSEG VARCHAR2 (4000)
KATICA.ELBIRALVA
ELBIRALVA_AZON NUMBER IDX_1
EREDMENYE VARCHAR2 (40)
INDOKLAS VARCHAR2 (200)
ELBIRALAS_DATUMA DATE
KIUTALT_OSSZEG NUMBER
KIUTALAS_DATUMA DATE
ELBIRALO_AZON NUMBER
KIFIZETES_MODJA VARCHAR2 (15)
SEGELY_IGENYLES_AZON NUMBER
UTOLSO_KIUTALAS_DATUMA DATE
KATICA.SEGELY_IGENYLESEK
SEGELY_IGENYLES_AZON NUMBER IDX_1
IGENYLO_AZON NUMBER
UGYINTEZO_AZON NUMBER
SEGELYFAJTA_AZON NUMBER
IGENYELT_OSSZEG NUMBER
DATUM DATE
VARCHAR2 (200)
VARCHAR2 (20)
KATICA.GYERMEKEK
GYERMEK_AZON NUMBER IDX_1
IGENYLO_AZON NUMBER
GYERMEK_CSALADI_NEVE VARCHAR2 (20)
GYERMEK_UTONEVE VARCHAR2 (20)
SZUL_HELY VARCHAR2 (20)
TAJ_SZAM NUMBER
SZUL_IDO DATE
KATICA.IGENYLO
IGENYLO_AZON NUMBER IDX_1
IGENYLO_CSALADI_NEVE VARCHAR2 (20)
IGENYLO_UTONEVE VARCHAR2 (20)
SZUL_HELY VARCHAR2 (20)
SZUL_DATUM DATE
ANYJA_NEVE VARCHAR2 (40)
ALLAMPOLGARSAGA VARCHAR2 (15)
SZEM_IG_SZAM VARCHAR2 (10)
IRANYITOSZAM NUMBER
TELEPULES VARCHAR2 (20)
KOZTERULET VARCHAR2 (25)
HAZSZAM NUMBER
ADO_AZON NUMBER IDX_2
TAJ_SZAM NUMBER IDX_3
TELEFONSZAM NUMBER
EMAIL VARCHAR2 (40)
IGENYLO_BANKJANAK_NEVE VARCHAR2 (100)
IGENYLO_BANKJANAK_SZAMLASZAMA NUMBER
IGENYLO_SZAMLASZAMA NUMBER
KATICA.UGYINTEZOK
UGYINTEZO_AZON NUMBER IDX_1
UGYINTEZO_CSALADI_NEVE VARCHAR2 (20)
UGYINTEZO_UTONEVE VARCHAR2 (20)
BEOSZTASA VARCHAR2 (20)

7. Felhasználói leírás

7.1 A rendszer telepítése

A szükséges alkalmazások:

- Operációs rendszer: Windows NT, 2000, XP
- Adatbázis: Oracle 10g
- Form Builder
- Report Builder

Ahhoz, hogy az alkalmazás futtatható legyen szükséges, hogy a fent említett alkalmazások már telepítve legyenek a számítógépen. Csak ezen programok telepítése után kerülhet sor az alkalmazás telepítésére.

7.2 Saját rendszer telepítése:

- Készítettem egy futtatható állományt, ami a CD-n található ezt kell lefuttatni. Ekkor megtörténnek az alkalmazáshoz szükséges állományok másolása a C:\ -re

Adatbázis beállítása:

- A rendszergazdának futtatni kell a *hivatal* könyvtárban található felhasznalo_letrehozasa.sql scriptet, ami a rendszer katica nevű felhasználóját hozza létre.
- Az SQL*Plus segítségével katica felhasználóként csatlakozni kell az adatbázishoz és futtatni a tábla létrehozo.sql és a tabla_feltolto.sql scripteket.

7.3 A program elindítása:

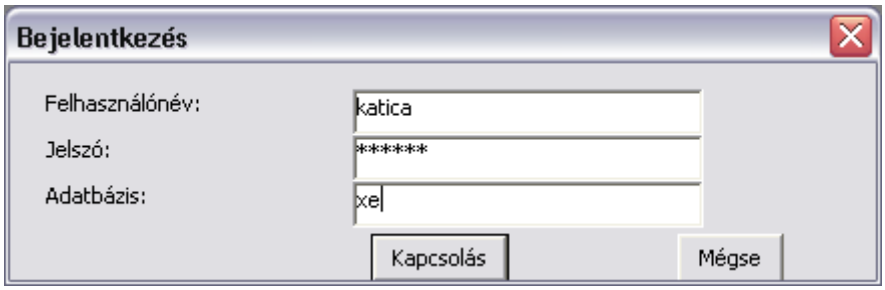
- Az alkalmazás futtatásához létrehoztam egy batch fájlt indit.bat néven, ez megtalálható a CD *hivatal* könyvtárában.

