DEBRECENI EGYETEM
INFORMATIKAI KAR

# WEBES ALKALMAZÁSFEJLESZTÉS

# (TELEDERMATOLÓGIAI RENDSZER FEJLESZTÉSE)

**Témavezető**: Dr. Kuki Attila
egyetemi adjunktus

**Készítette**: Varga Attila
programozó matematikus

Debrecen
2008

# Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Kuki Attilának a szakdolgozatom elkészítéséhez nyújtott útmutató tanácsaiért.
Továbbá köszönettel tartozom Kovács Zoltánnak, hogy lehetőséget biztosított és támogatta szakdolgozatom elkészítését, és Szilágyi Péternek a szakmai segítségért.
Köszönetet szeretnék mondani Komjáthi Csillának, a páromnak, hogy támogatott és mellettem állt a tanulmányaim folyamán.

# Tartalomjegyzék

# 1. Bevezetés

## *1.1. Témaválasztás*

Az egyetemi tanulmányaim mellett sikerült elhelyezkednem a Kripto Kutatásfejlesztési Kft.-nél, ahol jelenleg is dolgozom, mint kezdő szoftverfejlesztő informatikus. A dolgozatom témája egy Telederm projektnév alatt indult teledermatológiai rendszer prototípusának bemutatása, amelynek fejlesztésében lehetőségem van részt venni. A rendszer megtervezésében, dokumentációk elkészítésében, és főleg a kliens oldali szoftver kifejlesztésében.

A projekt célja egy a krónikus sebek otthoni kezelését lehetővé tevő távbőrgyógyászati rendszer prototípusának kifejlesztése, és ezzel jelentős kórházi és utazási költségek megtakarítása.

A rendszer hatékonyságát azonban elsősorban a távgyógyászati ellátási formák adják. Mivel minden adat elektronikus formában feldolgozásra kerül, a betegnek és a diagnózist megalkotó orvosnak nem szükséges egy földrajzi helyen tartózkodnia. Ha a rendszer egy perifériku egysége a beteg lakóhelyéhez közel rendelkezésre áll (természetesen speciálisan képzett egészségügyi szakdolgozók kezelésében) a rendszer egy másik végpontján pedig egy a beteg ellátásához szükséges specialista szakorvos, konzultáns található a személyes találkozás igénye nélkül diagnózis, terápiás javaslat, esetleg megfelelő ellátás is nyújtható.

## *1.2. Telemedicina*

Napjainkban rohamléptekben fejlődő számítástechnika, telekommunikáció, Internet hihetetlen gyorsasággal alakítja át világunkat, gazdaságunkat, az egész életünket. Manapság már nem szükséges az orvosnak okvetlenül a beteg mellet lennie ahhoz, hogy diagnózist állítson fel. Az Internet segítségével bárhová továbbíthatók a betegről készült dokumentumok. A beteg mellett lévő orvos vagy ápoló pedig könnyedén kérhet segítséget az akár több tízezer kilométerre lévő specialistától.

A telemedicina egy olyan gyűjtőfogalom, ami magába foglal minden távközléssel direkt módon támogatott egészségügyi gyógyító és diagnosztikai tevékenységet. Legegyszerűbben távgyógyászatnak fordíthatnánk.

## *1.3. Teledermatológia*

A telemedicina használata bőrgyógyászati konzultációra. Egy teledermatológiai konzultáció során egy bőrgyógyász szakorvos kiértékeli a kapott orvosi dokumentációt, ezek alapján diagnózist alkot, epikrízist (szakvéleményt) és terápiás javaslatot készít a páciens részére. Legegyszerűbben távbőrgyógyászatnak fordíthatnánk.

# 2. Alkalmazott technológiák és eszközök

## *2.1. Java technológia*

A Java egy objektumorientált, interpreteres, magas szintű programozási nyelv. A SUN Microsystems kezdte el fejleszteni a 1990-es évektől kezdve és napjainkban is fejleszti. A cél egy hordozható operációs rendszertől független programozási nyelv kifejlesztése volt.

A Java programozási nyelv esetében a forráskódot a fordító (compiler) egy közbülső úgynevezett Java bájt kódra fordítja, és ezt a kódot értelmezi, gépi kóddá alakítja és futtatja az interpreter (Java Virtual Machine – Java Virtuális Gép). Fordítás egyszer történik, értelmezés pedig minden futtatás alkalmával. A bájt kódot tetszőleges megfelelő verziójú JVM-el rendelkező platformon futtatható, ez biztosítja a Java programozási nyelv hordozhatóságát.

### *2.1.1. Swing*

A Java-ban a JFC (Java Foundation Classes) osztálykönyvtár biztosít a grafikus felület fejlesztéséhez eszközrendszert. A JFC két fő része az AWT és a Swing nyújt lehetőséget grafikus felület kifejlesztésére.

A Java–ban kezdetben csak az AWT (Abstract Window Toolkit) volt jelen. Mivel az AWT a grafikus komponensek megjelenítését az operációs rendszerre bízta, platformfüggő megjelenés és szűkös eszközkészlet jellemezte.

Az AWT nem valósította meg a Java egyik fő alapelvét a platformfüggetlenséget, ennek orvoslására jött létre a Swing, melynek fő célja a platformfüggetlenség biztosítása volt. A Java lehetőségeit kihasználó új komponens hierarchiát hoztak létre, ezek az új komponensek az AWT komponenseire épülnek az osztályhierarchia alapján öröklődés segítségével. A Swing tisztán Javaban van megvalósítva, platformfüggetlen megjelenés, grafikuskomponensek széles választéka, és mivel a Swing az AWT-re épül lassabb megjelenítés jellemzi.

### *2.1.2. Servlet*

Egy servlet olyan szerveroldalon futó speciális Java program, ami lehetővé teszi szerveroldalon HTML oldalak dinamikus generálását, HTML űrlapok feldolgozását, adatbázis manipulációt, a kiszolgáló funkcionalitását bővítik. Ha a kliens olyan HTML oldalt kér a kiszolgálótól, amelyet egy servlet állít elő, akkor a kiszolgáló delegálja a kérést a servlet felé, majd a servlet által generált oldalt továbbítja a kliensnek.

A `javax.servlet` csomag tartalmazza az összes servletspecifikus osztályt és interfészt. Minden servlet vagy a `javax.servlet.GenericServlet` osztály leszármazottja, vagy maga implementálja a `javax.servlet.Servlet` interfészt.

Egy protokoll független servlet a `javax.servlet.GenericServlet` alosztályt, míg egy HTTP servlet-nek a `javax.servlet.http.HttpServlet` alosztályt kell bővítenie. Minden olyan esetben, amikor a kiszolgáló egy servlet-hez irányít továbbít egy kérést, meghívja annak `service()` metódusát. Egy generikus servlet esetében felül kell írni a `service()` metódusát, hogy a kérést saját feladatának megfelelően kezelje. Egy HTTP servlet a generikustól eltérően általában nem írja felül a `service()` metódust, ehelyett a `doGet()` metódus felülírásával kezeli a GET kéréseket, és a `doPost()` metódus felülírásával a POST kéréseket.

