BKSZC Pogány Frigyes Szakgimnáziuma
Szoftverfejlesztő
54 213 05

Információs rendszer kis- és középvállalatok számára

Tóth József
konzulens

Kiss J. Gábor
2/14. ED

Budapest, 2020

Tartalom

Ábrajegyzék

1 Felhasználói dokumentáció

1.1 A programról általában

Egy feladat, egy projekt megvalósítása során a résztvevők közötti tájékoztatási rendszer jó működése biztosítja a sikert. Lényeges, hogy az adatok átadása gördülékeny, áttekinthető és nyomon követhető legyen. Ehhez biztosít kiváló lehetőségeket az elektronikus kommunikáció, mely napjainkra az információ közvetítésének alapvető módjává vált.

A kapcsolattartás sokféle lehetőségét kínálják a különböző közösségi hálózatok és levelező programok. Ezeket a megoldásokat azonban nehézkes a szűkebb vállalati környezetben alkalmazni.

A tájékoztatási rendszer hiánya negatívan befolyásolja a cégvezető (projektvezető) tevékenységét. Az emberei elbeszélnek egymás mellett, pedig egy kis félreértés is hatalmas hibákat okozhat az üzletmenetben. A munkatársak felesleges adminisztratív munkával töltik drága idejüket. Stresszesen, papírhalmokat átnyálazva keresnek adatokat, mert nincs ott az a kolléga, aki tudja hol találhatók. Egy egyszerű kérdésre is órákig vadásszák össze az adatokat az Excel táblázatokból. Nem tudja melyik munkatársa mit csinál, ki eredményes és kitől kellene megválnia. Óriási kínlódást jelent egy távozó munkatárs pótlása, mert csak az Ő fejében voltak meg fontos információk.

Az információs rendszer egy kapcsolattartási megoldást kínál, mely elősegíti egy vállalat ügymenetét. Minden munkatárs könnyen hozzájut a számára lényeges információkhoz. Mindig látszik, hogy kinek mi a feladata. Átlátható a működés. A kapkodás és stressz helyett tervezett és nyugodt mindenki munkája. Azonnali, naprakész információk alapján gyorsan, helyes döntéseket lehet hozni. A vezető minden munkatársáról pontosan tudja, hogy éppen mit csinál, min dolgozik.

A program alaprendszere egyszerűen módosítható és bővíthető a cég speciális igényinek figyelembevételével. Személyre szabott, a felelősséghez és munkakörhöz kapcsolódó hozzáférési jogosultságok állíthatók be. Az alapprogram adatbázisa tartalmazza a cég és a dolgozók adatait, valamint a vállalati projektekkel kapcsolatos munkájukat. A felhasználók a programban feljegyzéseik segítségével biztosítják az információ áramlását és dokumentálják tevékenységüket.

A programban elérhető lehetőségek igény szerint bővíthetők. A következő felsorolásban kiemelek néhány példát a lehetséges kiegészítő modulokra:

- munkaidő nyilvántartás
- szabadság nyilvántartás
- eszköz- és anyagnyilvántartás
- fizetések és juttatások
- útnyilvántartás (kiküldetési rendelvények)
- készletnyilvántartás
- partnerek
- kimenő és bejövő számlák nyilvántartása
- ajánlatok és szerződések nyilvántartása
- számlák nyilvántartása

…

Az információs rendszer programmal a résztvevők minden szinten pontos képpel rendelkeznek a projektek állásáról, folyamatosan értesülnek az új fejlesztésekről. Új modulok beépítésével az ügyviteli folyamatok automatizálhatók, így az időrabló feladatok néhány gombnyomással elvégezhetők lesznek.

1.2 Rendszerkövetelmények

1.2.1 Szoftver

1.2.1.1 Windows

Windows Vista SP2,
Windows 7 SP1,
Windows 8.x,
Windows 10 (8u51 vagy újabb),
Windows Server 2008 R2 SP1 (64-bit),
Windows Server 2012 and 2012 R2 (64-bit)

1.2.1.2 Mac OS X

Intel-alapú Mac és Mac OS X 10.7.3 (Lion) vagy későbbi,
Adminisztrátor jogok a telepítéshez.

1.2.1.3 Linux

- Oracle Linux 5.5+,
  Red Hat Enterprise Linux 5.5+,
  Suse Linux Enterprise Server 10 SP2,
  Ubuntu Linux 10.04 vagy újabb.

1.2.2 Hardver

Processzor: Pentium 2 266 MHz vagy gyorsabb,
RAM: min. 128 MB,
Szabad tárhely: min. 124 MB.

1.3 A program telepítése

A java futtatókörnyezet telepítése alapvető feltétele az alkalmazás használatának. A program fejlesztése java 1.8 környezetben történt, így futtatásához a Java SE Development Kit 8 telepítése ajánlott. Mivel kompatibilis a legújabb Oracle java verziókkal és az Open JDK-val is, ezért szükségtelen a már telepített java környezet megváltoztatása, vagy visszaállítása alacsonyabb verzió számúra.

A z alkalmazás MySQL alapú adatbázist kezel. Az adatbázis lehet lokális elérésű és lehet távoli elérésű is. A példában lokális elérésű adatbázist használok, mely nyilvános IP cím hiányában csak a munkahelyi magánhálózat gépeiről érhető el. Az adatbázis kezelést az XAMPP[1] webszerver-szoftvercsomag MariaDB összetevője biztosítja.

A java futtatókörnyezet és az XAMPP webszerver csomag telepítése, illetve meglétük ellenőrzése után az alkalmazás a mappájában található `B6kkvXxXx.jar` fájl elindításával futtatható. Ez egy úgynevezett portable (hordozható) program, mely telepítés nélkül, akár USB kulcsról is futtatható. Lényeges, hogy a tartalmazó mappát kompletten, mindenestől kell „hordozni”. Az alkalmazás működéséhez szükséges a `lib` mappa a benne található `mysql-connector-java-5.1.23-bin.jar` fájllal együtt.

Az adatbázis kezdeti beállítását
a `V01-01kkvDBcreate.sql`, a `V01-02kkvDBtables.sql`,
a `V01-03kkvDBusers.sql` és a `V01-04kkvDBdatas.sql` fájlok importálásával végezzük el. Ezek a beállítások érzékeny adatokat tartalmaznak, amik nem írhatók bele az alkalmazás kódjába. A továbbfejlesztés során szóba jöhet titkosítás az importálandó fájloknál.

1.4 A program használatának a részletes leírása

Az alkalmazás egy adatbázist használ, ami egy adatbáziskezelőn keresztül érhető el. Az adatbázis kezelést esetünkben az XAMPP csomag biztosítja. A helyi hálózat egyik gépén futnia kell az Apache webszervernek és a MariaDB adatbázis-

[1] Az XAMPP egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB (korábban a MySQL) adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői.

kezelőnek. Ezeket az XAMPP kontroll panelen indíthatjuk el, bekapcsolt állapotukat a a panelen a nevük zöld színű kiemelése jelzi.

1.4.1 Az alkalmazás indítása

Az alkalmazást a program mappájában található `B6kkvXxXx.jar` fájl futtatásával indítjuk. Első lépésben a program csatlakozik az adatbázishoz.

1. Ábra: Az alkalmazás csatlakozik az adatbázishoz

A csatlakozási folyamatot a menü sor alatti eszköz panelen kísérhetjük figyelemmel (1. ábra).

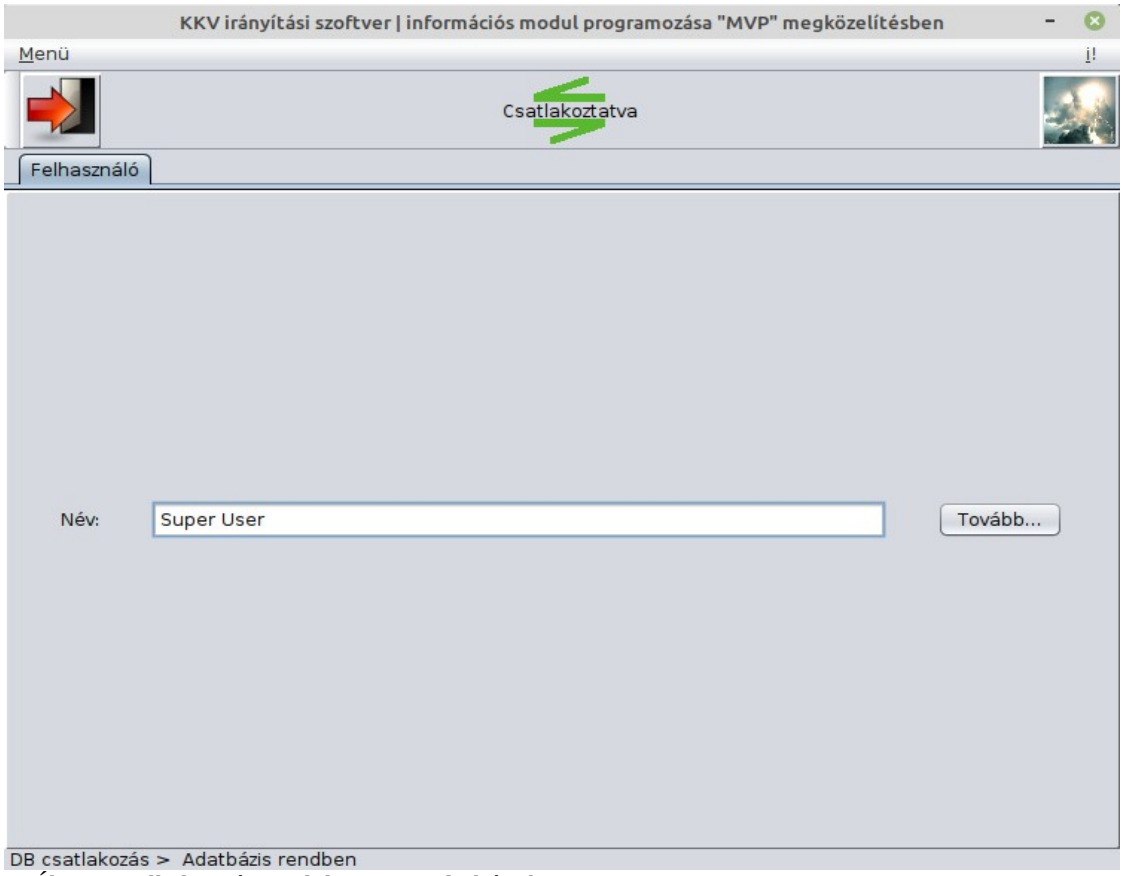
Amennyiben egy másik gépén működik a szerver és az adatbázis-kezelő, akkor az eszköztár jobb szélén található „felhő”-gombra kattintás után megjelenő ablakban ki kell egészíteni, távoli szerver esetén pedig meg kell adni a a szerver-gép IP-címét. Az elfogadás ([OK]-gomb) után ezen az új címen keresi az adatbázis-kezelőt a program és folytatja a csatlakozást.

Sikeres csatlakozás esetén

- az eszköztár jobb szélén a „felhő”-gomb világosabbra vált;
- a középső ikon átvált „csatlakoztatva” állapotba;
- az alsó állapotjelző sorban pedig megjelenik az „adatbázis rendben” felirat (2. ábra).

1.4.2 Belépés az adatbázisba

Az adatbázisba való bejelentkezéshez első lépésben az ablak közepén levő mezőben meg kell adni a felhasználó nevet, majd a „tovább” gombra kattintva a felhasználói jelszót.

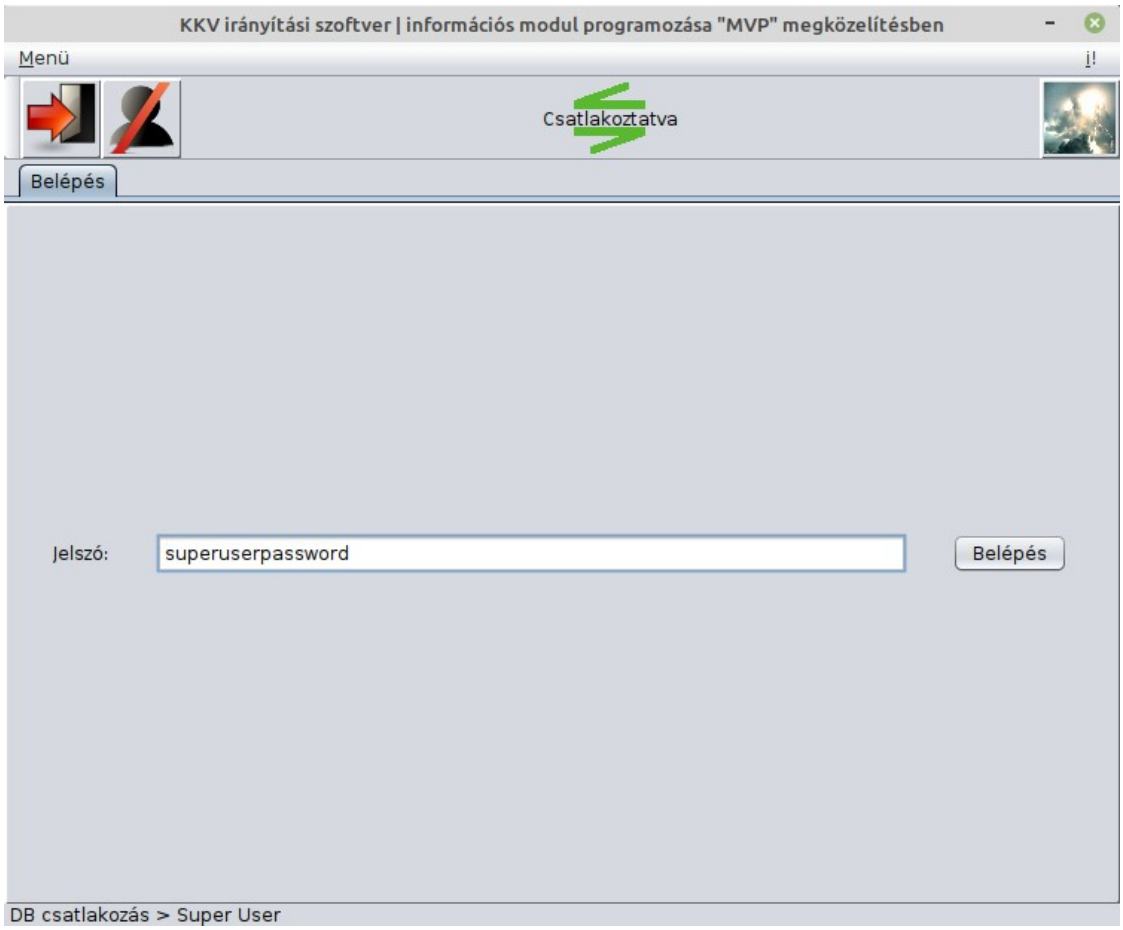


2. Ábra: Az alkalmazás csatlakozott az adatbázishoz

A bejelentkezés során az alkalmazás a felhasználói név alapján hozzárendeli a belépő felhasználóhoz a jogosultságot, ami alapján a program futása során csak az annak megfelelő elemeket jeleníti meg a munkaterületen. Ez azt jelenti a példa alkalmazásnál, hogy az adminisztrátori jogokkal rendelkező Super User az adatbázis teljes forgalmát láthatja, a csoport vezető a csoport összes feljegyzéséhez hozzáfér, míg a csoport tagjai esetünkben csak a saját maguk által létrehozott, vagy a nekik

szóló jegyzeteket láthatják. A jogosultságok és az általuk meghatározott hozzáférések megváltoztatása csak a nem kívánt szerepkör elvetésével és egy új szerepkör bevezetésével, teljes felépítésével történhet. Ebben az esetben ehhez az új jogosultsági szinthez új lekérdezések és új grafikus felületek is tartoznak, tartozhatnak.