8. Az alkalmazás használata

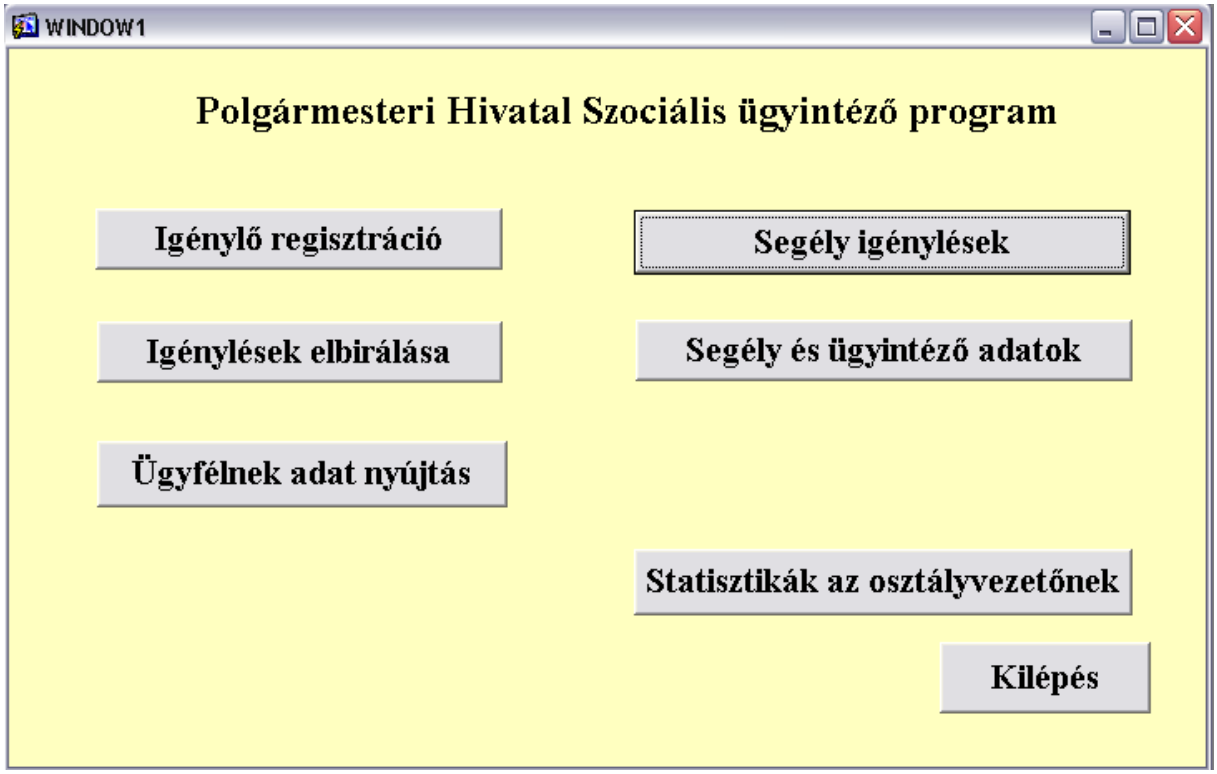
8.1 Bejelentkezés

A felhasználónak először be kell jelentkeznie mielőtt bármit, tenne.

A program a következő ablakkal indul:



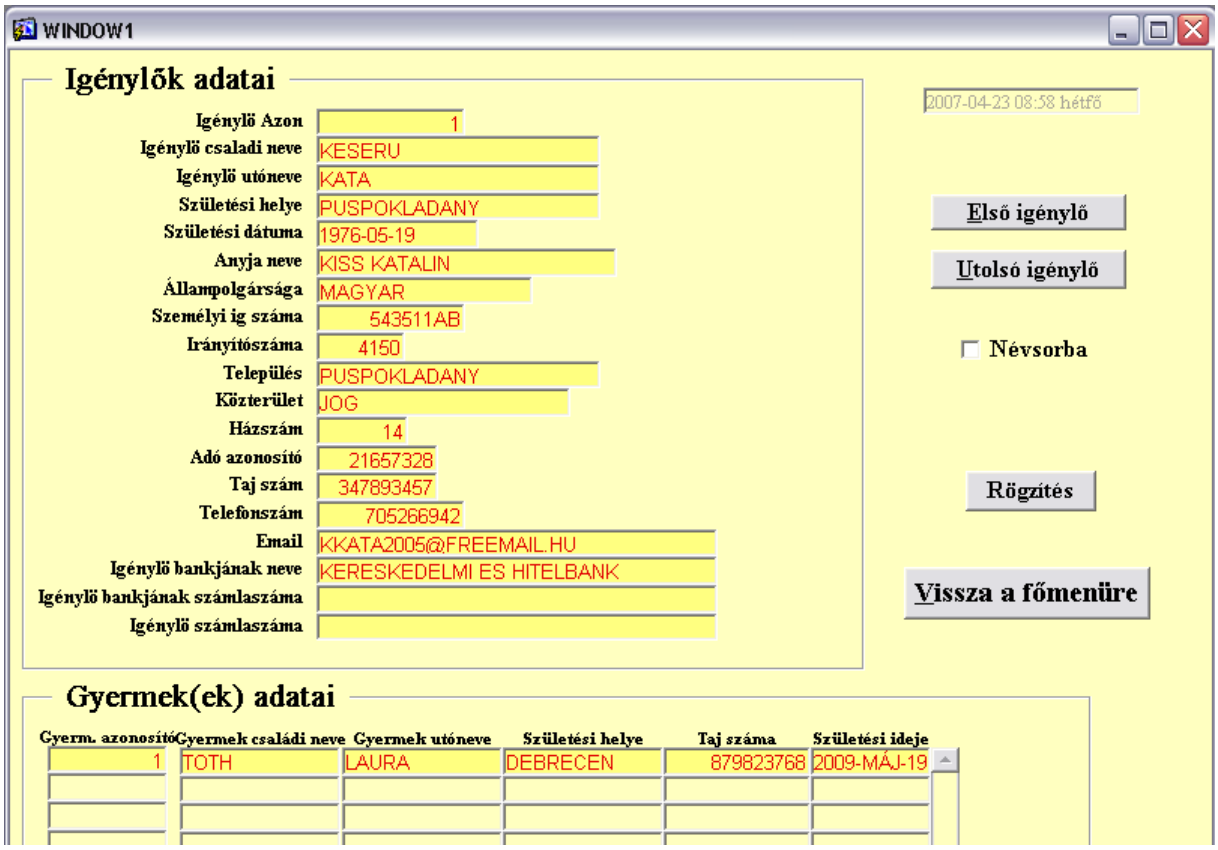
A bejelentkező ablaktól jutunk el a program főmenüjébe, ami a következőképpen néz ki:



A főmenüt igyekeztem úgy elkészíteni, hogy egy egységes felületet képezzen a szociális osztály különböző folyamatainak. Innen érhetőek el a különféle funkciók.

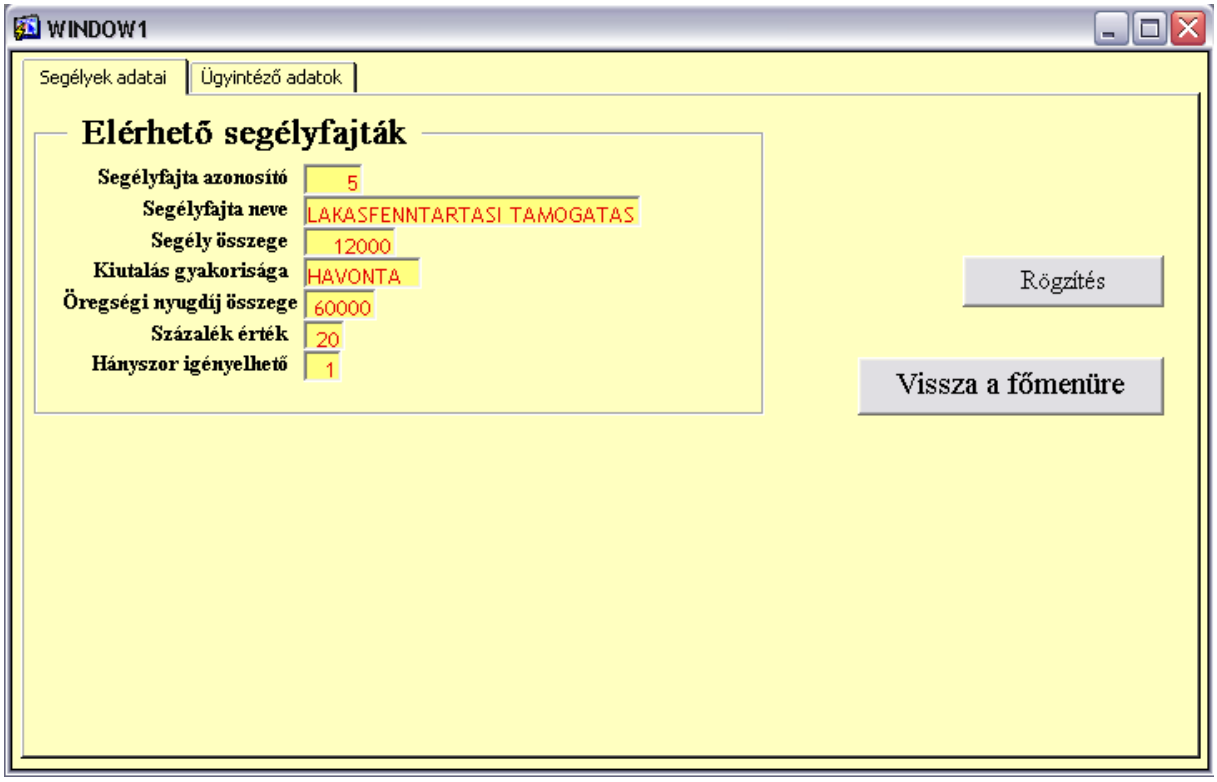
8.2 A rögzítő lehetőségei:

A rögzítő ezekből a funkciókból az igénylő regisztrációt és a segély ügyintéző adatokat használja. Az **„igénylő regisztráció”** gombra kattintva éri el a rögzítő a következő felületet. Itt van lehetőség új igénylők felvitelére vagy a már korábban regisztrált ügyfelek adatainak a módosítására, ha szükséges.

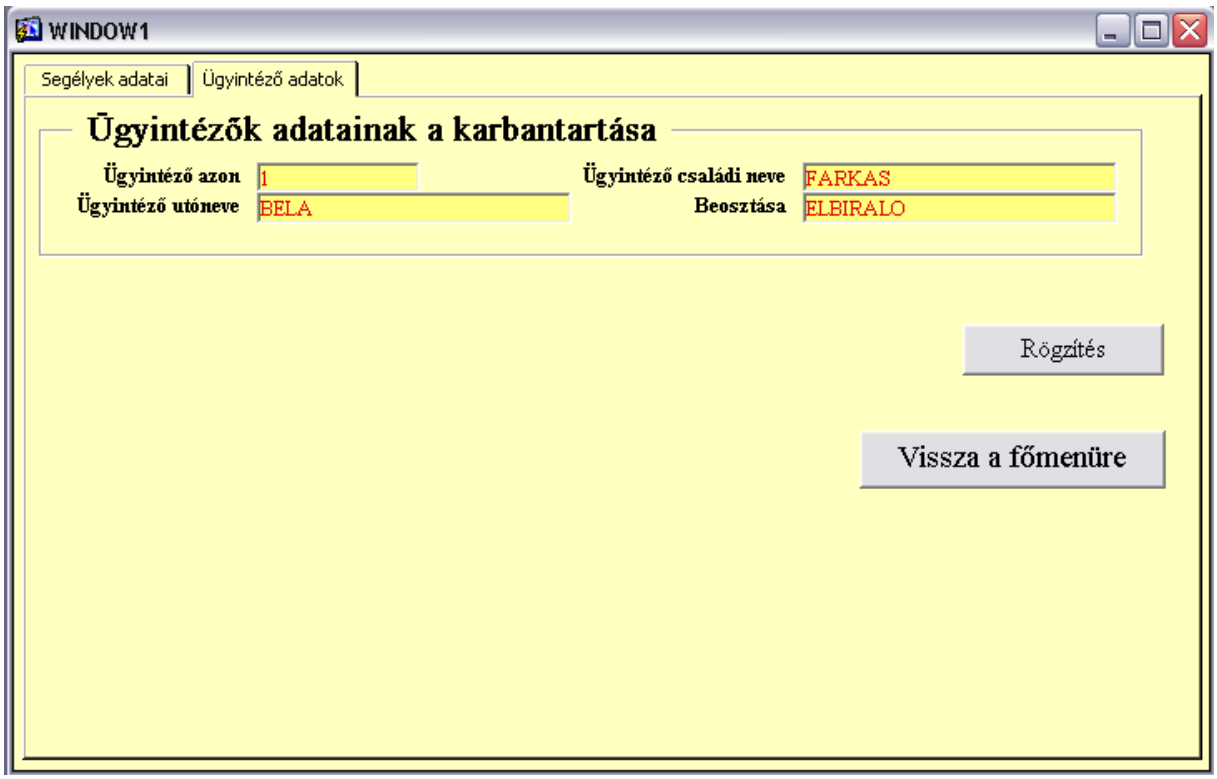


Mint ahogy a képen látható több funkciót nyújt a felület. A rögzítő a **„vissza a főmenüre”** gombra kattintva jut vissza a főmenüre.

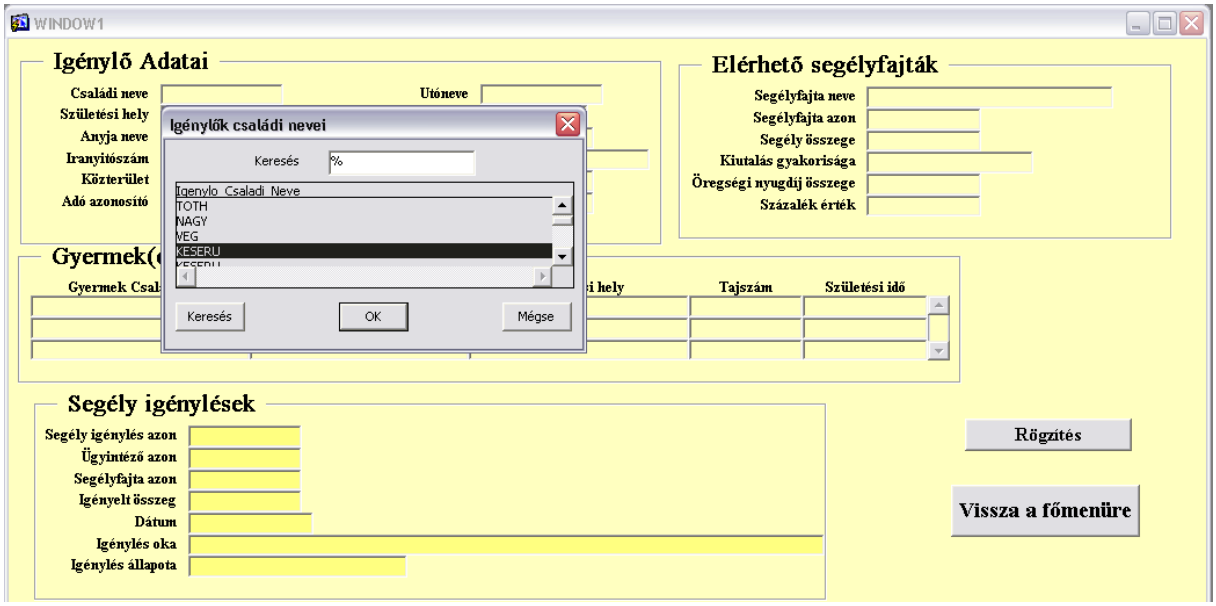
A másik főmenüből elérhető gomb a **„segély és ügyintéző”** adatok. Itt fülesvásznas megoldást választottam. Az egyik fülön érjük el a rendelkezésre álló segélyfajtákat, ahol megtekinthetjük az adott segélyfajta adatait, módosíthatjuk őket, illetve új segélyfajtát vihetünk fel. A másik fülön pedig a dolgozók adatai érhetőek el illetve módosíthatóak. A segélyfajták adatait elérő felület a következőképpen néz ki:



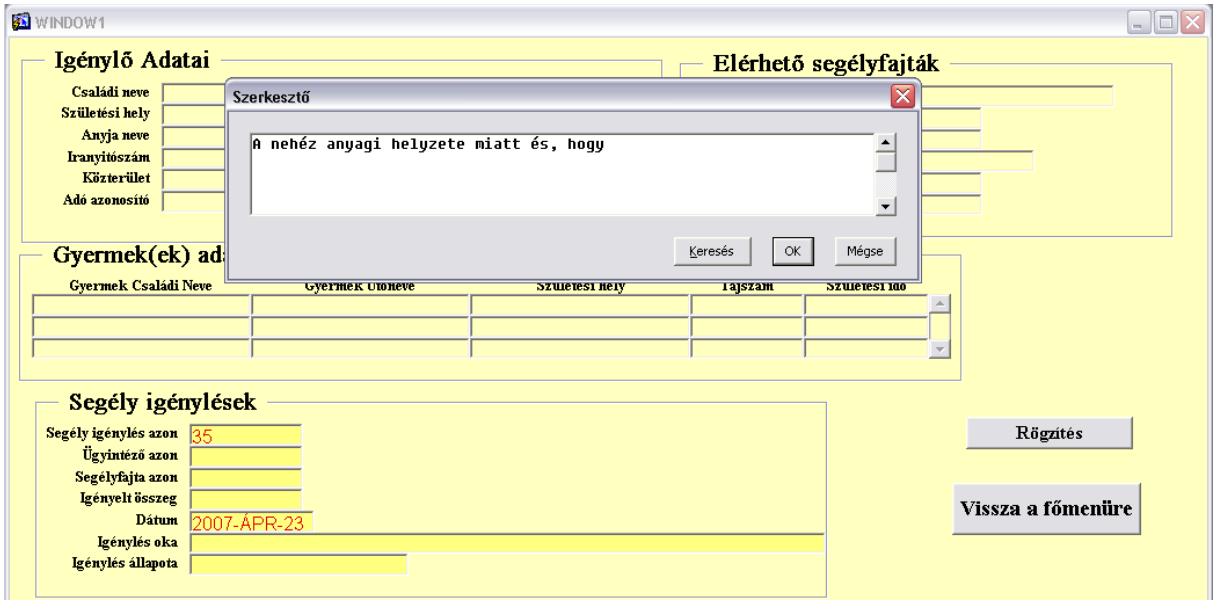
Az alkalmazottak adatait kezelő pedig a következőképpen:



A főmenüről éri el a rögzítő a **„segélyigénylés”** gombbal azt a felületet, ahol a tényleges segélyigénylés történik. A felület szolgál a segélyigénylés adminisztrálására. Mint, ahogy korábban már említettem, csak regisztrált személyek jogosultak segély igénylésére. A felület a következőképpen néz ki:



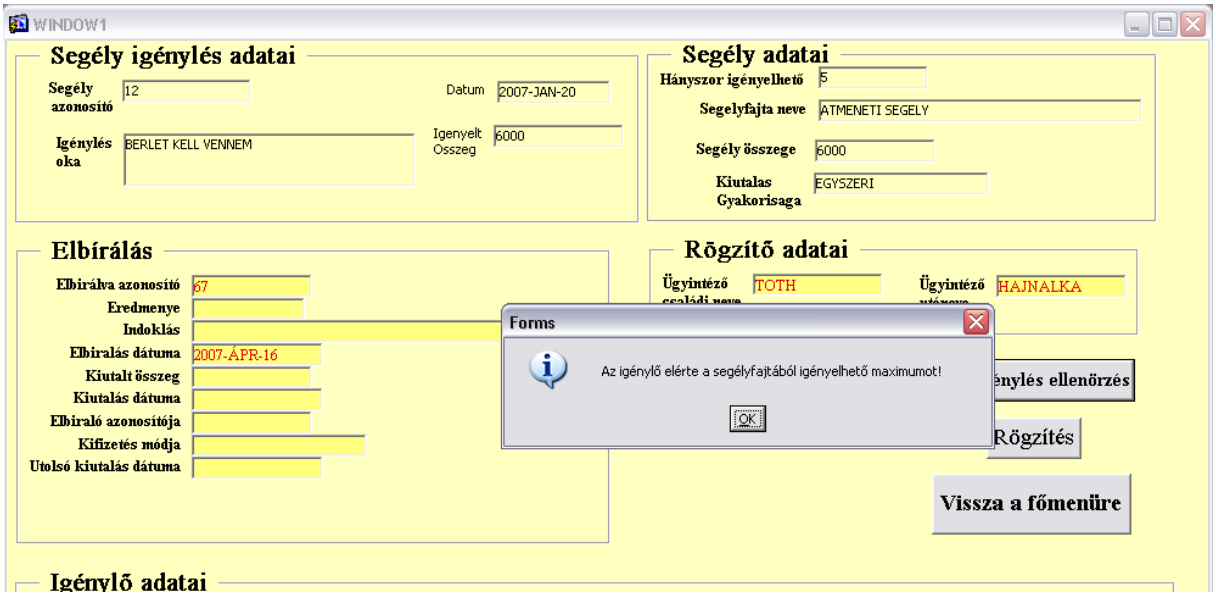
Mint ahogy a képen is látszik igyekeztem megkönnyíteni a rögzítő munkáját az adatok bevitelével. A hosszabb adatok szerkesztéséhez a beépített szövegszerkesztő hívható segítségül, amit a következő képpen lehet látni:



8.3 Az elbíráló lehetőségei:

Az elbíráló a rögzítő által felvitt segélyigényléseket bírálja el. A döntése attól függően, hogy az ügyfél jogosult lehet támogatott, illetve elutasított. A rendszer automatikusan ellenörzi, hogy az ügyfél az adott segélyfajtát igénybe veheti e. Erre tárolt eljárást készítettem. Az eljárás azt ellenőrzi, hogy az adott igénylő az adott segélyfajtából elérte e már a maximálisan igénybe vehetőt, azaz az elbírálva táblában az ő azonosítójával van-e annyi támogatott segély igénylés mint az adott segélyfajtából a maximum. Ha egy segélyigénylést elbírál az elbíráló akkor a segély_igénylés_allapota automatikusan 'FOLYAMATBAN'-levőről átvált 'ELBIRALVA' állapotba. Ezt is a beépített tárolt eljárás végzi. Az elbíráló felülete a következően néz ki:

Ha az ügyfél az adott segélyfajtát nem veheti igénybe arról, a program egy ablakban, tájékoztat, ami a kővetkező képen látható:



8.4 Az osztályvezető lehetőségei:

Az osztályvezető a főmenüről éri el azokat az adatokat, amit az ő kimutatásaihoz felhasználhat. Lehetősége van információkat megtudni az elbírált igénylésekről. Az osztályvezető a **„statisztikák az osztályvezetőnek”** gombra kattintva érheti el azt a felületet, amiről közvetlenül választhatja ki a neki szükséges információkat.