A servlet-eket a servlet konténer hozza létre, használja és szünteti meg. Egy servlet életciklusa az alábbi három eseményből áll:

1. A servlet *betöltése és példányosítása*: Miután a konténer betöltötte a servlet-et példányosítja azt. A konténer a javax.servlet.Servlet interfész

```
public void init(ServletConfig config)
```

metódusa segítségével inicializálja a servlet-et. A servlet konténertől függ, hogy mikor és hány példányt hoz létre a servlet-ből, az biztos, hogy a servlet-nek szóló kérés kiszolgálása előtt legalább egy példány létre lett hozva. A servlet inicializálásakor kapott ServletConfig paraméteren keresztül a servlet inicializációs paramétereit, valamit egy ServletContext objektumot lehet elérni.

2.Sikeres inicializálás után a servlet egy példánya képessé válik a *kliens kéréseinek a kiszolgálására*. Ha kérés érkezik egy adott servlet felé, akkor a servlet konténer meghívja annak

```
public void service (ServletRequest req, ServletResponse res)
```

metódusát. A `ServletRequest` objektum a beérkező igényekről tartalmaz információt, a `ServletResponse` objektum lényegében egy puffer, ahol a servlet válaszát helyezhetjük el.

3.Ha a servlet konténer úgy dönt, hogy már nincs szükség az adott servlet-re, akkor

```
public void destroy()
```

metódus hívással megszüntetheti a servlet példányt, amely kikerül a servlet konténer ellenőrzése alól és a garbage collector eltávolíthatja azt a memóriából.

### *2.1.3. JSP (JavaServer Pages)*

A JSP technológia segítségével dinamikus tartalmú weblapok készíthetők. Egy JSP lap a szabványos HTML kódon kívül speciális JSP elemeket is tartalmaz, melyek a beérkező kérésektől függően dinamikus tartalmát állítják elő. A statikus és a dinamikus tartalom jól elkülöníthető.

Ha a kliens felől egy kérés érkezik a kiszolgáló felé egy adott JSP lap elérésére, az igény kiszolgálása előtt a kiszolgáló a JSP lapot servlet-é alakítja. A JSP konténer feladata a JSP lap servlet-é alakítása, valamit a servlet lefordítása. Ezután következik a servlet inicializációja, majd a servlet a kérést megkapva előállítja a dinamikus tartalmat és ez továbbítódik a kliens felé.

A JSP lap servlet-é alakítására az első kérés alkalmával kerül sor, az átalakítás miatt az oldal betöltése több időt vesz igénybe. Miután a servlet elindult a további kérések a servlet-hez kerülnek.

### *2.1.4. EJB (Enterprise JavaBeans)*

Az EJB elosztott üzleti objektumok megvalósítására kifejlesztett kiszolgáló oldali komponensmodell. Olyan szerver oldali összetevő, ami az alkalmazás üzleti logikáját valósítja meg. Az EJB példányainak létrehozásáról és futtatásáról egy úgynevezett EJB konténer gondoskodik. A perzisztenciakezelést és a tranzakciókezelést az EJB konténer szolgáltatja.

Három típusa létezik:

- A Session Beanek, üzleti folyamatokat valósítják meg, melyek különböző műveleteket hajtanak végre. Két típusa van az állapotmentes (stateless) és az állapottal rendelkező (stateful). Az állapotmentes beanek (Stateless Session Beans) esetében nincs nyilvántartva, hogy a kliens melyik bean példánnyal kommunikál, nem őrzi meg az állapotát, akár minden új kérésre, új példányt hozhat létre a konténer. Az állapottal rendelkező beanek (Stateful Session Beans) megőrzik állapotukat, és a bean élettartama alatt mindvégig ugyanazzal a klienssel kommunikálnak.

- Az Entity Beanek, perzisztens üzleti objektumok, tárolásuk általában valamilyen relációs-adatbázisban történik. Tipikusan a relációs-adatbázis adatbázis egy táblájának, egy példánya pedig a relációs-adatbázis egy sorának feleltethető meg. A perzisztencia kezeléstől függően kétféle típusa van, konténer által kezelt perzisztencia, az EJB konténer gondoskodik az entitás bean állapotának adatbázisba való leképezéséről. A bean által vezérelt perzisztencia ebben az esetben a bean készítőjének kell megírni a betöltő, kiíró kódot.

- A Message-driven beanek, működésében hasonlít egy Session Beanhez, üzleti folyamatokat valósítanak meg asszinkron módon.

Perzisztencia egy magasabb szintű absztrakció a JDBC fölött. A perzisztencia réteg objektumok relációs adatbázisra való leképezésére való, hogy az objektumok kereshetőek, visszatölthetőek, frissíthetőek, és szerkeszthetőek legyenek. A Java Persistence API definiálja az egyszerű Java objektumok adatbázisra való leképezését.

## 2.2. JBoss

A JBoss az egyik legszélesebb körben használt Java2EE szabványt implementáló Java-ban írt alkalmazásszerver (Application Server). A JBoss alkalmazásszerver magába foglalja az EJB 3.0 támogatást Nyílt forráskódú, letölthető a http://www.jboss.org/ oldalról. A nyílt forráskódnak köszönhetően szabadon letölthető, használható, terjeszthető. Mivel 100%-ban Java-ban van megvalósítva, ezért képes együtt működni minden olyan operációs rendszerrel, amelyeken futtatható a JVM. Ami még jobbá teszi, hogy mikrokernel alapjába integrálja a Hibernate-t, az Apache Tomcat-et, az EJB 3.0-at és a JBoss Cache-t.

A JBoss telepítése igen egyszerű, a http://labs.jboss.com/jbossas/downloads/ oldalról letölthetjük a zip fájlba csomagolt bináris változatát. A letöltött zip állományt egy kiválasztott könyvtárba kicsomagoljuk, létrehozunk egy JBOSS_HOME nevű környezeti változót, úgy hogy a JBoss telepítési könyvtárára mutasson. Az így konfigurált JBoss alkalmazásszerver a %JBOSS_HOME%/bin alkönyvtárban található run script segítségével indítható.

## 2.3. PostgreSQL

A PostgreSQL egy nyílt forráskódú (open source) objektumrelációs adatbázis-kezelő rendszer (Object-Relational Database Management System – ORDBMS). A PostgreSQL szoftver kifejlesztése 1986-ban kezdődött a Kaliforniai Egyetemen Berkeley-ben kutatási projektként.

A nyílt forráskódnak köszönhetően szabadon használható, másolható, terjeszthető, tanulmányozható és módosítható. Mivel nincs szoftver-licenc költség a PostgreSQL-t használó cégek számára jelentős megtakarítást jelent.