Példánkban, amennyiben a felhasználó jogosultsággal rendelkezik az adatbázisba történő belépésre, akkor a [Tovább...] lenyomásával eljutunk a jelszó megadásához (3. ábra). Az állapot jelző sorban látható lesz a felhasználónév. Az eszköztárban a kilépés gomb mellett megjelenik a kijelentkezés gomb is, mely itt a bejelentkezés megszakításával visszavisz az előző képernyőre.

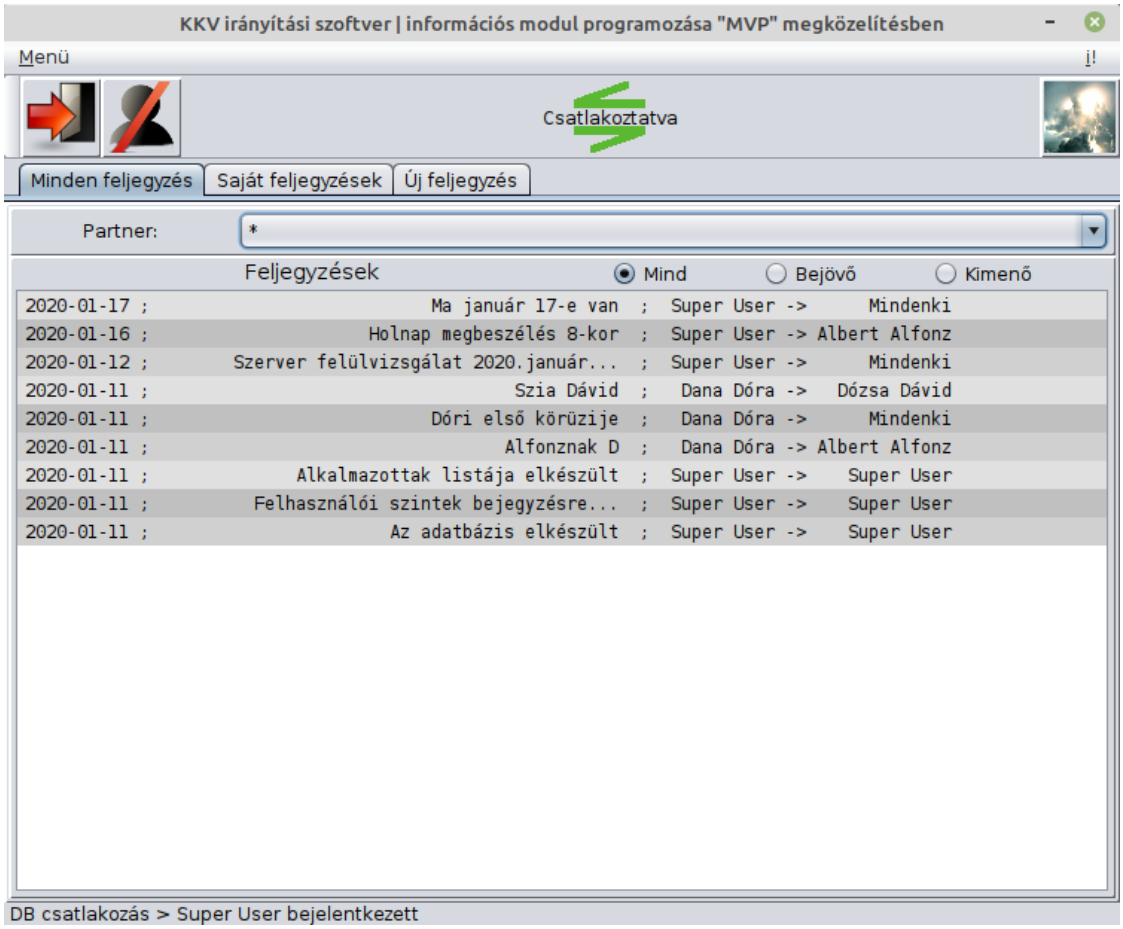


3. Ábra: A belépési jelszó megadása

Amennyiben az ablak közepén levő mezőbe a helyes jelszót adjuk meg, akkor a [Belépés] sikeresen megtörténik. A bejelentkezéshez használt felhasználónév és jelszó páros nem alkalmas az adatbázishoz történő közvetlen hozzáférésre (pl. phpMyAdmin-on keresztül), így is garantálva a rögzített adatok sérthetetlenségét.

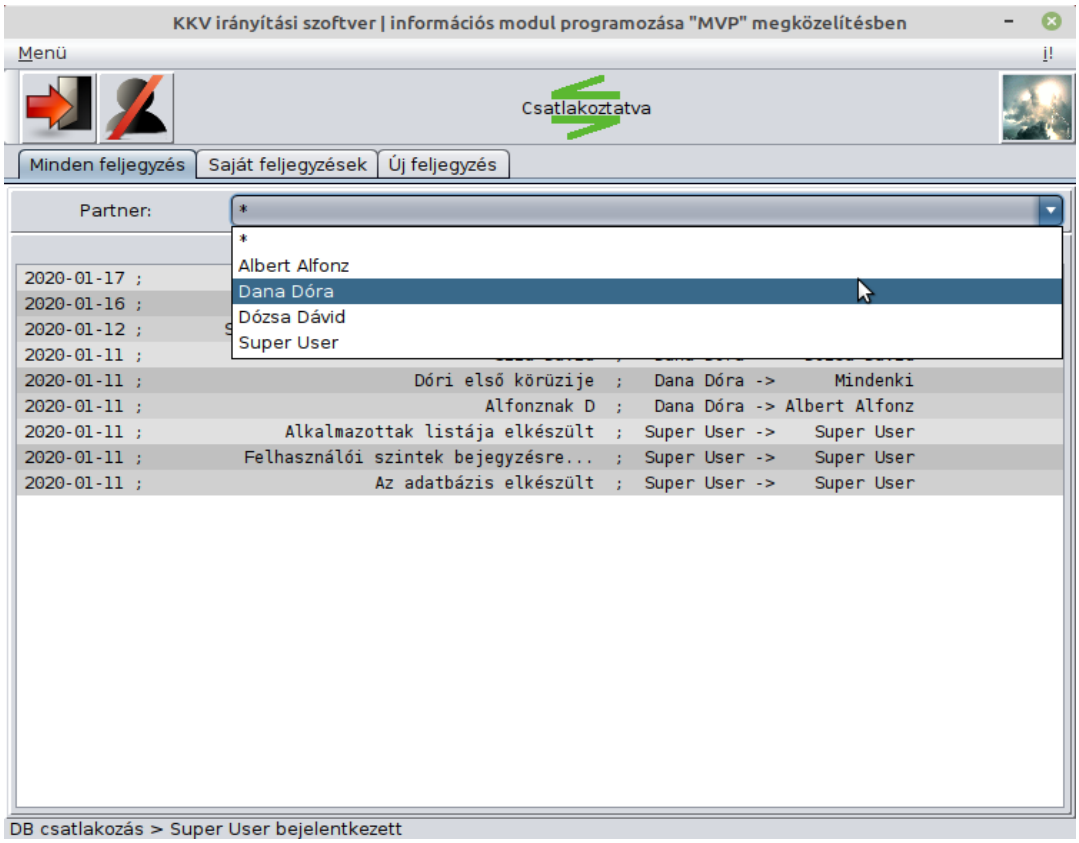
1.4.3 Az adatbázis használata

Tehát a [Belépés] gombra kattintva eljutunk a felhasználó jogosultságainak megfelelő munkafelülethez. Példánkban „Super User”-ként és vezetőként hozzáférhetünk a teljes információs adatbázishoz, csoporttagként pedig csak az általunk írt és a nekünk (is) címzett feljegyzésekhez. Feljegyzés írására esetünkben mindenki jogosult. Amennyiben nem vagyunk az információs adatbázissal támogatott fejlesztés részesei, akkor nincs jogosultságunk az adatbázisba történő belépésre sem.

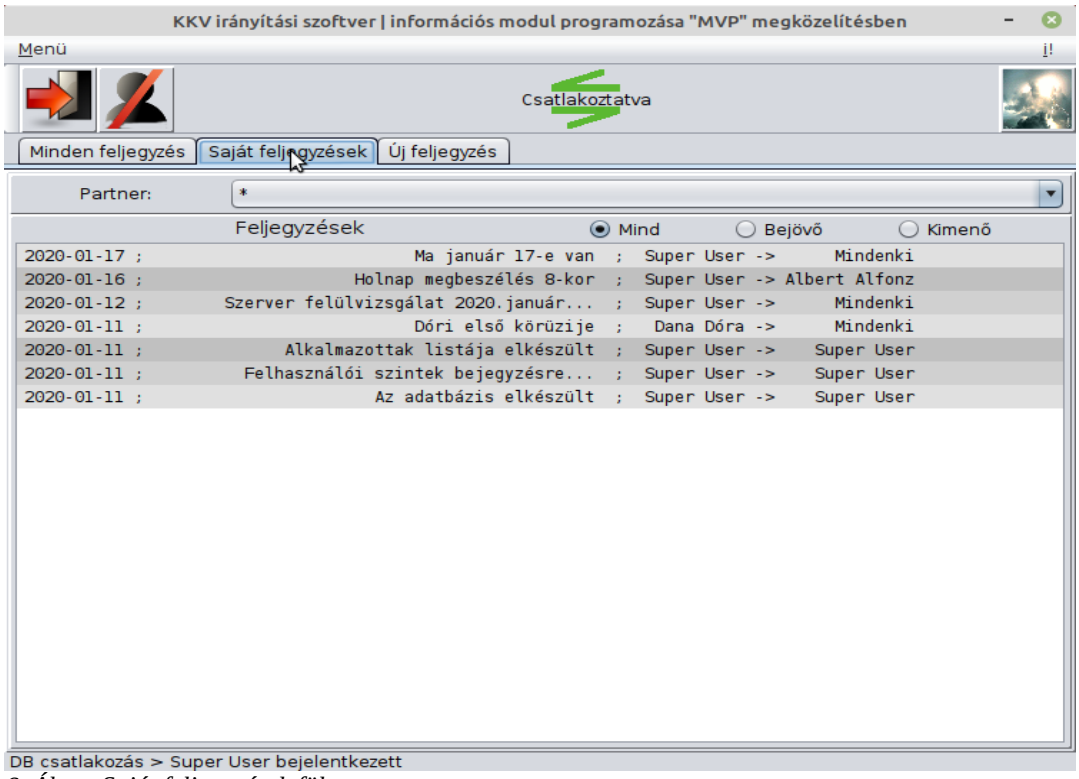


4. Ábra: Super User minden feljegyzés fül

A 4. ábrán látható a belépés utáni felhasználói felület. Az adminisztrátori és csoportvezetői jogokkal rendelkezőknek a „Minden feljegyzés” fül jelenik meg a „Saját feljegyzések” és az „Új feljegyzés” fülek fölött. A legördülő menüből kiválasztható, hogy kinek a tevékenysége érdekel (5. ábra). A * választása esetén az összes felhasználó feljegyzése látható lesz időrendileg csökkenő sorrendben. A „Mind”, a „Bejövő” és a „Kimenő” gombokkal tovább finomíthatjuk szűrést. Egy üzenetre kattintva egy felugró ablakban megjelenik a teljes üzenet, minden adatával együtt.



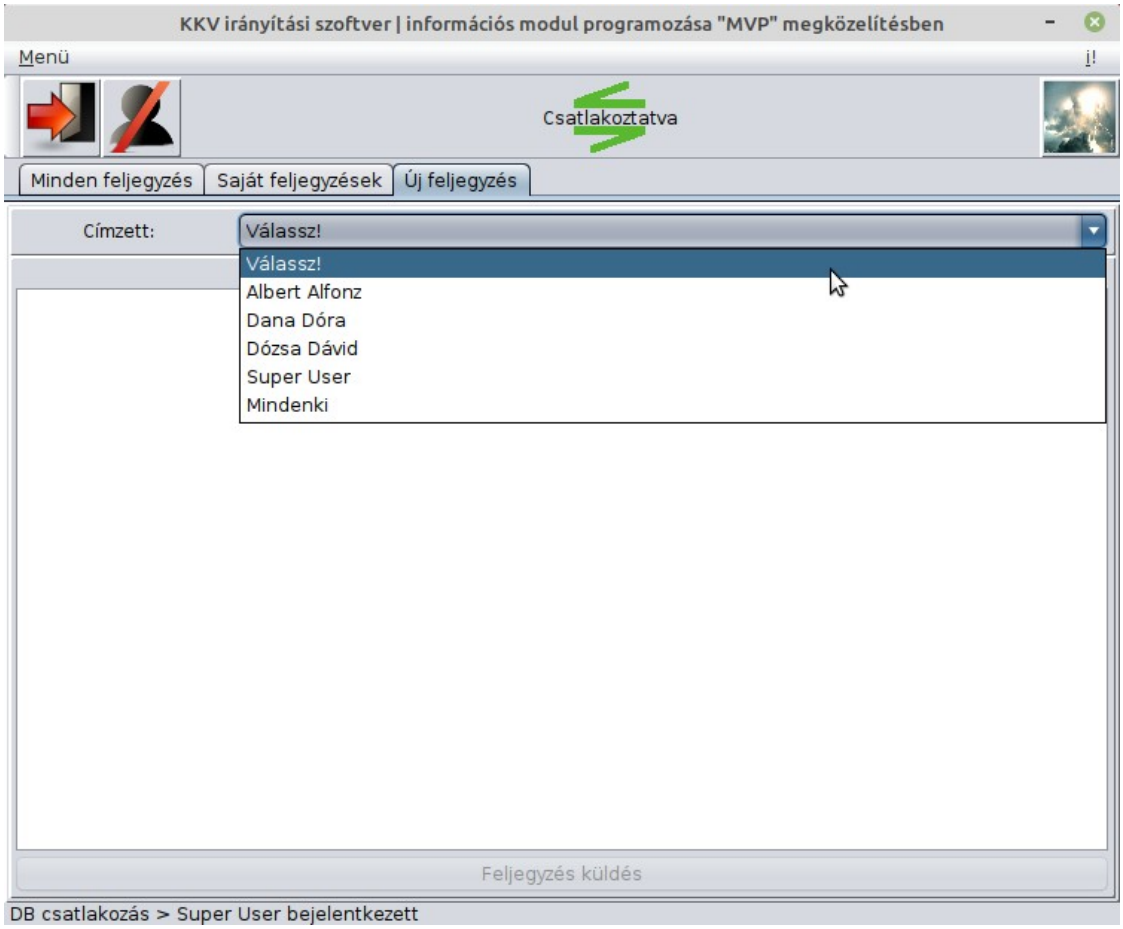
5. Ábra: Szűrés legördülő menüből név alapján



6. Ábra: Saját feljegyzések fül

A „Saját feljegyzések” fülön a felhasználó az általa írt és a neki címzett értesítéseket látja (6. ábra).