A felület a következőképpen néz ki:

Az osztályvezető a gombok, megnyomásával juthat el a riportokhoz. Amikre a teljesség igénye nélkül bemutatok néhányat az alábbiakban:

Igénylők listája

Report run on: Április 24, 2007 9:45 DU.

Azonosító	Családi név	Utónév	Születési hely	Születési dátum	Anyja neve	Szem. ig. szá	Település	Adó azon
5	FERENCZI	CSABA	KARCAG	74-JAN-30	KOZAK MARIA	742346BC	PUSPOKLADANY	76326429
4	FORIAN	JULIANNA	PUSPOKLADANY	55-FEB-02	MILE ERZSEBET	845311RT	PUSPOKLADANY	76240029
14	FRANC	JULIANNA	SOPRON	91-MÁJ-19	ORDASI TEREZ	998892UL	PUSPOKLADANY	10234544
11	HODOSI	SZABOLCS	BUDAPEST	80-JAN-05	CICAS JULIANNA	368892UU	DEBRECEN	44444444
9	KESERU	MATYAS	PUSPOKLADANY	51-MÁJ-09	KESERU ETELKA	368731BA	PUSPOKLADANY	54154177
7	KESERU	ANIKO	BERETTYOUJFALU	83-JÚN-06	KISS KATALIN	923619QW	PUSPOKLADANY	42984272
1	KESERU	KATA	PUSPOKLADANY	76-MÁJ-19	KISS KATALIN	543511AB	PUSPOKLADANY	21657328
17	MAGDA	MARINKO	ORADEA	67-DEC-30	BALOGH IREN	123492FF	PUSPOKLADANY	74631344
6	MIKLOSSY	TAMAS	DEBRECEN	83-NOV-22	KOVACS EVA	735421DV	DEBRECEN	98754242
324	MNBV	MHV	ORADEA	82-MÁJ-12	KFJDFGB FKJGE	252674ac	JGFK	46276767
8	NAGY	RITA	BUDAPEST	70-JAN-10	SZABO PIROSKA	234265AH	DEBRECEN	24523657
13	SZABO	TAMAS	PUSPOKLADANY	81-DEC-10	SZANTO IREN	368112RT	PUSPOKLADANY	44234244
16	SZABO	PETER	PUSPOKLADANY	70-MÁR-15	ROMBOLO IREN	968192IU	PUSPOKLADANY	87461222
18	TOTH	HAJNALKA	PUSPOKLADANY	75-MÁJ-19	FORIAN JULIANNA	403431ZZ	PUSPOKLADANY	51432543
2	TOTH	MIKLOS	DEBRECEN	82-MÁR-15	FORIAN JULIANA	532567AA	PUSPOKLADANY	65438763
3	TOTH	MIKLOS	PUSPOKLADANY	55-MÁJ-22	HORVAT VILMA	345236WE	PUSPOKLADANY	63463488
10	VEG	BELA	TIMBUKTU	60-DEC-06	BALOGH CECILIA	456783MN	KARCAG	98765239

Az osztályvezetőnek így lehetősége van megtekinteni a rendszerbe regisztrált ügyfelek adatait.

Segélyfajták

Page 1

Segelyfajta Neve	APOLASI DIJ			
Segély összege	Kiutalás gyakorisága	Öregségi nyugdij összege	Százalék értéke	Hányszor igényelhető
30000	HAVONTA	60000	50	1
Segelyfajta Neve	ATMENETI SEGELY			
Segély összege	Kiutalás gyakorisága	Öregségi nyugdij összege	Százalék értéke	Hányszor igényelhető
6000	EGYSZERI	60000	10	5
Segelyfajta Neve	BEISKOLAZASI SEGELY			
Segély összege	Kiutalás gyakorisága	Öregségi nyugdij összege	Százalék értéke	Hányszor igényelhető
9000	EGYSZERI	60000	15	1
Segelyfajta Neve	ETKEZESI TERITESI DIJTAMOGATAS			
Segély összege	Kiutalás gyakorisága	Öregségi nyugdij összege	Százalék értéke	Hányszor igényelhető
3000	HAVONTA	60000	20	1
Segelyfajta Neve	LAKASFENNTARTASI TAMOGATAS			
Segély összege	Kiutalás gyakorisága	Öregségi nyugdij összege	Százalék értéke	Hányszor igényelhető
12000	HAVONTA	60000	20	1

Ellenőrizheti, hogy a segélyek adatai pontosak-e.

A még folyamatban leő segélyigénylések

Page 1

Igenylo	Ugyintezo	Segelyfajta	Igenyelt	Datum	Igenyles Oka	% az egészből:
3	3	3	30000	07-JAN-19	A rokkant csaladtag eltartasa miatt	60.00%
	4	1	6000	07-MÁR-06	berlet kell venni	20.00%
	2	1	6000	07-FEB-06	berlet kell venni	20.00%
	% az egészből:	16.67%				
5	4	1	6000	07-JAN-31	berlet kell venni hogy be tudjak jarni dolgozni	33.33%
	3	2	3000	07-JAN-05	nehez anyagi helyzetem miatt	66.67%
	% az egészből:	10.00%				
11	2	7	9000	07-FEB-14	a tanulmanyaimhoz szuksegem van hogy konyvet vegyek	100.00%
	% az egészből:	23.33%				
14	3	1	6000	07-JAN-14	segitsegre van szuksegem, hogy tuzelot tudjak venni	100.00%
	% az egészből:	3.33%				
16	3	4	12000	07-MÁR-06	a csaladban bekuvetkezett tragikus halaleset	100.00%

Az osztályvezető nyomon követheti a még el nem bírált segélyigényléseket.

9. Adatok exportálása és importálása

9.1 Exportálás

Az adatbázisban található adatokról érdemes és hasznos bizonyos időközönként másolatot készíteni, hogy a rendszer sérülésekor ne vesszen el minden információ. Az adatvesztéssel járó sérülés után a másolatokból visszaállítható az adatbázis egy korábbi állapota. A másolatok készítésének a gyakoriságáról a rendszer kezelője szabadon dönthet, de ajánlatos hetente legalább egy mentés készítése. Az adatmentésnek az egyik lehetséges formája, hogy minden adat mentésre kerül. Nagy mennyiségű adat esetén ez több ideig tart és nagyobb tárterület kell neki, de a visszatöltés egyszerűbb.