Néhány a PostgreSQL szolgáltatásai közül: teljesíti az ANSI SQL szabvány kritériumait, triggerek, Unicode-támogatás, öröklődés, nézetek, stb.

A PostgreSQL legújabb verziója ingyenesen letölthető a http://www.postgresql.org/ oldalról. Grafikus telepítő varázslóval rendelkezik, ezért telepítése igen egyszerű. Letölthető hozzá kifejezetten PostgreSQL-hez fejlesztett

pgAdmin III nevű grafikus adatbázis adminisztrációs eszköz, amely szintén ingyenesen letölthető a http://www.pgadmin.org/ oldalról, és szabadon használható.

## *2.4. NetBeans IDE*

A NetBeans IDE (Integrated Development Environment) egy nyílt forráskódú integrált fejlesztői környezet, amely a Java nyelven alapul. A Sun Microsystems 2000 júniusában hozta létre a NetBeans nyílt forráskódú projektet, és jelenleg is a projekt főszponzora.

A NetBeans IDE olyan grafikus fejlesztői környezet, amely lehetővé teszi a programozók számára, hogy programokat írjanak, fordítsanak, teszteljenek, hibakeresést végezzenek az alkalmazásokban, programokat telepítsenek. Alkalmazásával könnyebben, gyorsabban tudjuk fejleszteni projektjeinket. Java nyelven íródott, de más programozási nyelvet is támogat. Számos modullal (plugin-el) bővíthető, ingyenes termék nincsenek korlátozások a használatára vonatkozóan.

A jelenleg elérhető nem béta verzió a NetBeans IDE 6.0.1, melyet a fejlesztés folyamán is használtam. A dolgozat készültekor a NetBeans IDE 6.1 még csak béta verzióban érhető el.

# 3. A rendszer tervezése

A teledermatológiai rendszer tervezése csoportosan történt, amiben nekem is lehetőségem volt részt venni. Ez volt az első olyan projektem, aminek a tervezése során lehetőségem volt megtapasztalni egy szoftverfejlesztéssel foglakozó cég munkáját. A fejlesztés megkezdése előtt a rendszerrel szemben támasztott követelmények és a rendszer funkcióinak meghatározása történt, melynek eredményeképpen létrejött két darab dokumentáció. Egy Szoftver Követelmény Specifikáció (Software Requirements Specifications - SRS), mely a „IEEE 830-1998 Recommended Practice for Software Requirements Specifications" szabvány szerint íródott, és egy Szoftver Specifikáció (Software Specifications).

## *3.1. Megbízhatóságról*

Az egészségügyi ellátásokkal szemben alapvető elvárás a megbízhatóság. A betegek egészségi állapotát az új ellátási forma sem veszélyeztetheti. Ezért a rendszert klinikailag tesztelni kell, a tesztek eredményeit pedig összevetni a hagyományos ellátási forma eredményeivel. Ennek vizsgálatára, fejlesztett ellátó rendszerünk prototípusa tulajdonképpen a rendszer egy klinikai tesztelésre alkalmassá tett speciális verziója. A kifejlesztendő rendszertől technikai hátterét tekintve nem tér el, azonban a vizsgált betegek ellátása mindig orvosi felügyelettel történik.

## *3.2. Klinikai vizsgálatokról*

***Primer betegvizsgálat***: Etikai szempontból lényeges, hogy a vizsgált betegek a vizsgálatok során a korábbi kezeléseket megkapják. A rendszer által nyújtott szolgáltatás számukra csak, mint plusz odafigyelés, gondosabb ápolás jelentkezik.

A beteg dokumentációit szöveges adatként tárolják, majd a sebekről standardizált módon közeli és távoli fényképeket készítenek. Ez után a jelenlévő

orvos megfelelően kezeli a sebet, és véleményt alkot a további teendőkről. Előnyös, ha a vizsgálati betegcsoportba olyan betegek kerülnek, akik régóta kezelés alatt állnak (esetleg korábban több orvos is vizsgálta már), ily módon a primer diagnózisukat tekintve a tévedés lehetősége elhanyagolható.

***Szekunder betegvizsgálat***: Ez után következik a rendszer biztonságosságának tesztelése: a szöveges adatokat és a fényképeket egy független orvoshoz juttatják el. Ő a beteget és az első vizsgáló szakvéleményét nem ismerheti. A kapott dokumentumok alapján megalkotja a saját szakvéleményét, terápiás javaslatot tesz, rendelkezik a további teendőkről.

***Kontroll vizsgálat***: A két orvos véleményének összevetését egy harmadik specialista végzi. A rendszerrel kapcsolatos gyakorlati tapasztalatokról tanulmány készül, melyet a fejlesztéshez hasznosíthatunk. A vizsgálat külső kontrollját a tervezett tudományos közlemény szakmai visszhangja adja.

## *3.3. Definíciók*

**Anamnézis**: kórtörténet és jelen panasz.

**Dermatológia:** bőrgyógyászat.

**Egészségügyi adat:** Az érintett személy testi, értelmi és lelki állapotára, kóros szenvedélyére, valamint a megbetegedés, illetve az elhalálozás körülményeire, a halál okára vonatkozó, az érintett által vagy róla más személy által közölt, illetve az egészségügyi ellátóhálózat által észlelt, vizsgált, mért, leképezett vagy származtatott adatok, vagy amelyek ezekkel kapcsolatba hozhatóak, azokat befolyásolja. Járványügyi érdekből a szexuális szokásokra vonatkozó információk is ide tartoznak.

**Egészségügyi dokumentáció:** az egészségügyi szolgáltatás során az egészségügyi dolgozó tudomására jutó, a beteg kezelésével kapcsolatos egészségügyi és személyazonosító adatokat tartalmazó feljegyzés, nyilvántartás vagy bármilyen más módon rögzített adat, függetlenül annak hordozójától vagy formájától.

**Egészségügyi ellátás:** a beteg adott egészségi állapotához kapcsolódó egészségügyi szolgáltatások összessége.

**Ellátó:** az a személy, aki a diagnózis megérkezése után a beteget ellátja. Lehetősége van a Beviteli egységet használni vagy bármely más eszközre megkapni a diagnózist.

**Epikrízis:** a kórtörténet rövid írásos összefoglalása, tartalmazza a terápiás javaslatot.

**Konzultáns:** az a személy, aki a Megjelenítő egység előtt ülve a diagnózist készíti és beviszi a rendszerbe. Az adott területhez értő specialista, szakorvos.

**Megjelenítő egység:** az a hardver, amely a vizsgálat eredményét megjeleníti. Szabványos, bármely orvosi szakterületen megtalálható (monitor).

**Státusz:** a fizikai vizsgálat eredménye.

**Személyazonosító adatok általában:** Név, Szül. hely, idő, Anyja neve, Lakcím, Leánykori név, Neme, TAJ szám, (ha van egyéb betegbiztosítása annak a száma).