A tárolt adatoknál fontos követelmény a tárolt információ hitelessége. Ezért az elkészült feljegyzés nem módosítható és nem is törölhető. Amennyiben valamilyen okból a feljegyzés hibás adatokat tartalmazott, azokat csak egy új feljegyzés készítésével lehet módosítani. Ezzel a felelősség fontos kérdését igyekeztem figyelembe venni. A szűrés az előzőekkel megegyező módon történik. A feljegyzés teljes tartalma és adatai kattintásra itt is megjelennek.



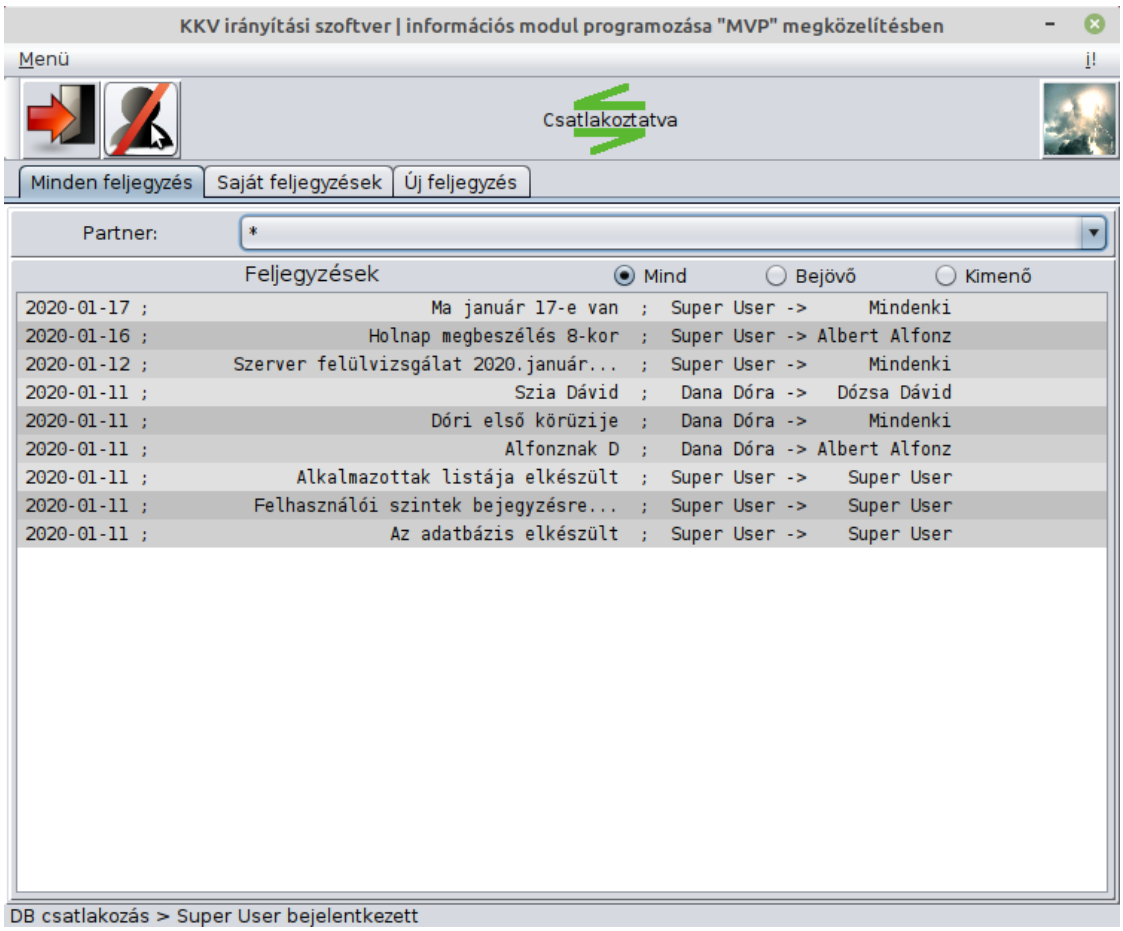
7. Ábra: Új feljegyzés írása

Az „Új feljegyzés” fülön saját feljegyzést, infó blokkot készíthetünk. Első lépésben ki kell választani azt a személyt, akivel az információt meg akarjuk osztani. Választható a „Mindenki” opció is (7. ábra). Ebben az esetben mindenki láthatja majd a kész dokumentumot. A feljegyzés gépelése közben engedélyezetté válik a „Feljegyzés küldés” gomb. A gombra kattintással visszavonhatatlanul rögzítésre kerül az adatbázisban a feljegyzés.

A feljegyzés moderálására csak a rendszergazda képes, az adatbázisba történő belépés után. Ekkor is csak a nem kívánatos esetleg sértő tartalmak semleges (pl.: xxx) karakterekkel történő feltöltése, felülírása a javasolt.

1.4.4 Kijelentkezés és kilépés a programból

A munka nézetből a bejelentkezés ablakhoz a kijelentkezés gombbal, vagy a Menüben található Kijelentkezés pont aktiválásával, vagy az **Alt+x** billentyűkombinációval lehet visszatérni.



8. Ábra: A kijelentkezés és kilépés gombok az eszköztáron

A programból a kilépés gombbal, vagy a Menüben található Kilépés pontjának, vagy az eszköztár kilépés gombjának aktiválásával, és az **Alt+F4** billentyűkombinációval lehet kilépni. Ezekben az esetekben egy rövid animációval záródik be az ablak. Amennyiben az ablakot bezáró [x] gombot használjuk a kilépéshez, akkor az animáció nem jelenik meg (8. ábra).

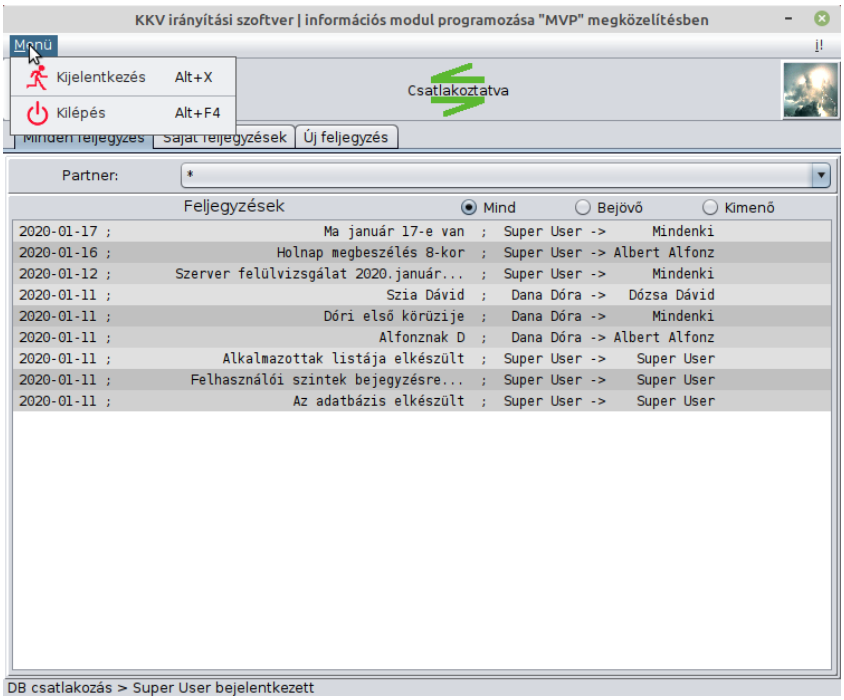
A program a beküldött adatokat azonnal tárolja, így a kilépés a heveny viszketést okozó visszakérdezés - „Biztosan kilép a programból? (I/N)” - nélkül történik.

1.4.5 A grafikus felület részei

1.4.5.1 A menü sáv

A menü sávon a főmenü (Menü) és az információs menü (i!) menü található. A program továbbfejlesztése során ez igények alapján további bővíthető.

A „Menü” pont induláskor csak a „Kilépés” lehetőségét tartalmazza. A „Kijelentkezés” almenü (9. ábra) a bejelentkezési folyamattal párhuzamosan kerül hozzáfűzésre.

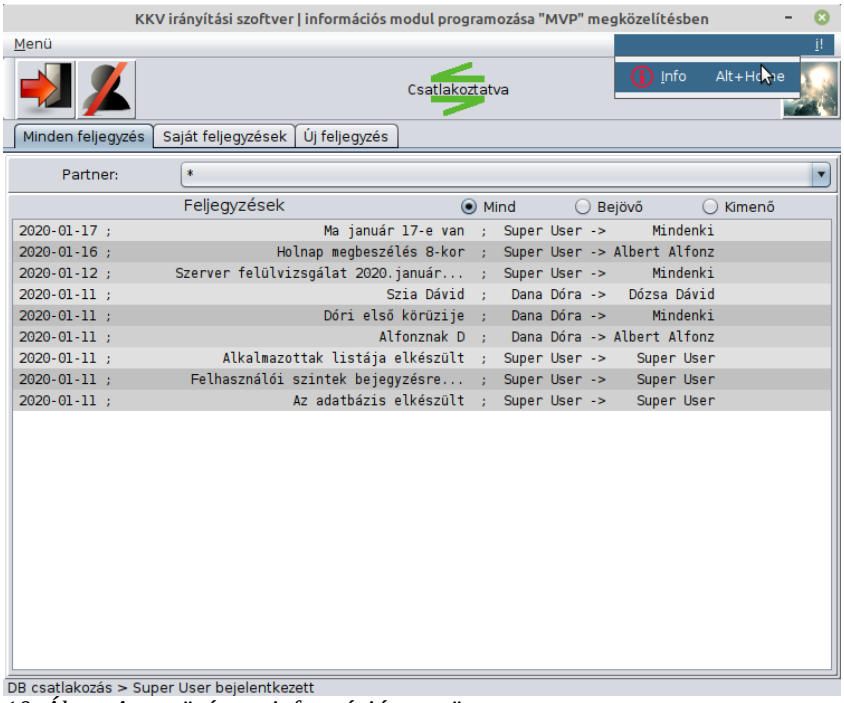


9. Ábra: A menüsáv, a főmenü

A menü sáv további menüpontokkal történő bővítése hasonlóan az almenü pontjaihoz, a programban megjelenő igény szerint dinamikusan történhet.

Például, amennyiben a „Super User” felhasználóra bízzuk a felhasználók kezelését a programban, akkor megjelenítjük a „Felhasználó” menüt a menüsávon és hozzáfűzésre kerülnek a lenyíló menüben az „Új felhasználó”, a „Felhasználó törlése” és a „Felhasználó adatainak módosítása” alpontok is. Természetesen csak akkor, ha a „Super User” lép be.

Az információs menü egy alpontjában a program néhány adata jeleníthető meg. A fejlesztés során itt helyet kaphatnak a segítség, a copyright, a cégadatok és más további alpontok is. Az információs menü aktiválható az **Alt+i** billentyű kombinációval is. Az **Alt+Home** kombináció pedig közvetlenül aktiválja a felugró információs ablakot (10. ábra).



10. Ábra: A menüsáv, az információs menü

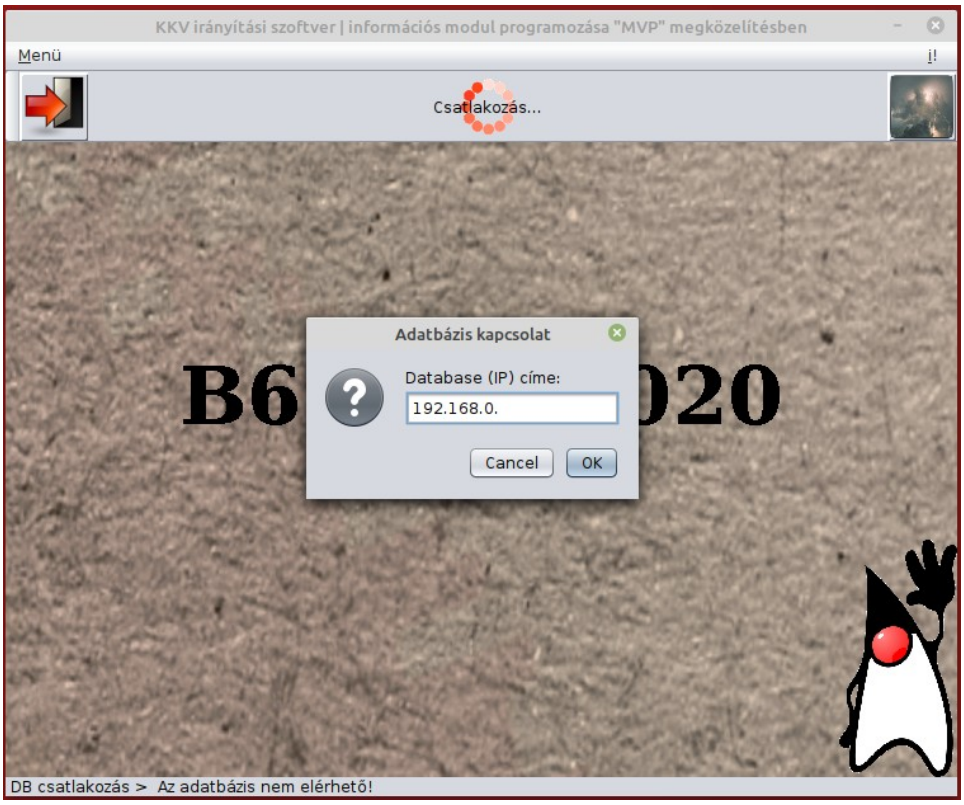
1.4.5.2 Az eszköztár

Az eszköztáron található gombok tárháza bővíthető, láthatóságuk az éppen elérhető funkciók szerint változtatható.