Az export fájl az adatokat a következő sorrendben tartalmazza:

- Típusdefiníciók
- Tábladefiníciók
- Táblák adatai
- Táblák indexei
- Integritási megszorítások, nézetek eljárások és triggerek

9.2 Importálás

Segítségével egy Oracle adatbázisba importálhatunk adatokat egy export fájlból. Csak az export eszközzel létrehozott állományokat tudja olvasni. Az import során az operációs rendszer egy állományát olvassa a program, és ez alapján adatbázis objektumokat hoz létre. Az import eszköz nem lehet korábbi verziójú, mint az exportálást végző eszköz.

Az Oracle az adatbázisbeli adatok mentésére és visszatöltésére két parancssori eszközt kínál, ezek az export és import. Az elkészített alkalmazáshoz is ezeket használtam. A mentéshez és visszatöltéshez készítettem 2 batch fájlt aminek a tartalma a következő:

Exportálás:

exp system/katica full=y consistent=y rows=y buffer=1000000 file=hivatal.dmp log=hivatal.log statistics=none

Ez a parancs abba a könyvtárba hozza létre a fájlokat, ahol kiadjuk az utasítást.

Importálás:

imp system/katica full=y ignore=y rows=y buffer=1000000 file=hivatal.dmp log=hivatal.log

Ez a parancs szolgál az adatbázis visszatöltésére egy már korábban elkészített export fájlból.

10. Programozói leírás

Ebben a fejezetben igyekszem megkönnyíteni a rendszer továbbfejlesztését azzal, hogy leírom az egyes programegységek hol találhatóak és milyen funkciójuk van.

Az adatbázisban szereplő táblák és objektumok létrehozására egy olyan SQL scriptet írtam, ami többször lefuttatható, azaz ha már léteznek az adatbázisban a táblák és az objektumok, akkor mielőtt újra létrehozná őket, törli a sémából. Ez a script megtalálható a mellékelt CD hivatal könyvtárában.

A táblák és a közöttük lévő kapcsolatok a **6.5.2-**es fejezetben látható.

Az adatok tárolását adatbázisban tárolt táblák végzik. Létrehoztam szekvenciákat amiket az adatbázisban tárolok ezeket azért hoztam létre hogy a táblák elsődleges kulcsának az automatikus generálása és egyedisége biztosított legyen. Ezeket használom az igenylo_azon, gyermek_azon, segely_igenyles_azon, elbiralva_azon, segelyfajta_azon, ugyintezo_azon, folyositott_juttatasok_azon automatikus generálására.

Az elbíráláshoz kapcsolatos eljárásokat nem az adatbázisban, hanem a formban tároltam így is mutatva a 4GL eszközök ehetőségeit. Az eljárás ellenőrzi, hogy egy adott igénylő az adott segélyfajtából elére e a maximálisan igénybe vehetőt. Ezt úgy végzem el, hogy az elbírálva táblában ellenörzöm, hogy az adott segélyfajtára vonatkozóan az igénylő azonosító szerepel e annyiszor, mint amennyi a segély maximálisan igénybe vehető mennyisége. Itt nyilván csak azon rekordok között folyik a vizsgálat, ahol az elbírálás értéke 'TAMOGATVA'.

A program különböző formokat és riportokat használ.

A formok a következők:

- A program menüje.
  Ez a form a program főmenüje innen érhetőek el az egyes felhasználók különböző funkciói.

- Igenylo_gyerekekkel.
  A rögzítő ezen a felületen viheti fel az ügyfél és a gyermekei adatait illetve módosíthatja őket, ha szükséges.
- Igenylo_info
  Ez a felület szolgál arra, hogy az igénylő a saját adatait megtekintse
- Segely_igenyles
  A rögzítő itt készítheti el a segély igénylést. Megadja az igénylő adatait és a többi szükséges információt.
- Elbiralas
  Az elbíráló ezen a felületen végzi a segély elbírálását és itt kap értesítést arról, hogy ha az igénylő elérte az adott segélyfajtából a maximálisan igénybe vehetőt. A formban található a fent részletezett ellenőrző eljárás is.
- Az osztályvezető lehetőségei
  Ezen a formon keresztül érheti el az osztályvezető a kívánt összegző riportokat
- Segely_ugyintezo_adatok
  Itt érhetőek el illetve módosíthatóak a segélyek, illetve alkalmazottak adatai.

A riportok a következők:

- Folyamatban levő segélyek
  Itt tekintheti meg az osztályvezető a még el nem bírált segély igényléseket
- Folyósított juttatások
  Itt tekinthető meg a már elbírált segély igénylések
- Ügyintézők
  Itt kaphatunk információt az egyes felhasználókról: elbíráló, rögzítő, osztályvezető
- Segélyfajták
  Itt tekinthetők meg a segélyek adatai.

Az adatbázist létrehozó script a következőkben látható:

--TOROLJUK A SZOCIALIS RENDSZER OBJEKTUMAIT, HA MAR LETEZNEK

```
--CASCADE CONSTRAINTS A TABLARA HIVATKOZO KULSO MEGSZORITASOKKAL EGYUTT
--PURGE (ORACLE 10G ) A TOROLT TABLAT ES INDEXET NE TEGYE A LOMTARBA


DROP TABLE ELBIRALVA CASCADE CONSTRAINT;
DROP TABLE FOLYOSITOTT_JUTTATASOK CASCADE CONSTRAINT;
DROP TABLE GYERMEKEK CASCADE CONSTRAINT;
DROP TABLE IGENYLO CASCADE CONSTRAINT;
DROP TABLE SEGELYFAJTAK CASCADE CONSTRAINT;
DROP TABLE SEGELY_IGENYLESEK CASCADE CONSTRAINT;
DROP TABLE UGYINTEZOK CASCADE CONSTRAINT;

DROP SEQUENCE  UGYINTEZO_AZON_KOV_SZAMA CASCADE CONSTRAINT;
DROP SEQUENCE SEGELY_IGENYLES_AZON_KOV_SZAMA CASCADE CONSTRAINT;
DROP SEQUENCE  SEGELY_AZON_KOV_SZAMA CASCADE CONSTRAINT;
DROP SEQUENCE  IGENYLO_AZON_KOV_SZAM CASCADE CONSTRAINT;
DROP SEQUENCE  GYERMEK_AZON_KOV_SZAM CASCADE CONSTRAINT;
DROP SEQUENCE  FOLYOSITOTT_AZON_KOV_SZAMA CASCADE CONSTRAINT;
DROP SEQUENCE  ELBIRALVA_AZON_KOV_SZAMA CASCADE CONSTRAINT;


--A TABLAK LETREHOZASA

CREATE TABLE  "UGYINTEZOK"
  (     "UGYINTEZO_AZON" NUMBER NOT NULL ENABLE,
        "UGYINTEZO_CSALADI_NEVE" VARCHAR2(20) NOT NULL ENABLE,
        "UGYINTEZO_UTONEVE" VARCHAR2(20) NOT NULL ENABLE,
        "BEOSZTASA" VARCHAR2(20) NOT NULL ENABLE,
```

```
        CONSTRAINT "UGYINTEZOK_PK" PRIMARY KEY ("UGYINTEZO_AZON")
ENABLE
  )
/

CREATE TABLE  "IGENYLO"
  (     "IGENYLO_AZON" NUMBER NOT NULL ENABLE,
        "IGENYLO_CSALADI_NEVE" VARCHAR2(20) NOT NULL ENABLE,
        "IGENYLO_UTONEVE" VARCHAR2(20) NOT NULL ENABLE,
        "SZUL_HELY" VARCHAR2(20) NOT NULL ENABLE,
        "SZUL_DATUM" DATE NOT NULL ENABLE,
        "ANYJA_NEVE" VARCHAR2(40) NOT NULL ENABLE,
        "ALLAMPOLGARSAGA" VARCHAR2(15) NOT NULL ENABLE,
        "SZEM_IG_SZAM" VARCHAR2(10) NOT NULL ENABLE,
        "IRANYITOSZAM" NUMBER NOT NULL ENABLE,
        "TELEPULES" VARCHAR2(20) NOT NULL ENABLE,
        "KOZTERULET" VARCHAR2(25) NOT NULL ENABLE,
        "HAZSZAM" NUMBER NOT NULL ENABLE,
        "ADO_AZON" NUMBER NOT NULL ENABLE,
        "TAJ_SZAM" NUMBER NOT NULL ENABLE,
        "TELEFONSZAM" NUMBER,
        "EMAIL" VARCHAR2(40),
        "IGENYLO_BANKJANAK_NEVE" VARCHAR2(100),
        "IGENYLO_BANKJANAK_SZAMLASZAMA" NUMBER,
        "IGENYLO_SZAMLASZAMA" NUMBER,
        CONSTRAINT "IGENYLO_PK" PRIMARY KEY ("IGENYLO_AZON")
ENABLE,
        CONSTRAINT "IGENYLO_CON_UNIQUE_ADO" UNIQUE ("ADO_AZON")
ENABLE,
        CONSTRAINT "IGENYLO_CON_UNIQUE_TAJ" UNIQUE ("TAJ_SZAM")
ENABLE
  )
```

```
/

CREATE TABLE  "GYERMEKEK"
  (       "GYERMEK_AZON" NUMBER NOT NULL ENABLE,
          "IGENYLO_AZON" NUMBER NOT NULL ENABLE,
          "GYERMEK_CSALADI_NEVE" VARCHAR2(20) NOT NULL ENABLE,
          "GYERMEK_UTONEVE" VARCHAR2(20) NOT NULL ENABLE,
          "SZUL_HELY" VARCHAR2(20) NOT NULL ENABLE,
          "TAJ_SZAM" NUMBER NOT NULL ENABLE,
          "SZUL_IDO" DATE NOT NULL ENABLE,
           CONSTRAINT  "GYERMEKEK_PK"  PRIMARY  KEY  ("GYERMEK_AZON")
ENABLE,
           CONSTRAINT "GYERMEKEK_CON" FOREIGN KEY ("IGENYLO_AZON")
            REFERENCES  "IGENYLO" ("IGENYLO_AZON") ENABLE
  )
/

CREATE TABLE  "SEGELYFAJTAK"
  (       "SEGELYFAJTA_AZON" NUMBER NOT NULL ENABLE,
          "SEGELYFAJTA_NEVE" VARCHAR2(60) NOT NULL ENABLE,
          "SEGELY_OSSZEGE" NUMBER NOT NULL ENABLE,
          "KIUTALAS_GYAKORISAGA" VARCHAR2(15) NOT NULL ENABLE,
          "OREGSEGI_NYUGDIJ_MIN_OSSZEGE" NUMBER,
          "SZAZALEK_ERTEK" NUMBER,
          "HANYSZOR_IGENYELHETO" NUMBER,
           CONSTRAINT          "SEGELYFAJTAK_PK"          PRIMARY          KEY
("SEGELYFAJTA_AZON") ENABLE    )
/

CREATE TABLE  "SEGELY_IGENYLESEK"
  (       "SEGELY_IGENYLES_AZON" NUMBER NOT NULL ENABLE,
          "IGENYLO_AZON" NUMBER NOT NULL ENABLE,
```

```
        "UGYINTEZO_AZON" NUMBER NOT NULL ENABLE,
        "SEGELYFAJTA_AZON" NUMBER NOT NULL ENABLE,
        "IGENYELT_OSSZEG" NUMBER NOT NULL ENABLE,
        "DATUM" DATE DEFAULT sysdate,
        "IGENYLES_OKA" VARCHAR2(200) NOT NULL ENABLE,
        "IGENYLES_ALLAPOTA" VARCHAR2(20) NOT NULL ENABLE,
         CONSTRAINT "SEGELY_IGENYLESEK_PK" PRIMARY KEY
("SEGELY_IGENYLES_AZON") ENABLE,
         CONSTRAINT "SEGELY_IGENYLESEK_CON" FOREIGN KEY
("SEGELYFAJTA_AZON")
          REFERENCES "SEGELYFAJTAK" ("SEGELYFAJTA_AZON") ENABLE,
         CONSTRAINT "SEGELY_IGENYLESEK_CON2" FOREIGN KEY
("IGENYLO_AZON")
          REFERENCES "IGENYLO" ("IGENYLO_AZON") ENABLE,
         CONSTRAINT "SEGELY_IGENYLESEK_CON3" FOREIGN KEY
("UGYINTEZO_AZON")
          REFERENCES "UGYINTEZOK" ("UGYINTEZO_AZON") ENABLE
   )
/

CREATE TABLE "ELBIRALVA"
   (    "ELBIRALVA_AZON" NUMBER NOT NULL ENABLE,
        "EREDMENYE" VARCHAR2(40) NOT NULL ENABLE,
        "INDOKLAS" VARCHAR2(200) NOT NULL ENABLE,
        "ELBIRALAS_DATUMA" DATE DEFAULT sysdate,
        "KIUTALT_OSSZEG" NUMBER,
        "KIUTALAS_DATUMA" DATE,
        "ELBIRALO_AZON" NUMBER NOT NULL ENABLE,
        "KIFIZETES_MODJA" VARCHAR2(15),
        "SEGELY_IGENYLES_AZON" NUMBER NOT NULL ENABLE,
        "UTOLSO_KIUTALAS_DATUMA" DATE,
```

```
        CONSTRAINT "ELBIRALAS_PK" PRIMARY KEY ("ELBIRALVA_AZON")
ENABLE,
        CONSTRAINT "ELBIRALAS_CON" FOREIGN KEY ("ELBIRALO_AZON")
         REFERENCES  "UGYINTEZOK" ("UGYINTEZO_AZON") ENABLE,
        CONSTRAINT "ELBIRALVA_CON_KULSO" FOREIGN KEY
("SEGELY_IGENYLES_AZON")
         REFERENCES "SEGELY_IGENYLESEK" ("SEGELY_IGENYLES_AZON")
ENABLE
  )
/
CREATE TABLE  "FOLYOSITOTT_JUTTATASOK"
  (     "FOLYOSITOTT_AZON" NUMBER NOT NULL ENABLE,
        "FOLYOSITOTT_OSSZEG" NUMBER NOT NULL ENABLE,
        "ELBIRALVA_AZON" NUMBER NOT NULL ENABLE,
        CONSTRAINT "FOLYOSITOTT_JUTTATASOK_PK" PRIMARY KEY
("FOLYOSITOTT_AZON") ENABLE,
        CONSTRAINT "FOLYOSITOTT_JUTTATASOK_CON4" FOREIGN KEY
("ELBIRALVA_AZON")
         REFERENCES  "ELBIRALVA" ("ELBIRALVA_AZON") ENABLE
  )
/

--SZEKVENCIAK LETREHOZASA
CREATE SEQUENCE "UGYINTEZO_AZON_KOV_SZAMA" MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 5
NOCACHE NOORDER NOCYCLE
/
 CREATE SEQUENCE "SEGELY_IGENYLES_AZON_KOV_SZAMA" MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 5
NOCACHE NOORDER NOCYCLE
/
```

```
CREATE SEQUENCE "SEGELY_AZON_KOV_SZAMA" MINVALUE 1 MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 5 NOCACHE
NOORDER NOCYCLE
/
CREATE SEQUENCE "IGENYLO_AZON_KOV_SZAM" MINVALUE 1 MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 10 NOCACHE
NOORDER NOCYCLE
/
CREATE SEQUENCE "GYERMEK_AZON_KOV_SZAM" MINVALUE 1
MAXVALUE 9999999999999999999999999 INCREMENT BY 1 START WITH 10
NOCACHE NOORDER NOCYCLE
/
CREATE SEQUENCE "FOLYOSITOTT_AZON_KOV_SZAMA" MINVALUE 1
MAXVALUE 9999999999999999999999999 INCREMENT BY 1 START WITH 1
NOCACHE NOORDER NOCYCLE
/
CREATE SEQUENCE "ELBIRALVA_AZON_KOV_SZAMA" MINVALUE 1
MAXVALUE 9999999999999999999999999 INCREMENT BY 1 START WITH 5
NOCACHE NOORDER NOCYCLE
/
```