**Teledermatológia:** távbőrgyógyászat.

**Telemedicina:** távgyógyászat. A távközléssel direkt módon támogatott egészségügyi gyógyító és diagnosztikai tevékenységet magába foglaló gyűjtő fogalom.

**Vizsgáló:** Az a személy, aki a vizsgálatot elvégzi és beviszi az eredményeket a Beviteli egységbe.

## *3.4. A rendszer funkcionalitása*

A kifejlesztett rendszer lehetőséget biztosít a krónikus sebekkel rendelkező betegek otthoni ápolására, mellyel jelentős kórházi és utazási költségek takaríthatók meg. Az otthon ápolt betegek krónikus sebeinek túlnyomó részét a vénás, vagy artériás keringészavar talaján kialakult lábszárfekély, a cukorbetegség (diabéteszes láb szindróma), traumás sérülés, esetleg elgennyedt műtéti sebzés adja.

A rendszer költséghatékonyságának egyik összetevője, hogy – mivel egységes rendszerről van szó - kiküszöböli a diagnosztikai és egyéb vizsgálatok duplikálásából keletkező kiadásokat, időt és munkaerőt takarít meg.

A kifejlesztésre kerülő rendszer a „*strore and forward*” módszer alapjaira épül. A „*store and forward*” módszer lényege, hogy az adott betegről dokumentáció és a sebeiről fénykép készül, amit egy távoli szakorvoshoz továbbítanak, aki megalkotja a diagnózist, és terápiás javaslatot tesz. Ebben az esetben a beteg sebeiről az információ grafikuskép-formátumban számítógép merevlemezére kerül, és az itt tárolt fájlok kerülnek továbbításra. Létezik egy másik lehetőség is, amikor a beteg sebeiről készülő bemeneti jelsor közvetlenül és folyamatosan az adatátviteli csatornára jut, amelyet a távoli szakorvos online mozgóképként érzékel. A „*store and forward*” módszerrel megvalósított rendszer kis adatátviteli sávszélességet igényel, egyszerűbb és olcsóbb.

## *3.5. A rendszer felhasználói*

A klinikai tesztfázis ideje alatt öt felhasználótípust különböztethetünk meg, helyi szakorvos, távoli szakorvos, specialista, adminisztrátor.

A helyi szakorvos a teszt fázis idején a beteg felvevő és a konzultáns szerepét is betölti, ő dönt arról, hogy melyik betegek kerülnek be a rendszerbe. Lehetősége van új beteget felvenni, képes a felvett betegek személyazonosító adatait megtekinteni, módosítani, kivéve a TAJ számot. Továbbá lehetősége van anamnézis, státusz, kép, diagnózis és epikrízis felvitelére, az adott beteg korábban elkészült orvosi

dokumentumainak megtekintésére. Képes hozzárendelni egy adott beteg orvovi dokumentációját egy távoli szakorvoshoz.

A távoli szakorvos megjelenítő egységnél végzi munkáját, az adott orvosi szakterülethez nagyon jól ért. A kapott adatok alapján diagnózist és terápiás javaslatot készíthet, és vihet fel a rendszerbe. A beteg személyes adatait nem láthatja, csak az eset azonosítására szolgáló kódot.

A specialista a klinikai teszt fázisban a helyi és a távoli szakorvos diagnózisait és epikríziseit összevetve alakítja ki véleményét. Az adott orvosi szakterülethez nagyon jól ért. A beteg személyazonosító adatait nem tekintheti meg. Nem tekintheti meg a korábban elkészült szakvéleményeit.

Az adminisztrátornak lehetősége van új felhasználó felvételére a rendszerbe, joga van a felvett felhasználók adatainak megtekintésére, módosítására. Egy adott felhasználó jelszavának megváltoztatására, felhasználó törlésére (logikai törlés), naplófájlok megtekintésére. Nincs lehetősége a felvett betegek személyazonosító adatainak, orvosi dokumentációinak létrehozására, megtekintésére vagy módosítására.

A rendszer felhasználóinak lehetőségük van megtekinteni személyes adataikat, de módosítani csak az adminisztrátornak van jogosultsága. A felhasználóknak lehetőségük van saját jelszavuk megváltoztatására.

## *3.6. Forgatókönyvek és használati-eset diagramok*

A használati-eset (use case) diagram a rendszer működését írja le a külső aktor szemszögéből, elemei az aktorok, a használati esetek, és az ezek közötti kapcsolatok. Az aktor üzenetet küld a rendszernek, a rendszer pedig a használati eset végrehajtása közben üzenetet küldhet vissza az aktornak. Forgatókönyv a használati eset egy konkrét végrehajtásának leírása.

A rendszerrel szemben támasztott követelmények pontosításának egyik módszere, ha egyes használati eseteket részletesen megpróbálunk leírni. A

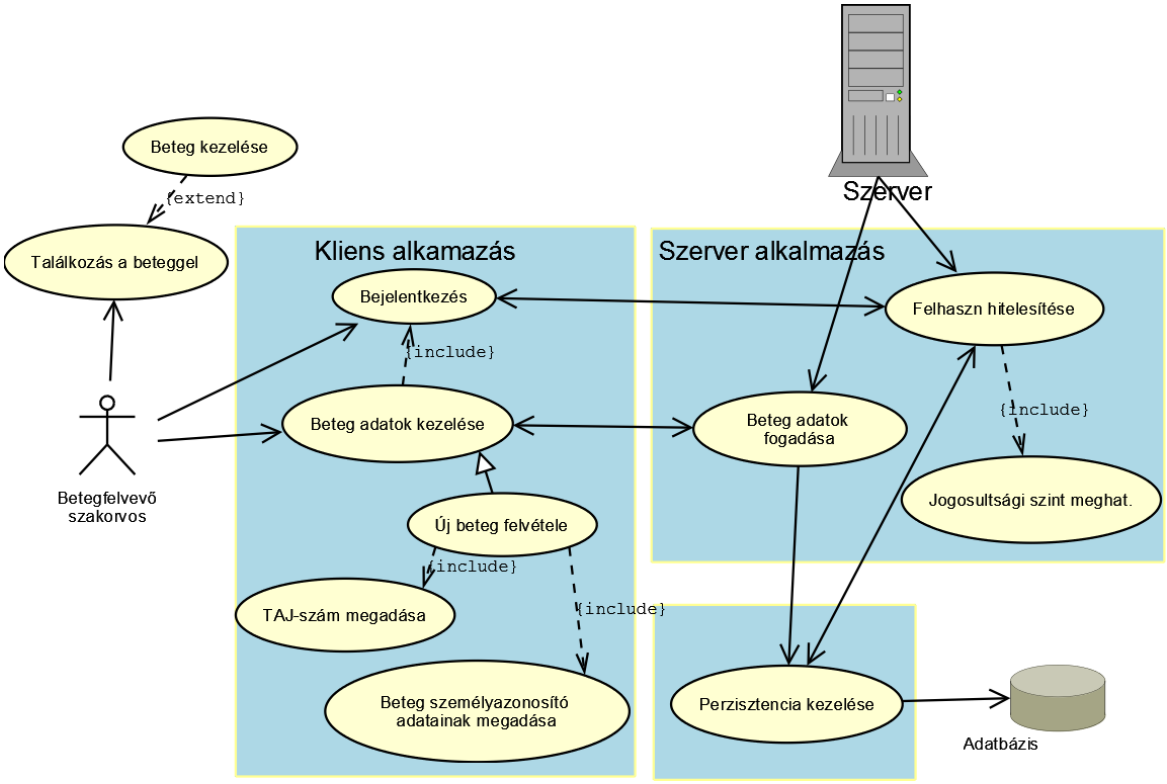
rendszerrel kapcsolatos használati esetek és forgatókönyvek közül mutatok be néhányat.