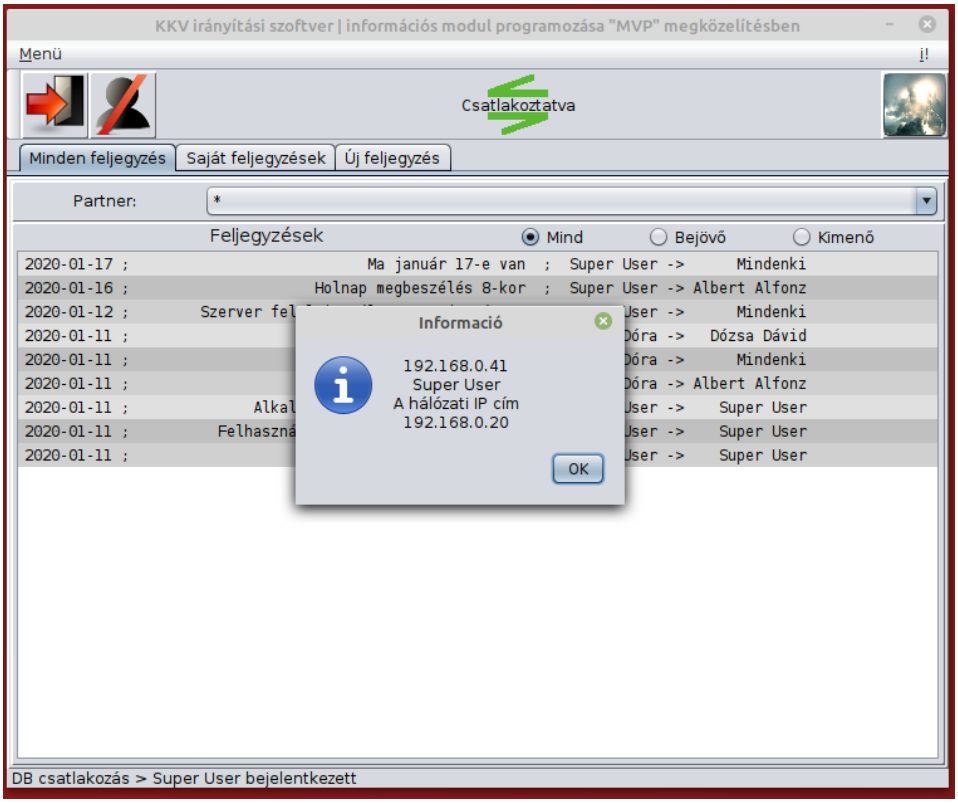
A „kilépés” és „kijelentkezés” gombok megjelenítése a végleges felületen indokoltnak látszik, hiszen ezt a két funkciót az átlagos felhasználó is használni fogja.

A „csatlakozás” és „felhő” gombok tesztüzemben mindenképpen hasznosak, bár az átlagos felhasználó érdeklődése nyilván nem terjed ki az általuk nyújtott információkra. Az előbbi ellenőrzi és jelzi az adatbázishoz történő csatlakozás folyamatát és a sikeres csatlakozás tényét. Az utóbbi egy sikertelen csatlakozási folyamat esetén lehetőséget ad a szerver IP-címének közvetlen megadására (11. ábra), illetve sikeres csatlakozás esetén információkkal szolgál a kapcsolatról (12. ábra).

A program jelenlegi verziójának eszköztára jelenleg tartalmazza a fent fölsorolt gombokat.



11. Ábra: A szerver IP-címének megadása



12. Ábra: Kapcsolati adatok

1.4.5.3 Az állapotjelző sáv

Az állapotjelző sávban a folyamatban lévő tevékenységgel kapcsolatos üzeneteket jeleníti meg a program.

1.4.6 Fejlesztési lehetőségek

A fejlesztések a grafikus felületekbe épített új funkciók könnyebben használhatóvá tehetik az alkalmazást, az új modulok hozzáfűzésével pedig a program hasznosságát fokozzák.

Grafikus fejlesztési lehetőségekre példák:

- Az üzenetek megtekintése panelhez radiogomb hozzáfűzése a saját magunknak címzett üzenetek kiválaszthatóságára.

- A teljes üzenet megtekintése ne külön felugró ablakban, hanem harmonika szerűen nyíljon meg a fókuszba került üzenet az üzenet listában.

- Keresés lehetőségének a beépítése az üzenetek panelbe.

Modul fejlesztési lehetőségek:

- A „Super User” felhasználóként belépett felhasználó egy külön panelen adminisztrálhassa a felhasználókat. Tehát legyen joga és lehetősége új felhasználó felvételére, egy felhasználó adatainak és felhasználói csoportjának módosítására, valamint egy felhasználó törlésére.

- Eszköz nyilvántartási modul hozzáfűzése az alkalmazáshoz. Ez az adatbázisban új felhasználói csoporto(k) hozzáfűzését, új adattáblák kialakítását jelenti. Új SQL lekérdezéseket kell összeállítani az eszközök adatainak az eléréséhez és módosításához. Új panelek kialakítása szükséges, melyeken keresztül a hozzáféréssel rendelkező felhasználók kezelni tudják az adatbázisban az eszközök adatait.

A fejlesztői dokumentációban az eszköz nyilvántartási modul hozzáfűzését megvalósító lépéseket részletesebben is kifejtem.

- További modulok hozzáfűzése (Lásd a 2. oldal tetején látható listát).

2 Fejlesztői dokumentáció

2.1 Témaválasztás

Egy vállalkozás sikerességének egyik fontos feltétele, hogy működése fennakadás mentes legyen. Ezt a dolgozók közötti kommunikáció minősége nagy mértékben befolyásolja. Ezért lényeges, hogy a feladatok megszabásához, a megoldások ismertetéséhez, a közös munka végzéséhez és az eredmények értékeléséhez rendelkezzenek egy közös módszerrel. Ezt a módszert képezheti le egy számítógépes program, mely egy projekttel kapcsolatosan az összes utasítást és magyarázatot, az egyedi véleményeket, a megszerzett ismereteket és az elvégzett tevékenységeket egy helyen tárolja és elérhetővé teszi.

Egy ilyen alkalmazás biztosítja a megvalósítás nyomon követhetőségét és a befejezés utáni kiértékelést is jelentősen megkönnyíti. Az adatbázis „napra kész” állapota miatt az esetleg kieső résztvevők könnyebben helyettesíthetők.

Természetesen ez a programként fejlesztett módszer is csak akkora mértékben hatékony, amekkora mértékben alkalmazzák. A felhasználóknak maguknak is fel kell ismerniük a tájékoztatási rendszer hasznosságát, el kell fogadniuk az alkalmazását és folyamatosan használniuk is kell az általa nyújtott lehetőségeket.

A kapcsolattartás sokféle lehetőségét kínálják a különböző közösségi hálózatok és levelező programok. Ezek a megoldások azonban nem alkalmasak vállalati környezetben a feladatvégzéssel kapcsolatos információk biztonságos megosztására. Erre csak a vállalat saját informatikai környezetében működő zárt hálózati rendszer lehet alkalmas.

A felsorolt gondolatok és érvelések figyelembevételével fogalmazódott meg a kisvállalati információs rendszer gerincének az ötlete. Az információs rendszer alap változata egy levelezésszerű kapcsolattartási megoldást kínál. A felhasználók a programban feljegyzések segítségével dokumentálják tevékenységüket. Segítségével a munkatársak könnyen hozzájutnak az ismert adatokhoz és a naprakész információk alapján gyorsan és helyesen tudnak dönteni.

A programot úgy terveztem, hogy új igények felmerülése esetén egyszerűen módosítható és bővíthető legyen.

2.2 Az alkalmazott fejlesztői eszközök

A program fejlesztéséhez a JAVA programozási nyelvet választottam. A JAVA programok a legnépszerűbb operációs rendszereken futtathatók, így platform függetlennek mondhatók.

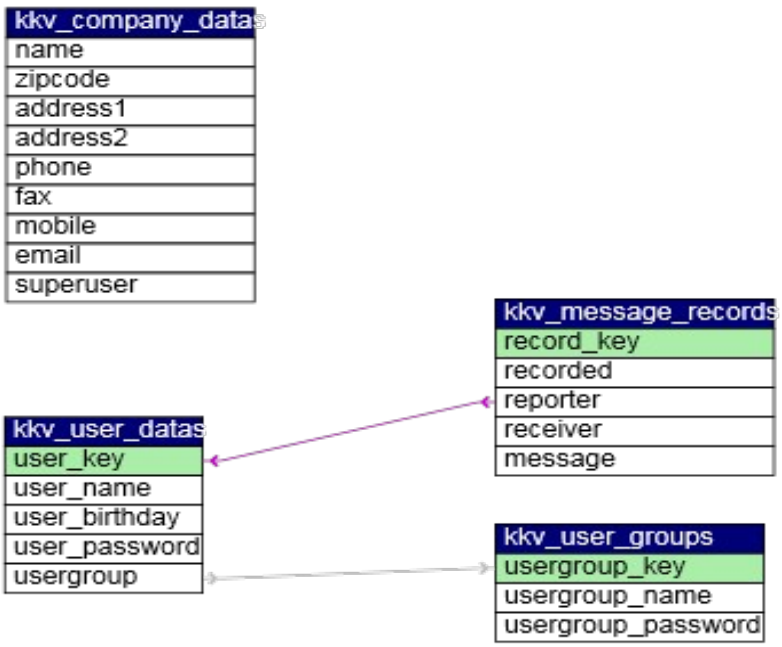
A fejlesztéshez szükséges környezetet egy Virtualboxban futó Windows 7 operációs rendszeren alakítottam ki. Ebbe telepítettem a JAVA kód fejlesztéséhez a NetBeans IDE 8.2-es keretrendszert és az adatbázis kezeléshez a szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomagot, az XAMPP-t.

A képek, ikonok és ábrák készítéséhez és módosításához a GIMP képszerkesztő programot együtt használtam a rendszer saját képernyőkép készítőjével. A folyamatábrák esetén a Dia program lehetőségeit aknáztam ki. Az adatmodell képeit a phpMyAdmin adatbázis-kezelő „Designer” menüpontjából exportáltam.

2.3 Az adatmodell leírása

2.3.1 *Az adatbázis*

Az alkalmazás a kkv_sample nevű adatbázist használja az adatok tárolására. A bemutatott változatban négy tábla található (13. ábra).



13. Ábra: A kkv_sample adatbázis táblái

2.3.2 A vállalat adatai tábla

A kkv_company_data tábla (14. ábra) a kisvállalat adatait tartalmazza. Az oszlopok tartalma egyértelmű (név, irányítószám, cím1, cím2, telefon, fax, mobil, email), egyedül a superuser oszlop szorul magyarázatra. A vállalat adatait módosítani jogosult felhasználói csoport (kkv_user_groups) kulcsát (usergroup_key) adja meg.

```
/* COMPANY DATAS ------------------------------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020         */
/* KKV COMPANY DATAS:                    */
/* NAME, ADDRESS, PHONES, MAIL           */
CREATE TABLE `kkv_company_datas` (
  name VARCHAR(60) NOT NULL,
  zipcode INT NOT NULL,
  address1 VARCHAR(60) NOT NULL,
  address2 VARCHAR(60) NOT NULL,
  phone VARCHAR(20) NOT NULL,
  fax VARCHAR(20) NOT NULL,
  mobile VARCHAR(20) NOT NULL,
  email VARCHAR(60) NOT NULL,
  superuser INT DEFAULT 2
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;
```

14. Ábra: A kkv_company_data tábla szerkezete

2.3.3 A felhasználói csoportok tábla

A kkv_user_groups tábla (15. ábra) az alapja a felhasználók jogosultsági besorolásának. A csoport név (usergroup_name) egyben az adatbázis eléréséhez szükséges felhasználónév, míg a csoport jelszó (usergroup_password) az adatbázis eléréséhez szükséges belépési jelszó.

```
/* USEGROUPS        ------------------------------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020         */
/* KKV USERGROUPS:                       */
/* USER PRIVILEGES AND MYSQL GRANTS      */
CREATE TABLE `kkv_user_groups` (
  usergroup_key INT AUTO_INCREMENT,
  usergroup_name VARCHAR(40) NOT NULL,
  usergroup_password varchar(255) NOT NULL,
  CONSTRAINT pk_usergroup PRIMARY KEY (usergroup_key)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;
```

15. Ábra: A kkv_user_groups tábla szerkezet

2.3.4 A felhasználók adatai tábla

A kkv_user_datas tábla (16. ábra) az adatbázis elérésére jogosult felhasználók adatait tartalmazza. A név (user_name) és születési dátum (user_birthday) egyértelmű-

en meghatározza a felhasználót. A jelszó (user_password) az alkalmazásba – és *nem* az adatbázisba - történő belépéshez szükséges. A felhasználói csoport (usergroup) pedig megadja, hogy melyik csoport tagjaként jogosult az adatbázist használni. A sorban az első felhasználó a „BroadCastUser”, a neki címzett feljegyzéseket, mint körüzeneteket, mindenki láthatja.

```
/*            USERS ---------------------------------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020         */
/* KKV USER DATAS                        */
/* NAME, BIRTHDAY, USERGROUP             */
CREATE TABLE `kkv_user_datas` (
  user_key INT AUTO_INCREMENT,
  user_name VARCHAR(40) NOT NULL,
  user_birthday DATE NOT NULL,
  user_password varchar(255) NOT NULL,
  usergroup INT,
  CONSTRAINT pk_user PRIMARY KEY (user_key, user_name)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;

/* SET FOREIGN KEY IN KKV USER DATAS     */
ALTER TABLE `kkv_user_datas` ADD FOREIGN KEY(usergroup)
  REFERENCES `kkv_user_groups`(usergroup_key);
```

16. Ábra: A kkv_user_datas tábla szerkezete

2.3.5 *A feljegyzések tábla*

A kkv_message_records tábla (17. ábra) tartalmazza az írás idejét (recorded), az írót (reporter), a címzettet (reciever) és a feljegyzést (message).

```
/*                   RECORDS - MESSAGES ------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020         */
/* KKV RECORDS                           */
/* TOOL, QUANTITY                        */
CREATE TABLE `kkv_message_records` (
  record_key INT AUTO_INCREMENT,
  recorded DATETIME DEFAULT CURRENT_TIMESTAMP,
  reporter INT NOT NULL,
  receiver INT,
  message TEXT,
  CONSTRAINT pk_comment PRIMARY KEY (record_key)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;

/* SET FOREIGN KEY IN KKV COMMENT        */
ALTER TABLE `kkv_message_records` ADD FOREIGN KEY(reporter)
  REFERENCES `kkv_user_datas`(user_key);
```

17. Ábra: A kkv_message_records tábla szerkezete

2.3.6 A felhasználók

Az adatbázis használatához különböző hozzáféréssel rendelkező csoportok jogosultak.