A táblákat adatokkal való feltöltéséhez a következő scriptet készítettem ez a CD-n a hivatal könyvtárban található meg.

11. Összefoglalás

Az alkalmazás elkészítése közben egyre inkább szembesültem, azzal, hogy az általam választott téma mennyire sokrétű. „Kutakodnom” kellett, hogy átfogó képet kapjak a szociális osztály ügyintézéséhez. A diplomamunka szövegének az elkészítéséhez nagyon sok forráshivatkozást át kellett olvasni illetve keresni, hogy egy egységes egészet alkosson. A szöveget magyarázó ábrákkal és képernyőképekkel tettem szemléletessé, így ha olyan ember olvassa a dokumentációt, aki esetleg még nem foglalkozott a területtel így is meg tudja érteni. A program megírása elején kompatibilitási problémákba ütköztem, amit az okozott, hogy a Microsoft által készített Virtual Pc 2004 alkalmazás nem telejsen kompatibilis a laptopomon szereplő Microsoft Windows XP Home Edition operációs rendszer. Ezt a problémát a program újabb kiadása Microsoft Virtual Pc 2007-es kiadása némileg orvosolta. Az egymás felett futtatott két rendszer és adatbázis szerver nagy erőforrásigénye arra késztetett, hogy ne virtulais gépen futassam az alkalmazást, hanem egy teljesen önálló szervert telepítsek a gépemen szereplő Microsoft Windows XP Home Edition fölé. Az ismerőseimmel és a tanáraimmal való egyeztetés után döntöttem az Oracle Express Editiom mellett, mert ennek kisebb az erőforrás igénye és felesleges lett volna nagyobb adatbázis-kezelő alkalmazást telepítenem, mivel nem használtam volna ki a benne rejlő funkciókat. Miután futott az adatbázis-szerver, jöhetett a Form és Report Builder telepítése. A telepítés után a Form Builder nem tudott csatlakozni a szerverhez, így a konfigurációs állományok testre szabása következett. Miután minden feltétel adott volt, elkezdhettem az alkalmazás fejlesztését. Az alkalmazás elkészítésénél nem csak a felületek kényelmes és design-os elkészítése volt nehéz, hanem a felületek mögött rejlő mögöttes folyamatok megtervezése, összehangolása és kivitelezése. A nagyobb problémát a segély ügyintézéssel kapcsolatos ellenőrző folyamatok jelentették. Ennek a megoldására segítséget kértem tanáraimtól, illetve arra kényszerültem, hogy a területtel kapcsolatos könyveket tanulmányozzam. Véleményem szerint az alkalmazás tökéletesen ellátja a feladatát. A könnyű bővíthetőség - új segélyfajták felvétele, segélyadatok módosíthatósága, és új alkalmazottak felvitele - kényelmessé teszi a program későbbi használatát is. Amit a későbbiekben még meg lehetne oldani, és ezzel bővíteni a rendszer funkcióit, hogy a rendszer képes legyen e-mail-ben tájékoztatni az ügyfeleket az éppen esedékes kifizetésekről. Esetlegesen egy webes felületről elérhető információnyújtás az ügyfélnek az adatairól, arról hogy milyen átutalások kerültek már kifizetésre és hogy a soron

következő átutalások mikor várhatóak. Ezzel is ember központúbbá válna az alkalmazás, mivel így az ügyfelek, ha rendelkeznek személyi számítógéppel és internet kapcsolattal „megspórolhatnák” az utat és az időt, amit azzal töltenek, hogy kivárják a sorukat, hogy megtudják a kívánt adatokat.

12. Irodalomjegyzék

- Juhász István és Gábor András: PL/SQL-programozás
- Márton Ágnes által készített elektronikus jegyzet - Alkalmazásfejlesztés 4GL eszközökkel. Ez a dokumentum letölthető a mobidiákról.
- Jeffrey D. Ullman- Jenifer Widom: Adatbázisrendszerek. Alapvetés
- Hector Garcia-Molina-Jeffrey D. Ullman-Jennifer Widom: Adatbázisrendszerek megvalósítása.
- A BME-n készített 4GL elektronikus jegyzet. www.mit.bme.hu/~tilly/pages/4gl.htm
- Az interneten elérhető segélyfajta leírások www.veszprem.hu-n

13. Köszönetnyilvánítás

Elsősorban szeretném megköszönni a barátnőm Keserű Kata segítségét, megértését, ötleteit és támogatását, ami nagyban hozzájárult ahhoz, hogy a diplomamunkám létrejöjjön. Köszönöm a családomnak a felém fordított megértését és anyagi támogatását. Szeretném még megköszönni a témavezető tanáromnak Márton Ágnesnek a szakmai segítségett, amit nyújtott.