Az alábbi forgatókönyvek és használati-eset diagramok a betegfelvevő szakorvossal kapcsolatosak.

- **Bejelentkezés a rendszerbe**

  - A szakorvos elindítja az alkalmazást.
  - A bejelentkező képernyőn megadja felhasználónevét és jelszavát.
  - A Bejelentkezés gombot választja.
  - A rendszer ellenőrzi a megadott felhasználónév és jelszó párost.
    - A bejelentkezés sikeres volt, és a megadott felhasználónév megegyezik a jelszóval, akkor megjelenik a jelszóváltoztató panel, ahol a bejelentkezéshez meg kell változtatnia a jelszavát.
      - A jelszóváltoztató panelen megadja a régi és kétszer az új jelszavát.
      - Az OK gombot választja.
        - A jelszóváltoztatás sikeres volt, akkor sikeresen bejelentkezett a rendszerbe.
        - A jelszóváltoztatás sikertelen volt.
          - Újra próbálkozhat.
          - Vissza gombot választva visszaléphet a bejelentkező panelra.
    - A bejelentkezés sikertelen volt.
      - Újra próbálkozhat.
      - Kilépés gombot választva kiléphet a rendszerből.

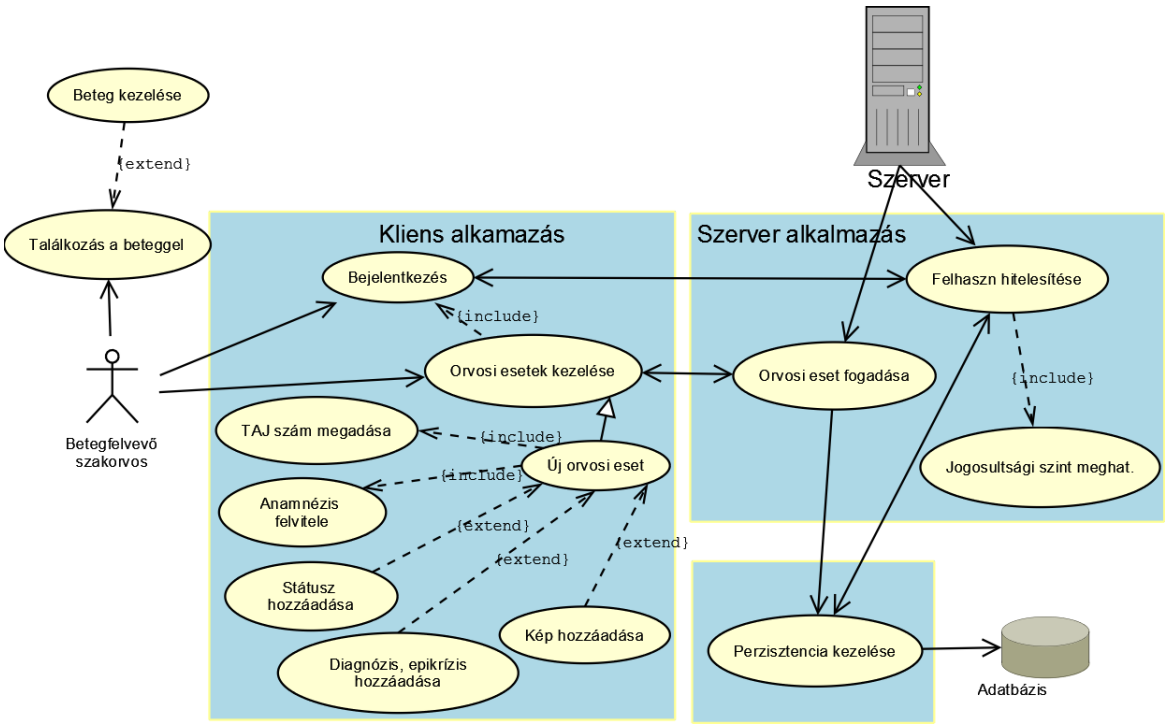
- **Új beteg felvétele:**




1. ábra: Új beteg felvétele use case diagram

- A szakorvos bejelentkezik a rendszerbe.
- Kiválasztja a Beteg adatok kezelése menüpontot.
  - Kiválasztja az új beteg felvétele menüpontot.
    - A szakorvos megadja a beteg TAJ számát.
    - Ha a TAJ szám helyes.
      - A szakorvos beviszi a beteg személyazonosító adatait.
      - Létrehozás gombot választva.
    - Ha a TAJ szám helytelen.
      - Újra próbálkozhat.
    - Kilépés az Új beteg felvitele funkcióból.