Az adatbázis biztonságát szem előtt tartva létre kell hozni egy „root” szintű adminisztrátor felhasználót, majd törölni kell a telepítés során alapértelmezésben bállított jelszó nélkül root felhasználó valamennyi példányát.

Az információs rendszer adminisztrátorát „rootschield” -nak kereszteltem és minden hálózatról teljes hozzáférést biztosítottam neki (18. ábra).

```
GRANT ALL PRIVILEGES ON *.* TO 'rootschield'@'%' IDENTIFIED BY PASSWORD
'*609F49858EF550B6556A011E4E551F69383453B5' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rootschield'@'127.0.0.1' IDENTIFIED BY PASSWORD
'*609F49858EF550B6556A011E4E551F69383453B5' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rootschield'@'::1' IDENTIFIED BY PASSWORD
'*609F49858EF550B6556A011E4E551F69383453B5' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'rootschield'@'localhost' IDENTIFIED BY PASSWORD
'*609F49858EF550B6556A011E4E551F69383453B5' WITH GRANT OPTION;
```

18. Ábra: A "rootschield" felhasználói csoport

Az információs rendszer adatbázis elérhetőségét a programban a „testuser” hozzáférésével tesztelem. Ez a felhasználói tesztcsoport rendelkezik a legalacsonyabb jogosultságokkal (19. ábra), de minden hálózatról hozzáféréssel bír.

```
GRANT USAGE ON *.* TO 'testuser'@'%' IDENTIFIED
  BY PASSWORD '*3A2EB9C80F7239A4DE3933AE266DB76A7846BCB8';
GRANT SELECT (user_password, user_name, usergroup, user_key)
  ON `kkv_sample`.`kkv_user_datas` TO 'testuser'@'%';

GRANT USAGE ON *.* TO 'testuser'@'127.0.0.1' IDENTIFIED
  BY PASSWORD '*3A2EB9C80F7239A4DE3933AE266DB76A7846BCB8';
GRANT SELECT (user_password, user_name, usergroup, user_key)
  ON `kkv_sample`.`kkv_user_datas` TO 'testuser'@'127.0.0.1';

GRANT USAGE ON *.* TO 'testuser'@'::1' IDENTIFIED
  BY PASSWORD '*3A2EB9C80F7239A4DE3933AE266DB76A7846BCB8';
GRANT SELECT (user_password, user_name, usergroup, user_key)
  ON `kkv_sample`.`kkv_user_datas` TO 'testuser'@'::1';

GRANT USAGE ON *.* TO 'testuser'@'localhost' IDENTIFIED
  BY PASSWORD '*3A2EB9C80F7239A4DE3933AE266DB76A7846BCB8';
GRANT SELECT (user_password, user_name, usergroup, user_key)
  ON `kkv_sample`.`kkv_user_datas` TO 'testuser'@'localhost';
```

19. Ábra: A "testuser" felhasználói csoport

A következő csoport az „employee” alkalmazotti csoport, mely a feljegyzések írására és a neki szóló feljegyzések olvasására jogosult (20. ábra). Az ebbe a csoportba tartozó felhasználók csak helyi hálózatról, azaz munkahelyi gépről léphetnek be.

```
GRANT USAGE ON *.* TO 'employee'@'localhost' IDENTIFIED
  BY PASSWORD '*334AD31E3FB6986ED62F9C8D2F252E2FC475D5FA';
GRANT SELECT, INSERT
  ON `kkv_sample`.`kkv_message_records` TO 'employee'@'localhost';
```

20. Ábra: A "employee" felhasználói csoport

A ranglétra legmagasabb fokán álló „director” irányító csoport betekinthet az összes feljegyzésbe és írhat saját feljegyzéseket (21. ábra). Az ebbe a csoportba tartozó felhasználók csak helyi hálózatról, azaz munkahelyi gépről léphetnek be.

```
GRANT USAGE ON *.* TO 'director'@'localhost' IDENTIFIED
  BY PASSWORD '*1ABF1A3C7938A769F31C955E3D5CFC2FA2007598';
GRANT SELECT (user_name, user_birthday, usergroup, user_key),
  INSERT ON `kkv_sample`.`kkv_user_datas` TO 'director'@'localhost';
GRANT SELECT (reporter, message, receiver, recorded)
  ON `kkv_sample`.`kkv_message_records` TO 'director'@'localhost';
```

21. Ábra: A "director" felhasználói csoport

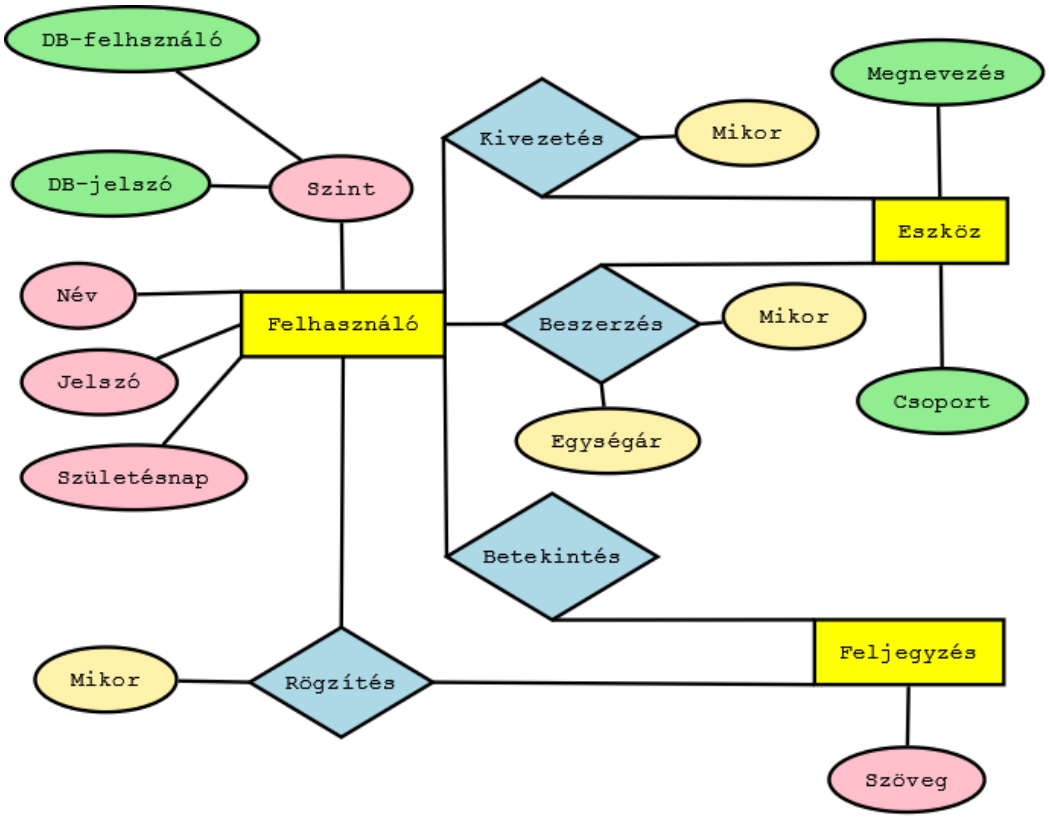
A rendszer majdnem adminisztrátora(i) a superuser(ek). A csoport (22. ábra) jogosult a felhasználói adatok módosítására és ezen felül rendelkezik az irányító csoport jogaival is.

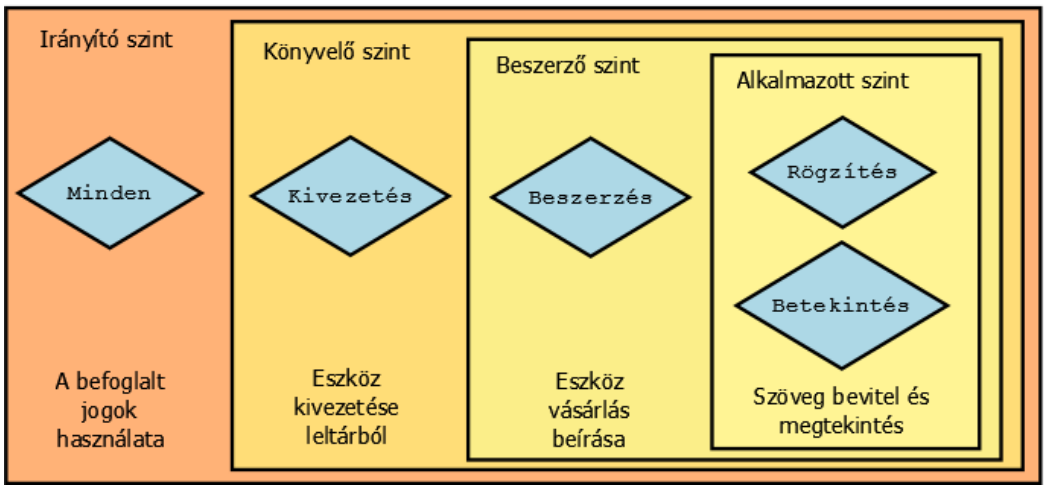
```
GRANT USAGE ON *.* TO 'superuser'@'%' IDENTIFIED
  BY PASSWORD '*4FB15991C85A2EB8A3B4B2D3411AA2AD094E7229';
GRANT SELECT, INSERT ON `kkv_sample`.* TO 'superuser'@'%';
GRANT UPDATE (zipcode, address1, fax, email, phone, mobile, name, address2)
  ON `kkv_sample`.`kkv_company_datas` TO 'superuser'@'%';
GRANT UPDATE (user_password, user_birthday, usergroup, user_name)
  ON `kkv_sample`.`kkv_user_datas` TO 'superuser'@'%';
GRANT UPDATE (usergroup_name)
  ON `kkv_sample`.`kkv_user_groups` TO 'superuser'@'%';
GRANT SELECT, INSERT
  ON `kkv_sample`.`kkv_message_records` TO 'employee'@'localhost';

GRANT USAGE ON *.* TO 'superuser'@'localhost' IDENTIFIED
  BY PASSWORD '*4FB15991C85A2EB8A3B4B2D3411AA2AD094E7229';
GRANT SELECT, INSERT ON `kkv_sample`.* TO 'superuser'@'localhost';
GRANT UPDATE (zipcode, address1, fax, email, phone, mobile, name, address2)
  ON `kkv_sample`.`kkv_company_datas` TO 'superuser'@'localhost';
GRANT UPDATE (user_password, user_birthday, usergroup, user_name)
  ON `kkv_sample`.`kkv_user_datas` TO 'superuser'@'localhost';
GRANT UPDATE (usergroup_name)
  ON `kkv_sample`.`kkv_user_groups` TO 'superuser'@'localhost';
GRANT SELECT, INSERT
  ON `kkv_sample`.`kkv_message_records` TO 'employee'@'localhost';
```

22. Ábra: A "superuser" felhasználói csoport

2.3.7 *Az adatbázis fejlesztése*

Az eszköznyilvántartó modullal történő fejlesztése után az adatbázis szerkezetét, a kapcsolatokat és az egységeket a 23. ábrán látható E-K diagram szemlélteti. Az ábra alsó részén az egymásba ágyazott kapcsolati elemek a felhasználói szintek rangsor szerinti végrehajtási jogait mutatják az irányítói szinttel bezárólag.





23. Ábra: A bővített adatbázis E-K diagramja és a hozzáférési besorolások

A fejlesztés során tehát három táblázattal és két felhasználói csoporttal bővül az adatbázis. Az új csoportokhoz fűződő a jogosultságokat a fejlesztés során az irányító és superuser felhasználók is megkap(hat)ják.

2.3.7.1 Az eszköz csoportok tábla

A kkv_tool_groups tábla (24. ábra) az eszközökhöz rendelhető csoportneveket (toolgroup_name) tartalmazza.

```
/* TOOLGROUPS                 ----------------------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020         */
/* KKV TOOLGROUPS:                       */
/*                                       */
CREATE TABLE `kkv_tool_groups` (
  toolgroup_key INT AUTO_INCREMENT,
  toolgroup_name VARCHAR(255) NOT NULL,
  CONSTRAINT pk_toolgroups PRIMARY KEY (toolgroup_key)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;
```

24. Ábra: A kkv_tool_groups tábla szerkezete

2.3.7.2 Az eszközök adatai tábla

A kkv_tool_datas tábla (25. ábra) a nyilvántartásba vehető eszközök adatait tárolja. Az eszköz nevét, csoportját és a minimális darabszámot.

```
/*            TOOLS          ----------------------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020         */
/* KKV TOOL DATAS                        */
/* NAME, TOOLGROUP                       */
CREATE TABLE `kkv_tool_datas` (
  tool_key INT AUTO_INCREMENT,
  tool_name VARCHAR(255) NOT NULL,
  toolgroup INT,
  tool_min_quantity INT,
  CONSTRAINT pk_tool PRIMARY KEY (tool_key)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;
```

25. Ábra: A kkv_tool_datas tábla szerkezete

2.3.7.3 Az eszköznyilvántartás tábla

A kkv_tool_records tábla (26. ábra) adataiban a nyilvántartási folyamatot tükröződik. A nyilvántartásba vétel és selejtezés dátumát, a könyvelőre mutató kulcsot, az eszközre utaló kulcsot, a beszerzési árat és egy megjegyzést foglal egy adatsorba.

```
/*                    RECORDS - TOOLS ---------------------------------------------*/

/* KKV SAMPLE DATABASE 2019-2020          */
/* KKV RECORDS                            */
/* TOOL, QUANTITY                         */
CREATE TABLE `kkv_tool_records` (
  record_key INT AUTO_INCREMENT,
  recorded DATETIME DEFAULT CURRENT_TIMESTAMP,
  rollout DATE,
  reporter INT NOT NULL,
  tool INT,
  price INT,
  notes TEXT,
  CONSTRAINT pk_comment PRIMARY KEY (record_key)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;
```

26. Ábra: A kkv_tool_records tábla szerkezete

2.3.7.4 Az új felhasználó csoportok

Az eszköznyilvántartó modul két felhasználói csoporttal fejleszti az adatbázist. Ez a két csoport a beszerzői – shopper – csoport (27. ábra) és az eszközök nyilvántartásával foglalkozó könyvelői – accountant – csoport (28. ábra).

```
CREATE USER IF NOT EXISTS 'shopper'@'localhost' IDENTIFIED BY 'shoppermysql';
GRANT SELECT
    ON `kkv_sample`.`kkv_tool_records` TO 'shopper'@'localhost';
GRANT INSERT (`recorded`, `reporter`, `tool`, `price`, `notes`)
    ON `kkv_sample`.`kkv_tool_records` TO 'shopper'@'localhost';
GRANT SELECT (`recorded`, `reporter`, `receiver`, `message`)
    ON `kkv_sample`.`kkv_message_records` TO 'shopper'@'localhost';
```

27. Ábra: A "shopper" felhasználói csoport

```
CREATE USER IF NOT EXISTS 'accountant'@'localhost' IDENTIFIED BY 'accountantmysql';
GRANT SELECT (`user_key`, `user_name`, `user_birthday`, `usergroup`), INSERT
    ON `kkv_sample`.`kkv_user_datas` TO 'accountant'@'localhost';
GRANT SELECT, INSERT
    ON `kkv_sample`.`kkv_tool_datas` TO 'accountant'@'localhost';
GRANT SELECT, INSERT
    ON `kkv_sample`.`kkv_tool_records` TO 'accountant'@'localhost';
GRANT SELECT (`recorded`, `reporter`, `receiver`, `message`)
    ON `kkv_sample`.`kkv_message_records` TO 'accountant'@'localhost';
```

28. Ábra: Az "accountant" felhasználói csoport

Ezek a csoportok az alkalmazotti csoport jogosultságain felül az eszközökhöz fűződő adatok bejegyzésére és egyes adatok módosítására is jogosultak.

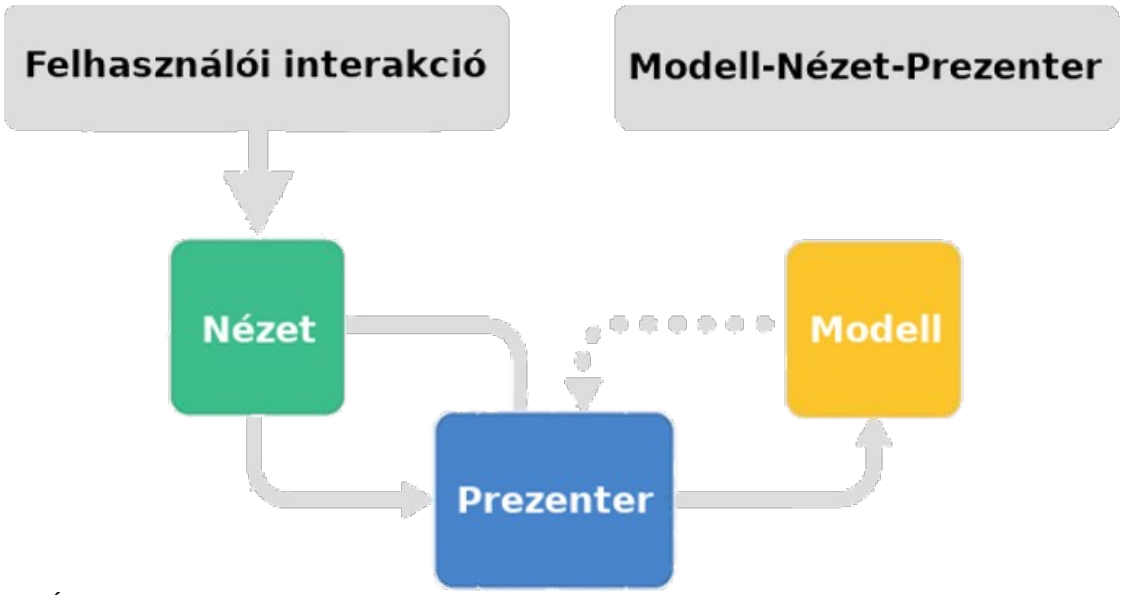
2.4 Részletes feladatspecifikáció, algoritmusok

2.4.1.1 A szoftvertervezési minta

A programozás során a Modell-Nézet-Prezenter szoftvertervezési mintát alkalmaztam (angolul: Modell-View-Presenter, röviden: MVP), mivel ez felelt meg legjobban a feltételként támasztott modulos bővíthetőségnek. A minta rétegeinek jól körülhatárolt szerepe segíti a kód nyomonkövethetőségét.

Az MVP-modell főleg olyan alkalmazásokban nyújt jelentős előnyöket – hasonlóan az MVC-hez –, ahol komplex adathalmazokon kell műveleteket végezni, és ezek eredményeit a felhasználó elé tárni. Az MVP modell három rétegre osztja az alkalmazást. Mindhárom rétegnek jól körülírható feladatai vannak, és csak a szomszéd réteggel kommunikálhatnak. Ez nagyfokú rugalmasságot ad: a két szélső komponens egymás tudta nélkül bármikor lecserélhető.[2]

Az MVP minta három rétege közötti kommunikációt a 29. ábra mutatja be.



29. Ábra: A Modell-Nézet-Prezenter szoftverfejlesztési minta

2.4.1.2 A program forráskódjának szerkezete

A java programkódot a program szerkezetét tükröző csomagokba rendeztem. Az ***aBasis*** csomagba az alaposztályok kerültek, melyek a program futása során az adatbázis és a nézet elemekkel történő interakció során kerülnek meghívásra a kontrollerből. Az ***aControl*** csomag tartalmazza a program gerincét, az ablakot megvalósító fő

[2] Részletek a Vikipediából

keretet és a kontrollert/prezentert alkotó irányító osztályokat. Az ***aDatas*** csomagban a program indítási adatait tartalmazó szöveges fájlok és az azok beolvasását szolgáló java osztályok találhatók. Az ***aGlobal*** csomagban az alkalmazásban használt szövegek és beállítások változóit tartalmazó osztályok kaptak helyet. Az ***aSources*** csomag tartalmazza az ikonok, hátterek és más grafikus elemek képeit. Az ***aSurfaces*** csomag pedig a grafikus felhasználói felület nézeteit alkotó panelosztályoknak a helye.

2.4.1.3 Az alkalmazás grafikus felhasználói felületének a felépítése

A program grafikus felületét a XxXx00MainFrame.java keret osztály valósítja meg. A keretre egy menüsáv, egy eszköztár és egy panel került. A panelt indításkor az XxXx00Graphity.java panel osztály írja le. Ez egy egyszerű panel, mely a program egyes részeihez más-más grafikus hátteret jelenít meg.

Ezt a panelt az alkalmazás fő részében egy füles panel cseréli le az XxXx00-ContolPanel.java osztály, melyen mindig csak a bejelentkezett felhasználóhoz kapcsolódó lapok jelennek meg. Ezek

az XxXx01GetUserPanel.java osztály, a felhasználó név bekérésére,

az XxXx01GetLoginPanel.java osztály, a jelszó bekérésére,

az XxXx02MsgViewPanel.java osztály, a feljegyzések megjelenítésére,

és az XxXx02MsgPushPanel.java osztály, egy új feljegyzés rögzítésére.

A program későbbi fejlesztése során ide kerülnek az eszközök lekérdezését, leltárba vételét és selejtezését megoldó panelek.

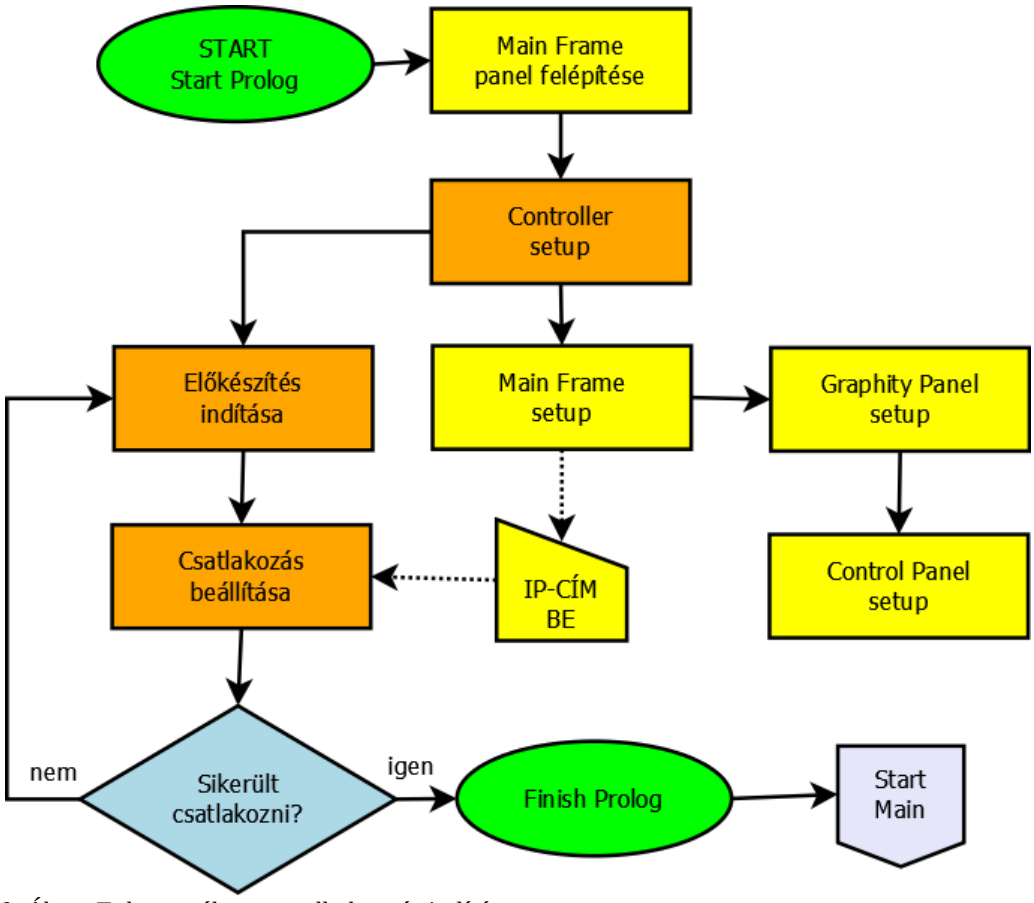
2.4.1.4 Az alkalmazás felosztása

Az alkalmazást három fő részre osztottam. A bevezető és befejező rész párhuzamos szálakban futó programrészeket tartalmaz. A program törzse tematikusan bővíthető, azaz továbbfejlesztésre alkalmas összetevőkből áll.

2.4.1.5 A bevezető rész

A bevezető rész felépíti a grafikus felhasználói felületet és ezzel párhuzamosan (egy külön szálban) ellenőrzi az adatbázis elérhetőségét (30. ábra). Amennyiben az adatbázist megtalálja, automatikusan tovább lép a fő program részbe.

Amennyiben nem a *localhoston* helyeztük el az adatbázist, az eszköztár jobb szélén található „felhő”-gombhoz tartozó ablakban megadhatjuk az IP-címét.



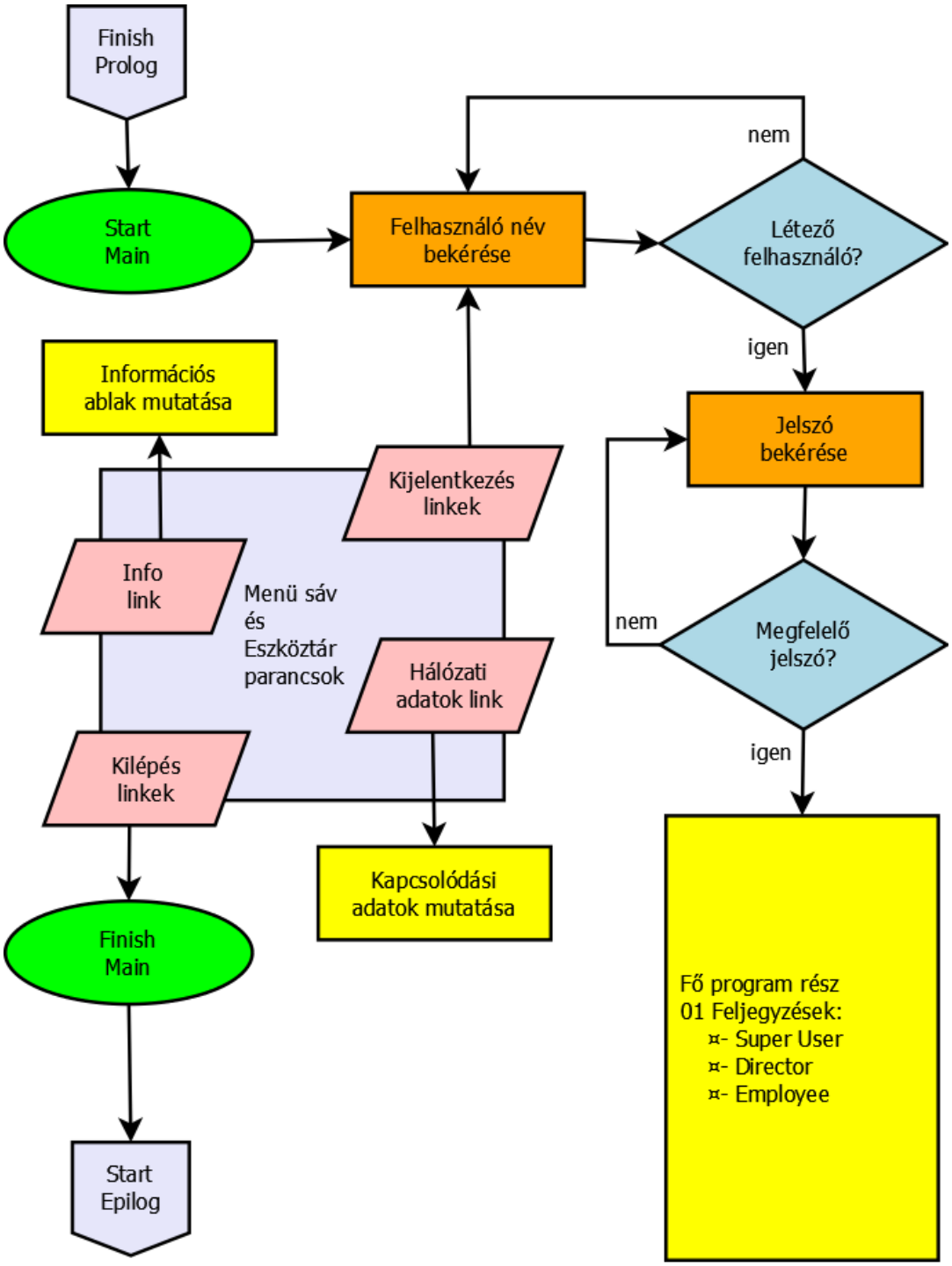
30. Ábra: Folyamatábra - az alkalmazás indítása

2.4.1.6 A fő programrész

A program gerince a felhasználó bejelentkezésével indít, majd a felhasználó azonosítása után a felhasználóhoz tartozó munkafelületet jeleníti meg. Ezután a felhasználó interakciói alapján elvégzi a kívánt adatbázissal kapcsolatos műveleteket azonnal elérhetővé téve az eredményt (31. ábra).