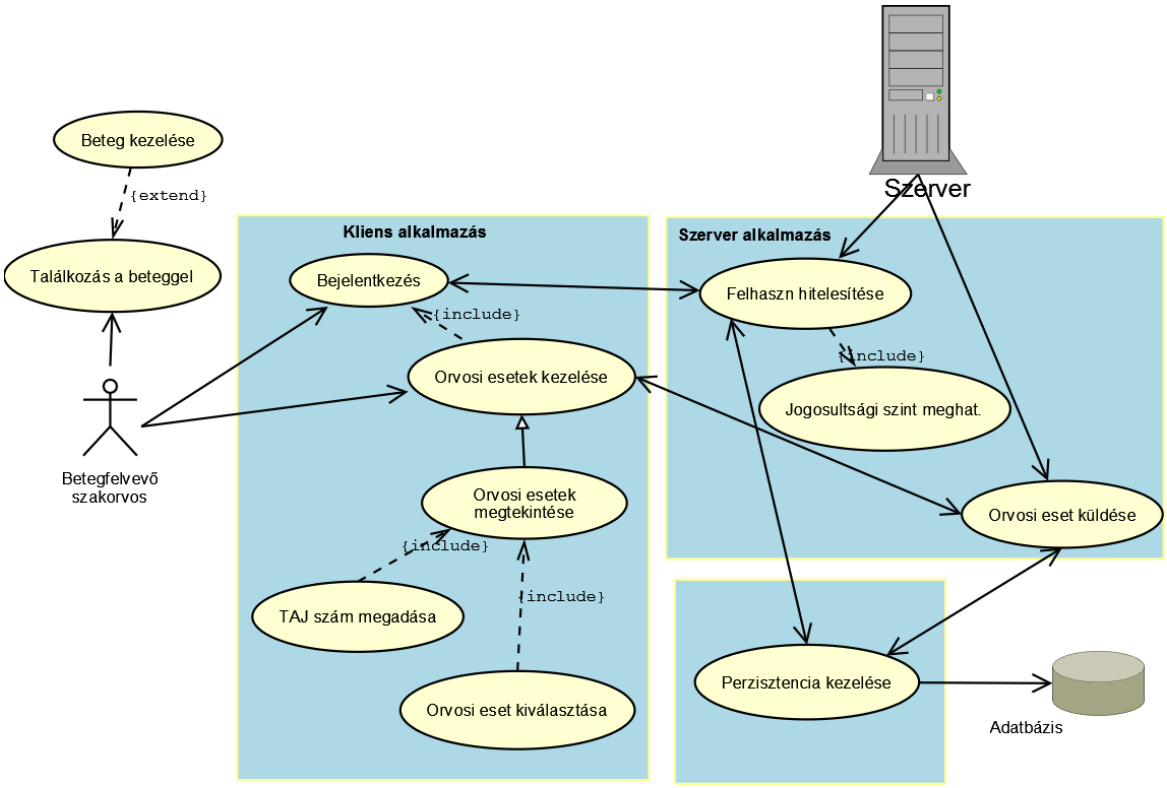
- **Új orvosi eset létrehozása:**




2. ábra: Új orvosi eset létrehozása

- A szakorvos bejelentkezik a rendszerbe.
- Kiválasztja az Orvosi esetek kezelése menüpontot.
  - Kiválasztja az Új orvosi eset menüpontot.
    - A szakorvos megadja a beteg TAJ számát.
      - Ha a TAJ szám helyes.
        - A felviszi az anamnézist.
        - Lehetősége van továbbá státusz, kép, diagnózis és epikrízis megadására.
        - Az Orvosi eset mentése gombot választja.
      - Ha a TAJ szám helytelen.
        - Újra próbálkozhat.
    - Kilépés az Új orvosi eset funkcióból.

- **Orvosi esetek megtekintése**




3. ábra: Orvosi eset megtekintése

– A szakorvos bejelentkezik a rendszerbe.

– Kiválasztja az Orvosi esetek kezelése menüpontot.

– Kiválasztja az Orvosi esetek megtekintése menüpontot.

– A szakorvos megadja a beteg TAJ számát.

– Ha a TAJ szám helyes.

– Az orvosi esetek közül egyet kiválasztva megtekintheti azt.

– Ha a TAJ szám helytelen.

– Újra próbálkozhat.

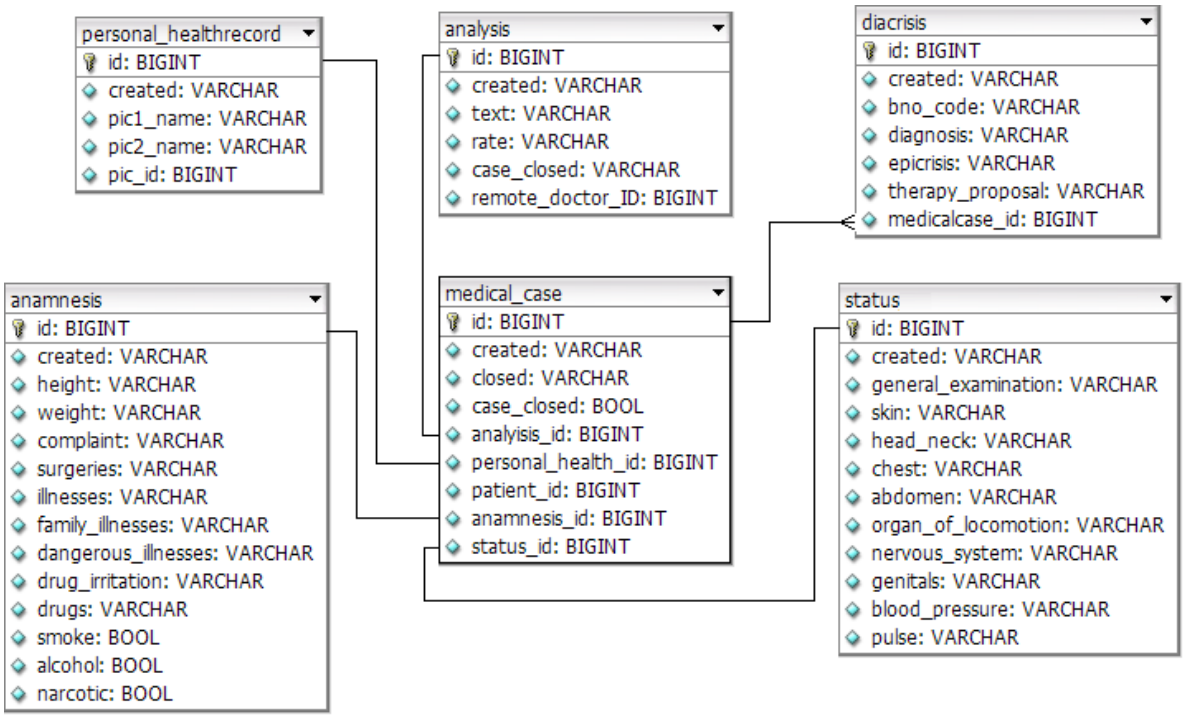
– Kilépés az Orvosi esetek megtekintése funkcióból.

### *3.7. Adatbázis megvalósítása*

Az adatbázis megtervezésénél már egy korábbi projekt adatbázis sémájára támaszkodva terveztük meg a rendszer adatbázis struktúráját. Tárolni kellett a felhasználók személyes adatait és a belépéshez szükséges felhasználónév-jelszó párost. Továbbá a felvett betegek személyazonosító és az orvosi adataikat. Az objektumok relációs adatbázisba történő reprezentálását az EJB technológia mentén létrehozott Entity Bean-ek segítségével oldottuk meg.

Az én feladatom az orvosi adatok perzisztenciakezelést megvalósító enterprise bean-nek létrehozása volt. A adatok perzisztens tárolásához Entity Bean-eket, a folyamatok reprezentálásához pedig Stateless Session Bean-eket alkalmaztam

Egy adott Entity Bean osztály egy táblát reprezentál a relációs adatbázis modellben, egy adott példánya pedig egy rekordot a táblában. Az objektum-relációs leképezést az EJB konténer oldja meg. Az orvosi adatokhoz kapcsolódó Entity Bean osztályok szerkezetét a függelékben található diagram szemlélteti, a leképezés után az adatbázisban létrejövő sémát pedig a 4. ábra.