Az ablak alapfunkciói mellett a menük és az eszköztár parancsai biztosítják az alkalmazás átjárhatóságát. Itt tudunk kijelentkezni és kilépni is, valamint információkat és adatokat megtekinteni.

Ehhez a programrészhez lehet modul rendszerben hozzárendelni új funkciókat a hozzájuk tartozó új felhasználói jogosultságokkal, adatbázis táblákkal és SQL parancsokkal, valamint nézetekkel. A már beillesztett bejelentkezési és feljegyzések modul példája mutatja a bővítési folyamat során alkalmazható megoldásokat. A kialakított minta alkalmas az agilis szemléletű programfejlesztésre.



31. Ábra: Folyamatábra - az alkalmazás gerince

2.4.1.7 A befejező harmadik rész

A harmadik rész feladata a program befejezése. Ez történhet azonnali hatállyal, illetve animált ablakbezárással (Az XxXx01Finale.java fájl egy külön szálon futó esemény lezajlása után lép tovább az ablakot bezáró metódusra).

2.4.2 Az alkalmazás csomagjai

2.4.2.1 Az aBasis csomag osztályai

Az XxXx01DataObject.java osztály: fejlesztésre fenntartva.

Az XxXx01DbThread.java osztály: az adatbázis elérhetőségét vizsgáló szál.

Az XxXx01Finale.java osztály: a befejező animációt megvalósító szál.

Az XxXx01MySqlResult.java osztály: az adatbázis lekérdezés eredményét feldolgozó osztály.

Az XxXx01Sprite.java osztály: egy mozgatható grafikus elemet megvalósító szál. A mozgatást megvalósító funkciókat nem használtam ebben az alkalmazásban.

Az XxXx01SqlAdmin.java osztály: Az adatbázissal történő kapcsolatot megvalósító osztály. A lekérdezéseket végrehajtó egyetlen metódus kezeli mind az eredménykészlettel visszatérő, mind az adatbázist módosító SQL parancsokat.

Az XxXx01UserGroup.java osztály: az adatbázis kezelésére jogosult felhasználói csoport osztálya.

2.4.2.2 Az aControl csomag osztályai

Az XxXx00Control.java osztály: az alap kontroller/prezenter osztály.

Az XxXx00MainFrame.java osztály: Az alkalmazás ablakát és indítását megvalósító java keret osztály.

Az XxXx01Control.java osztály: A kontroller osztály első bővítménye. A felhasználó név és jelszó bekérését és ellenőrzését valósítja meg.

Az XxXx01Queries.java osztály: A felhasználó név és jelszó ellenőrzéséhez szükséges adatbázis lekérdezések parancsait olvassa be és hajtja végre.

Az XxXx02Control.java osztály: A kontroller osztály második bővítménye. A feljegyzésekkel kapcsolatos interakciókat valósítja meg.

Az XxXx02Queries.java osztály: A feljegyzésekkel kapcsolatos interakciókhoz szükséges adatbázis lekérdezések parancsait olvassa be és hajtja végre.

Az XxXxDialogs.java osztály: Felugró információs ablakok megjelenítését megvalósító osztály.

2.4.2.3 Az aDatas csomag osztályai

Az XxXx00Datas.java osztály: fejlesztésre fenntartva.

Az XxXx00Queries.java osztály: beolvassa az adatbázisból a felhasználó ne-

veket egy szöveg tömbbe. Az alap kontrollerhez kapcsolódik.

Az XxXx00Queries.txt fájl: a felhasználó nevekkel kapcsolatos SQL lekérdezési parancsokat tartalmazó szöveges fájl. Az alap kontrollerhez kapcsolódik.

Az XxXx01List.txt fájl: a felhasználói csoportokat tartalmazza.

Az XxXx01Queries.txt fájl: a felhasználói csoportokkal kapcsolatos SQL lekérdezési parancsokat tartalmazó szöveges fájl. A kontroller első bővítményéhez tartozik.

Az XxXx02Queries.txt fájl: a feljegyzésekkel kapcsolatos SQL lekérdezési parancsokat tartalmazó szöveges fájl. A kontroller második bővítményéhez tartozik

Az XxXxReadDatas.java osztály: fejlesztésre fenntartva.

Az XxXxReadList.java osztály: a felhasználói csoportokat beolvasó osztály.

Az XxXxReadQueries.java osztály: az SQL lekérdezési parancsokat beolvasó osztály.

Az XxXxSaveLogs.java osztály: fejlesztésre fenntartva.

Az XxXx__Queries.txt fájl: fejlesztésre fenntartva.

2.4.2.4 Az aGlobal csomag osztályai

Az XxXx00Global.java osztály: Az alap kontrollerhez kapcsolódó szöveges adatokat (feliratok) és numerikus globális változók beállításait tartalmazza.

Az XxXx01Global.java osztály: A kontroller első bővítményéhez kapcsolódó szöveges adatokat (feliratok) és numerikus globális változók beállításait tartalmazza.

Az XxXx02Global.java osztály: A kontroller második bővítményéhez kapcsolódó szöveges adatokat (feliratok) és numerikus globális változók beállításait tartalmazza.

2.4.2.5 Az aSources csomag

Az alkalmazásban felhasznált és felhasználni tervezett grafikai elemeket tartalmazza. Ezek gif, jpg és png típusú képek.

2.4.2.6 Az aSurfaces csomag osztályai

Az XxXx00ControlPanel.java osztály: a felhasználói tevékenységet kiszolgáló paneleket befogadó színpad.

Az XxXx00GraphityPanel.java osztály: az alkalmazás kinézetét tartalmazó grafikai panel. A program minden részéhez külön háttér és animáció rendelhető hozzá.

Az alap kontroller osztályhoz kapcsolódik.

Az XxXx01GetUserPanel.java osztály: a felhasználó nevének bekérésére szolgáló fül panel osztálya. A kontroller első bővítményéhez kapcsolódik.

Az XxXx01LoginPanel.java osztály: a jelszó bekérésére szolgáló fül panel osztálya. A kontroller első bővítményéhez kapcsolódik.

Az XxXx02MsgPushPanel.java osztály: a feljegyzés bevitelére és rögzítésére szolgáló fül panel osztálya. A kontroller második bővítményéhez kapcsolódik.

Az XxXx02MsgViewPanel.java osztály: a feljegyzésekbe történő betekintésre szolgáló fül panel osztálya. A kontroller második bővítményéhez kapcsolódik.

2.5 Forráskód

2.5.1.1 Az adatbázishoz kapcsolódás

Az XxXx01SqlAdmin.java osztály az adatbázis elérését valósítja meg. Az szerint, hogy az adatbázishoz történő csatlakozáshoz az URL, felhasználónév és jelszó hármasból melyik ismert, más és más konstruktorral inicializáljuk a kapcsolat objektumot. Az inicializálás során ellenőrzésre kerül az is, hogy mely táblák állnak rendelkezésre az adatbázisban:

```
…
    private void startDb() {
        mySqlDatabases = Arrays.asList("information_schema", "mysql",
                "performance_schema", "phpmyadmin", "test");
        try (Connection conn = DriverManager.getConnection(
                                        dbUrl, userName, password);
                PreparedStatement stmt =
                            conn.prepareStatement(SHOW_DBASES_QUERY);
                ResultSet results = stmt.executeQuery()) {

            while (results.next()) {
                String database = results.getString(1);
                if (!mySqlDatabases.contains(database)) {
                    System.out.println(database);
                }
            }
            dbReady = true;
        } catch (SQLException ex) {
            dbReady = false;
            System.out.println(ex.getMessage() + "\n");
            //c.showError(ex.getMessage());
        }
        dbClosed = true;
    }
…
```

Az inicializálás során a konzolra kiírásra kerülnek az adatbázisban található felhasználói táblák nevei, a dbClosed jelzi, hogy az konstruktor lefutott, a dbReady pedig az adatbázis létezését mutatja.

2.5.1.2 Az adatbázis elérhetőségének a tesztelése

Az XxXx01DbThread.java osztály külön szálban ellenőrzi az adatbázis elérhetőségét:

```
…
public class XxXx01DbThread extends Thread {
…
…
    @Override
    public void run() {
        XxXx01SqlAdmin dbTest = new XxXx01SqlAdmin(
                                    this.URL, this.User, this.Secret);
        if (dbTest.isDbClosed()) {
    System.out.println("Thread:> " + this.getName() + " finished,");
        }
        if (dbTest.isDbReady()) {
            c.statusDbFound();
            sleepThread(MYSQL_TEST_SLEEPTIME);
            c.finishProlog(this.getName(), dbTest.isDbReady());
        } else {
            c.statusDbNotFound();
            sleepThread(MYSQL_TEST_SLEEPTIME);
            c.dbConnection();
        }
    }
…
…
}
```

2.5.1.3 Az SQL parancsok végrehajtása

Az XxXx01SqlAdmin.java osztályban az SQL parancsokat és lekérdezéseket az „executeQuery” metódus futtatja. A metódus megkapja az SQL parancsot a „PreparedStatement”-nek megfelelő szövegként és csatolt objektum listában a hozzá tartozó argumentumokat. A visszatérő értéke lekérdezés esetén az eredmények formázott listája lesz, parancs esetén a pedig a módosított sorok száma. A metódus jelenleg argumentumként csak számot és szöveges változót dolgoz föl. Ez továbbfejleszthető más típusú objektumok felismertetésére, amennyiben ez igénnyé válik.

A visszatérési értéket a kontroller a típusa (*instanceof*) alapján értékeli ki.

```
…
    private int rsColumnNumber;
    private String[] rsColumNames;
    private int[] rsMaxCellWidth;

    public Object executeQuery(String s_query, List arguments) {
        Object toReturn;
            try (Connection conn = DriverManager.getConnection(
                        this.dbUrl, this.userName, this.password);
                PreparedStatement ps = conn.
                                        prepareStatement(s_query)) {
            int index = 0;
            for (Object argument : arguments) {
                index++;
                if (argument instanceof String) {
                    ps.setString(index, (String) argument);
                        System.out.println("String> "
                                    + index + ", " + argument);
                } else if (argument instanceof Integer) {
                    ps.setInt(index, (int) argument);
                        System.out.println("Integer> "
                                    + index + ", " + argument);
                }
            }
            boolean status = ps.execute();
            if (status) {
                toReturn = new XxXx01MySqlResult();
                try (ResultSet rs = ps.getResultSet()) {
                    rsMetaData(rs);

((XxXx01MySqlResult)  toReturn).setRsNumberOfColumns(rsColumnNumber);

((XxXx01MySqlResult) toReturn).setRsNameOfColumns(rsColumNames);

((XxXx01MySqlResult) toReturn).setRsMaxSizeOfColumns(rsMaxCellWidth);

                    Object[] row;
                    while (rs.next()) {
                        row = new Object[rsColumnNumber];
                        for (int ii = 0; ii < rsColumnNumber; ii++) {
                            row[ii] = rs.getString(ii + 1);
                        }
                  ((XxXx01MySqlResult) toReturn).addRowToTable(row);
                    }
                }
            } else {
                toReturn = ps.getUpdateCount();
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
            return null;
        }
        return toReturn;
    }

…
```

2.5.1.4 Az SQL utasítások beolvasása szöveges fájlból

Az SQL utasításokat az XxXxReadQueries.java osztály olvassa be a szöveges fájlokból. A visszatérési értéke egy Map tipusú változó, mely párban tárolja a parancsok nevét és szövegét.

…

```
public class XxXxReadQueries {

    public static Map<String, String> inputFromFile(String f) {
        Map<String, String> toReturn = new HashMap<>();
        Scanner SC = new Scanner(
                XxXxReadDatas.class.getResourceAsStream(f),
CODE_PAGE);
        String s_key;
        String s_query;
        while (SC.hasNext()) {
            s_key = SC.nextLine();
            if (SC.hasNext()) {
                s_query = SC.nextLine();
                while (SC.hasNext() && !s_query.contains(";")) {
                    s_query += System.lineSeparator();
                    s_query += SC.nextLine();
                }
            } else {
                s_query = "";
            }
            toReturn.put(s_key, s_query);
        }
        //System.out.println("Queries: " + toReturn);
        return toReturn;
    }
}
```

…

2.5.1.5 A programrészek eltérő háttereinek megjelenítése

Az egyes programrészek a XxXx00GraphityPanel.java osztályban kaphatják meg különböző hátterüket. Ezt a „paintComponent” metódus felülírásával oldottam meg. Egy *enum* tipusú változóban („state”) rögzíti a program az előrehaladásának megfelelő állapotot. Majd ez alapján készít egy többszörös elágazást, mely mindegyik állapothoz hozzárendel egy-egy hátteret.

Itt adott a lehetőség szöveg és mozgó grafikus elem (sprite) háttéren történő megjelenítésre is. A kép kirajzoltatásakor az „imageObserver” megfelelő megadásával a „gif” típusú fájlokat is lejátszathatjuk.