4. ábra: Az orvosi adatokhoz kapcsolódó relációs adatséma

Az alábbi forráskód az anamnézist reprezentáló Entity Bean osztály egy részletét szemlélteti. Az @Entity annotáció jelzi az EJB konténer számára, hogy ezen osztály példányait relációs adatbázisba szeretnénk leképezni. A @Table és a @Column annotációkkal az adatbázis táblák és oszlopaik neveit adhatjuk meg. Az elsődleges kulcs megadása az @Id annotációval történik, amit jelen esetben a MedicalRecord osztálytól örököl az Anamnézis osztály. Az @Inheritance annotáció azt jelzi, hogy az adott osztálynak megfelelő tábla egy másik táblával kapcsolatban áll.

```
package com.telederm.ejb.medical;

import java.io.Serializable;
import javax.persistence.*;

@Entity
@Table(name="ANAMNESIS")
@Inheritance(strategy=InheritanceType.JOINED)
public class Anamnesis extends MedicalRecord implements Serializable {

   private Boolean smoke, alcohol, narcotic;
   private String height, weight, complaint, surgeries, illnesses, familyIllnesses, dangerousIllnesses,
drugIrritation, drugs, created;


   public Anamnesis() {
   }

   @Column(name="HEIGHT")
   public String getHeight() {
      return this.height;
   }

   public void setHeight(String height) {
      this.height = height;
   }

   @Column(name="WEIGHT")
```

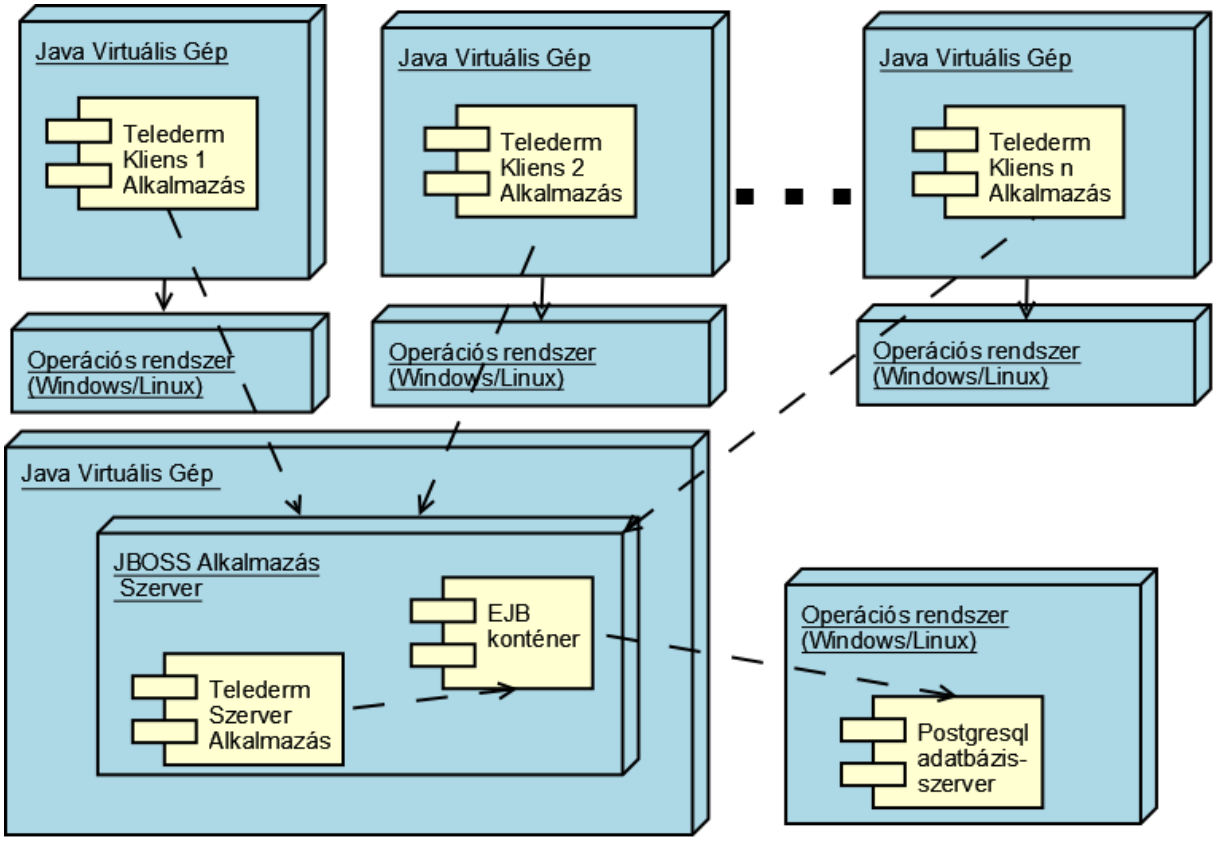
```
    public String getWeight() {
        return this.weight;
    }

    public void setWeight(String weight) {
        this.weight = weight;
    }
…
```

Az adatok lekérdezéséhez nevesített lekérdezéseket használtam, ezzel még jobban el tudtam választani adatbázis kezeléshez szükséges kódot, és az adott osztályra nézve adhattam meg a lekérdezéseimet. Az alábbi kódrészlet egy nevesített lekérdezést szemléltet.

```
@NamedQueries({
    @NamedQuery(name="getMedicalCaseOnTaj", query="from MedicalCase as mc where
mc.patient.taj = :taj")
})
```

A @NamedQueries annotációval tudjuk bevezetni a nevesített lekérdezéseinket, a @NamedQuery jelzi, hogy lekérdezés következik. A name adattag után megadott néven lehet majd elérni a lekérdezést, a query adattag után megadott lekérdezés hajtódik végre. Használhatunk úgynevezett EJB QL lekérdező nyelvet és natív SQL kódot is a lekérdezés megadásakor. A fenti példában EJB QL lekérdezést láthatunk, ebben az esetben kizárólag objektumokkal dolgozunk, ezzel adatbázis független adatbázis kezelést megvalósítva.

# 4. A rendszer felépítése




5. ábra: A rendszer technológiai diagramja

A távdiagnosztikából fakad a rendszer elosztott jellege. Implementációs nyelvként a Java-át választottuk, mivel platformfüggetlen és alkalmas elosztott rendszerek létrehozására. A teljes rendszer kialakítását a Java technológiára alapozva oldottuk meg. Szerverként az igen közkedvelt és elterjedt Java-ban írt JBoss alkalmazásszervert választottuk. Adatbázis kezelőként PostgeSQL 8.2 adatbázis szerverre eset a választás, mivel ingyenes és szabadon felhasználható.

A funkciók, igények, és követelmények meghatározása után következett a rendszer implementálása. A rendszer adminisztrátori részét webes felületen akartuk megoldani, a szakorvosok által használatos kliens részt pedig Java alkalmazás

formájában kívántuk megvalósítani, így a fejlesztés további menete két jól elkülöníthető részre bomlott.