…

```
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        switch (state) {
            case PROLOG: {
                g.drawImage(GRAPHITY_PROLOG, 0, 0,
                        this.getWidth(), this.getHeight(), this);
                String str = GRAPHITY_PROLOG_TXT;
                g.setColor(Color.BLACK);
                g.setFont(new Font("Serif", Font.BOLD, 63));
                int width = g.getFontMetrics().stringWidth(str);
                int height = g.getFontMetrics().getHeight();
                g.drawImage(SPRITE, SPRITE_X, SPRITE_Y,
                        SPRITE_WIDTH, SPRITE_HEIGHT, this);
                g.drawString(str,
                        (graphityWidth() - width) / 2,
                        (graphityHeight() - 3 * height / 4) / 2);
                break;
            }
            case EPILOG: {
                g.drawImage(GRAPHITY_FINALE, 0, 0,
                        this.getWidth(), this.getHeight(), this);
                String str = PROCESS_STATE.EPILOG.toString();
                g.setColor(Color.BLACK);
                g.setFont(new Font("Serif", Font.BOLD, 31));
                int width = g.getFontMetrics().stringWidth(str);
                int height = g.getFontMetrics().getHeight();
                g.drawString(str,
                        (graphityWidth() - width) / 2,
                        (graphityHeight() - 3 * height / 4) / 2);

                break;
            }
            case EXIT: {
                g.drawImage(GRAPHITY_FINALE, 0, 0,
                        this.getWidth(), this.getHeight(), this);
                break;
            }
            default: {
                g.drawImage(GRAPHITY_MAIN, 0, 0,
                        this.getWidth(), this.getHeight(), this);
                g.drawImage(SPRITE, SPRITE_X, SPRITE_Y,
                        SPRITE_WIDTH, SPRITE_HEIGHT, this);
            }
        }
        if (c != null) {
            c.drawGraphity(g);
        }
    }
```

…

2.6 Tesztelési dokumentáció

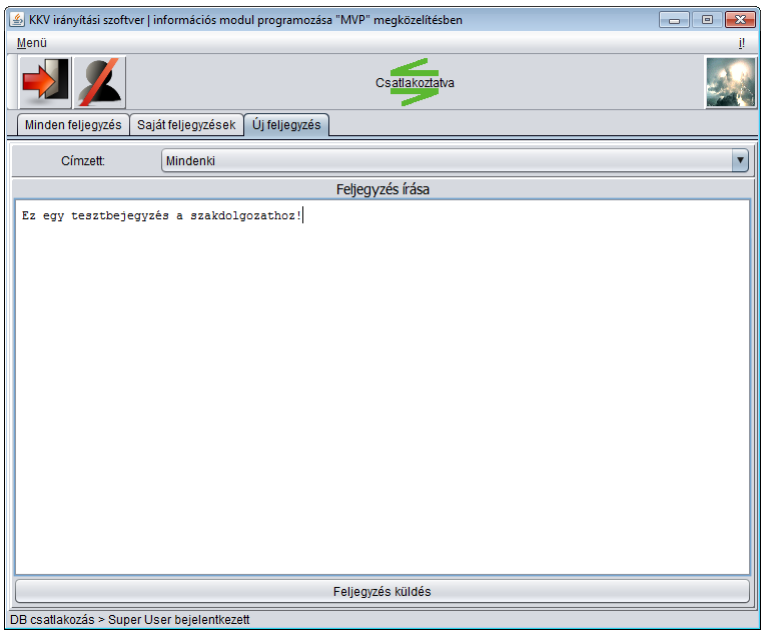
A programot fejlesztése során fontos szerepet kapott a konzolos kiíratás. A kód lényeges pontoknál megjeleníti, hogy hol tart a program és hogy az éppen használt változók milyen értékkel bírnak. Ezek fontos információk voltak a hibajavítás folyamatában. Azokban az esetekben, amikor nem az elvárásnak megfelelő eredmény jelent meg, a hibás rész egyértelműen azonosítható, így javítható volt.

A bevezető részben egy ilyen elemzés után került külön szálba az adatbázis elérhetőségét tesztelő folyamat. A hibajelenség a következő volt: kikapcsolt adatbázis szerver esetén az alkalmazás indításakor csak egy üres keret jelent meg grafikai elemek nélkül. A konzolon látható üzenetekből egyértelművé vált, hogy a folyamatos kapcsolódási kísérletekről mindaddig nem lép tovább a program, amíg a szervert be nem kapcsolják. Erről viszont semmilyen visszajelzést nem kapunk a grafikus felületen, hiszen a program nem lép tovább, így nem jeleníti meg az ablakban azt. A külön szálban elindított rész már nem akadályozza a fő szál folyamatosságát. A mellékszál pedig befejezése előtt meghívja a sikeres, vagy a sikertelen kapcsolódáshoz rendelt metódust a kontrollerben, így adva irányt a fő szál folyamatainak.

Az SQL parancsok tesztelését közvetlenül az adatbázisban végeztem. A megírt parancsokat a mellékelt V02-05kkvDB05queries.sql fájlba gyűjtöttem össze. A továbbfejlesztés részeként eddig bemutatott eszköznyilvántartáshoz is tartalmaz letesztelt utasításokat. A programba csak a sikeres működésükről történő megbizonyosodás után kerültek beillesztésre. Így adatbázissal kapcsolatos hiba esetén rögtön a java kódban lehetett keresni azt.

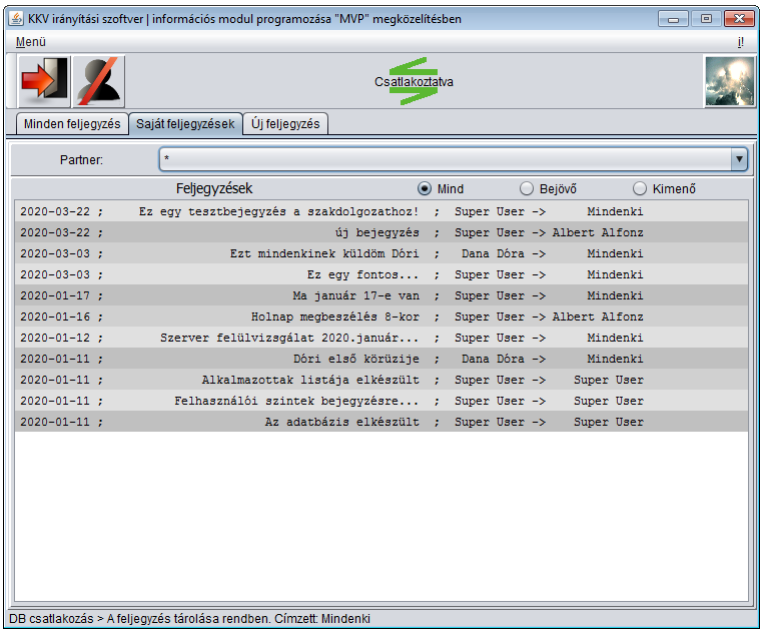
A program a futása során az állapotsorban rövid üzenetekkel tájékoztat tevékenységünk eredményéről. Az adatbázishoz történő csatlakozás után jelzi, hogy az „adatbázis rendben” elérhető. Bejelentkezéskor kiírja az érvényes felhasználó nevét, illetve hibás azonosító esetén, hogy „nincs ilyen nevű felhasználó”. Hibás jelszó esetén jelzi, hogy „hibás jelszó”-val próbáltunk belépni. A sikeres bejelentkezésről is visszajelzést kapunk (32. ábra). Új bejegyzés rögzítése után „a feljegyzés tárolása rendben” üzenetet kapjuk a címzett megjelölésével (33. ábra). A sikertelen rögzítésről is (mely szerver leállás, hálózatkiesés esetén következhet be) értesülünk, hogy „a feljegyzés tárolása sikertelen” (34. ábra). Az alkalmazás továbbfejlesztése során további releváns üzeneteket fűzhetünk a globális változókat tartalmazó java osztályokhoz.

A feljegyzések bevitelekor az új feljegyzés (32. ábra) rögtön ellenőrzésre is kerül, hiszen a program az adatbázisban történő rögzítés után rögtön végrehajt egy kiolvasást is.



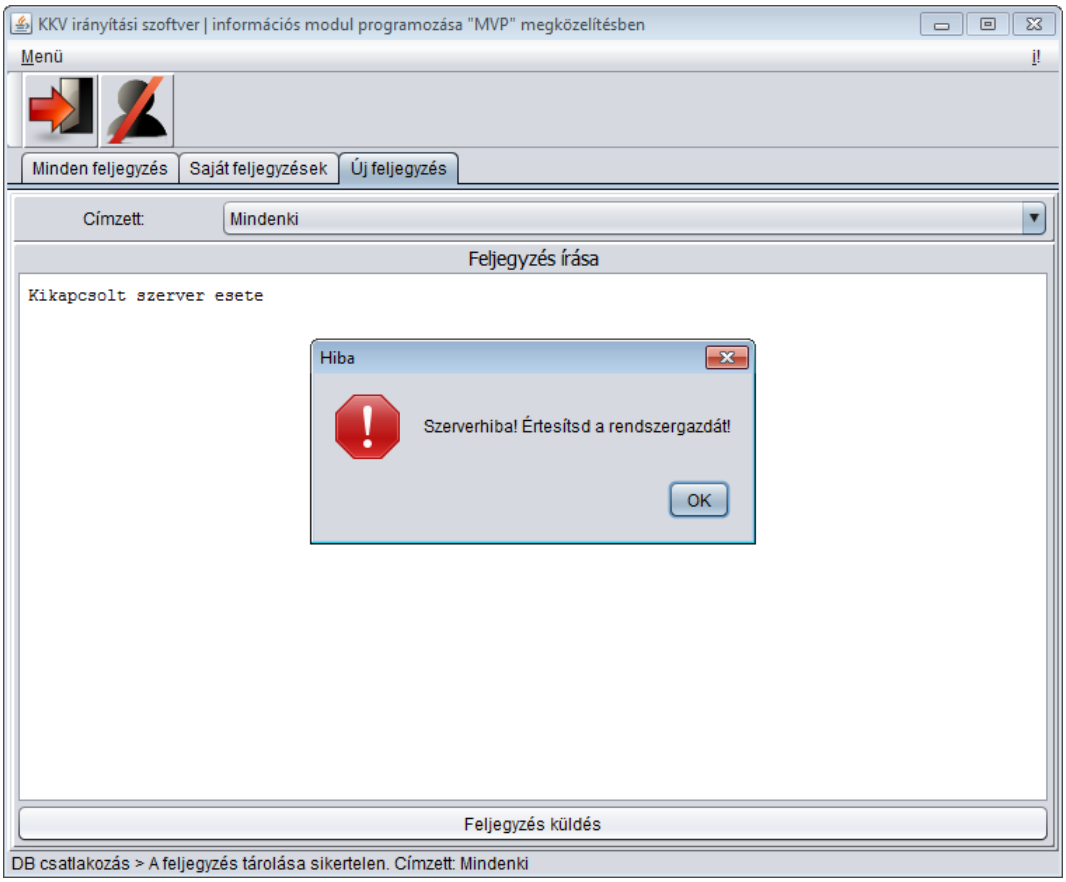
32. Ábra: Feljegyzés bevitel

A felhasználói felület pedig átvált az üzenetek megmutatása fülre, ahol az első helyen az új üzenet áll.



33. Ábra: A feljegyzés azonnal a képernyőre kerül

Amennyiben sikertelen a feljegyzés rögzítése, akkor a rendszergazdát kell értesíteni (34. ábra).



34. Ábra: Szerverhiba esete

Ha csak egyszerű szerverleállásról van szó, akkor a szerver elindítása után a program folytatja működését. Csak az eszköztárról hiányzó csatlakozás gombok figyelmeztetnek az lezajlott eseményekre (Ez az alkalmazás újraindításával orvosolható).

A továbbfejlesztés során vizsgálni kell az ilyen típusú problémák kezelését is. Hiszen az adatbázis elérhetetlensége nem ronthatja a vállalkozás működését (szinkronizált tartalék adatbázis, stb.).

A program adatbázis csatlakozásának teszteléséhez a következő változatokat végeztem el:

1. A program és az adatbázis-kezelő egy gépen futott.
2. Az adatbázis-kezelő VitualBoxban futott Windows7 alatt, a program a hoston Linux alatt.
3. A program és az adatbázis-kezelő VitualBoxban futott Windows7 alatt .

4. A program és az adatbázis-kezelő egy helyi hálózat két különböző gépén futott. Az adatbázis-kezelő Linux host alatti VirtualBox-ban, A program pedig Linux alatt egy másik gépen.
5. A program Linux alatt futott, míg a távoli adatbázis https://www.freesqldatabase.-com/ címen kapott helyet.

Az adatbázis kapcsolat minden esetben létrejött, a tesztelési adatbázis tartalma minden esetben megjelenítésre került („Super User” felhasználóval teszteltem mindet). Egy új feljegyzés bevitele is sikeresen megtörtént.

Az új üzenet írásakor az üzenet küldés funkciót át kellett alakítani, mert a nem létező „Válassz!” felhasználónak is hajlandó volt elküldeni az üzenetet, amit javítani kellett. A javítás után a küldés gomb állapotát a következő metódus állítja be:

```
…
      public void enablePush() {
        boolean b = textMessage.getText().length() >=
                                            MSGTAB_MESSAGES_MIN_LENGTH;
        b = b && !(comboTargetList.getSelectedIndex() == 0);
        btnSendMessage.setEnabled(b);
      }
…
```

A metódus meghívását a comboBox és a szövegdoboz változása is (a kontrol-leren keresztül) triggereli :

```
…
      private void textMessageKeyTyped(KeyEvent evt) {
        c.messageEnable();
      }
      private void comboTargetListActionPerformed(ActionEvent evt) {
        c.messageEnable();
      }
…
```

A felsorolt tesztek alapján a rendszer (ismert) hiba nélkül működik. A tovább-fejlesztés során különösen érzékeny pontnak számít az adatbázis hozzáférési jogosult-ságainak a megadása. Az ezeknél előforduló hibák okozták a legtöbb gondot a program üzemeltetése során.

2.7 Továbbfejlesztési lehetőségek

A továbbfejlesztés egy lehetőségét (az eszköz nyilvántartó felépítését) a dokumentációkban folyamatosan ismertettem.

Több tartalmi fejlesztés is hasonló módon hajtható végre. A fő programhoz az új tartalomnak megfelelő az elkészültektől függetlenül programozható nézet és kontroller osztályokat kell „csak” hozzáfűzni. Az új tartalmi kiegészítésekkel akár egy komplett vállalat irányítási rendszer létrehozása is célként jelölhető meg.

Az alkalmazást természetesen nem csak tartalmilag lehet fejleszteni. Tovább lehet lépni a felhőalapú adattárolás és az alkalmazás távoli elérése is irányába is. A távoli elérés alatt mind az adminisztrátori jogú távfelügyeletet, mind a nem lokális hálózatról történő felhasználást is értem.

További megfontolás tárgya lehet az alkalmazás böngészőben futó változatának az elkészítése is, ezzel az elérhető maximális függetlenséget biztosítva a felhasználók számára.

A szerverhibák, áramkimaradások és egyéb hasonló események súlyos következményekkel járnak. Ezek elkerülésére mindenképpen ki kell dolgozni egy módszert az adatok vészhelyzet esetén történő elérhetőségére.

2.8 Irodalomjegyzék, forrásmegjelölés

Oracle Corporation - NetBeans dokumentáció - https://netbeans.org/

Oracle Corpotation - Java futtatókörnyezet - https://www.java.com/en/download/

WampServer - http://www.wampserver.com/en/

XamppServer - https://www.apachefriends.org/hu/download.html

Az OpenJDK hivatalos weboldala - http://openjdk.java.net/

Stack Overflow's Network - https://stackoverflow.com/

Szabad Információs Társadalom - https://szit.hu/doku.php

Java Examples - https://www.javacodex.com/

Wikipedia:

¤ Modell-nézet-prezenter - https://hu.wikipedia.org/wiki/Modell-nézet-prezenter

¤ Java (programozási nyelv) - https://hu.wikipedia.org/wiki/Java

¤ JUnit - https://hu.wikipedia.org/wiki/JUnit

¤ Flowchart - https://en.wikipedia.org/wiki/Flowchart