## *4.1. Adminisztrátori felület:*

A webes felülettel ellátott adminisztrációs rész megvalósítása egyik munkatársam feladata volt. A fejlesztéshez JSP és Servlet technológiákat használt fel. Az ő feladata volt a felhasználók autentikációjának megvalósítása is.


6. Ábra: adminisztrátori beléptető ablak

## *4.2. Szakorvosok által használt kliens*

Én a szakorvosok által használt kliens rész megvalósítását kaptam feladatul. A képkezelésnél a képek adatbázisban való tárolásának magvalósítása és a Segítség menüpont tartalmi kialakítása nem tartozott közvetlenül a feladataim közzé. A cél egy olyan alkalmazás kifejlesztése volt amely átlátható, könnyen és gyorsan kezelhető.

Elsőnek a grafikus felületet terveztem meg, azután a rendszerbe való bejelentkezést kellett megoldanom, majd végül a grafikus felületek összehangolt működését.

Az alkalmazás grafikus felület kialakításához a Java Swing grafikus komponens gyűjteményt használtam. Mivel a Swing tisztán Java-ban van megvalósítva, ezért az alkalmazás futtatható minden olyan számítógépen amelyikre JVM telepíthető. A fejlesztés folyamán NetBeans IDE-t használtam, amelyben egyszerűen és gyorsan lehet létrehozni, módosítani grafikus felületeket. Egyszerűen drag and drop technikával lehet a Swing komponenseket ráhelyezni egy adott felületre.

Kerettel és fejléccel rendelkező ablakokat a `javax.swing.JFrame` osztály reprezentál. Az alkalmazás főablakát megvalósító komponens a `JFrame` osztály leszármazottja. Az alkalmazás a jobb helykihasználás érdekében keret és fejléc nélkül teljes kényernyős módban nyílik meg és fut.

Az információk, figyelmeztetések, hibaüzenetek, és beviteli dialógus ablakok a `javax.swing.JDialog` osztály leszármazottjai valósítják meg.

Az adatbevitelhez és adatmegtekintéshez ugynevezett belső ablakokat használtam melyek a `javax.swing.JInternalFrame` leszármazottjai reprezentálják.

Az egészségügyi dokumentáció létrehozásához szükséges adatok megadásához a beviteli mezőket struktúrátlan, és áttekinthetően kellett megjeleníteni. Az egészségügyi dokumentáció megjelenítéséhez a többoldalas panel komponenst választottam melyet a `javax.swing.JTabbedPane` osztály reprezentál. A panelek fülein a dokumentáció összetevőinek a nevei láthatók, melyekre kattintva aktivá válik az adott lap és láthatóvá a tartalma. Előfordulhat, hogy az anamnézis és a státusz adatok két orvosi egészségügyi ellátás között csak néhány adatban térnek el. Ezeknek az adatoknak az újbóli felvitele lassíthatja az ellátást. Ezen információk gyors kitöltését hivatott megvalósítani egy tetszőlegesen választott korábbi egészségügyi dokumentáció adataival az Anamnézis kitöltése és a Státusz kitöltése funkció.

Elkészítettem az anamnézis, státusz, kép, diagnózis és epikrízis felvitelére szolgáló paneleket, majd ezek kombinált megjelenítését oldottam meg az adott felhasználó jogosultságának megfelelően.

Az 5. fejezetben bemutatom a kliens alkalmazást, ahol látható lesz a grafikus felület felépítése.

## *4.3. Biztonságról*

A webes felülettel rendelkező adminisztrátori rész a HTTPS (Hypertext Transfer Protocol Secure) protokollt használja az adatok biztonságos továbbítására. A HTTPS protokoll a HTTP (Hypertext Transfer Protocol) protokoll biztonságos, titkosított SSL

csatornán kommunikáló változata. A kiszolgáló a kliens kérésere elküld egy digitális tanúsítványt, amely tartalmazza a szerver nyilvános kulcsát. A kliens ellenőrzi a tanúsítvány érvényességét, hitelességét, és hogy a szerver domain neve megegyezik-e a tanúsítványban küldött adatokkal. Ha minden megfelelő, akkor létrejöhet a kapcsolat. Az adatok titkosított formában kerülnek továbbításra, ezzel ellehetetlenítve a lehallgatott információk értelmezését. A HTTPS a 433-as portot használja a kommunikációra.

Az adminisztrátori felületen és az egyéb felhasználók által használt kliens alkalmazás esetén is a rendszerbe való belépéshez meg kell adni egy felhasználónév jelszó párost. A jelszavak az MD5 egyirányú kódolási algoritmus segítségével képzet hash formában vannak letárolva az adatbázisban. A bejelentkezéskor a jelszavak szintén ilyen formában kerülnek elküldésre, és összehasonlításra. Ha az autentikáció sikeres volt, a felhasználó a jogkörének megfelelően megjelenő menüpontokon keresztül érheti el a rendszer funkcióit.

# 5. A Telederm kliens oldali alkalmazásának bemutatása

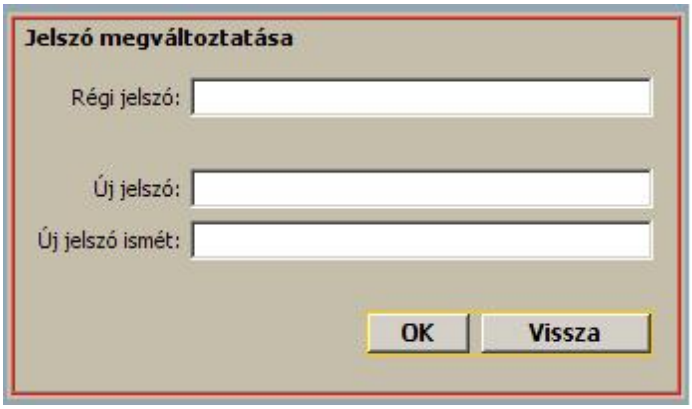
A következő fejezetekben az általam fejlesztett kliens alkalmazást mutatom be képernyőképekkel illusztrálva, a rendszer terjedelmére való tekintettel a teljesség igénye nélkül.

## *5.1. Bejelentkezés*

Az alkalmazás indítása után a bejelentkezési felület (7. ábra) fogadja a felhasználót. A belépéshez meg kell adni egy felhasználónév jelszó párost. A felhasználó legelső belépése esetén a jelszó a felhasználónévvel azonos. A Bejelentkezés gombra kattintva elkezdődik a felhasználó azonosítása. Sikeres azonosítás esetén a kezdőfelület jelenik meg az adott felhasználó jogosultságának megfelelően elérhető menüpontokkal.


7. ábra: Bejelentkezési felület

A legelső sikeres bejelentkezés esetén, a jelszó módosító ablak (8. ábra) jelenik meg, ahol a felhasználónak kötelezően meg kell változtatnia jelszavát. Az új jelszó nem lehet azonos a régivel. Sikeres jelszóváltoztatás esetén jelenik meg a kezdőfelület az elérhető menüpontokkal.




8. ábra: Jelszómódosító ablak

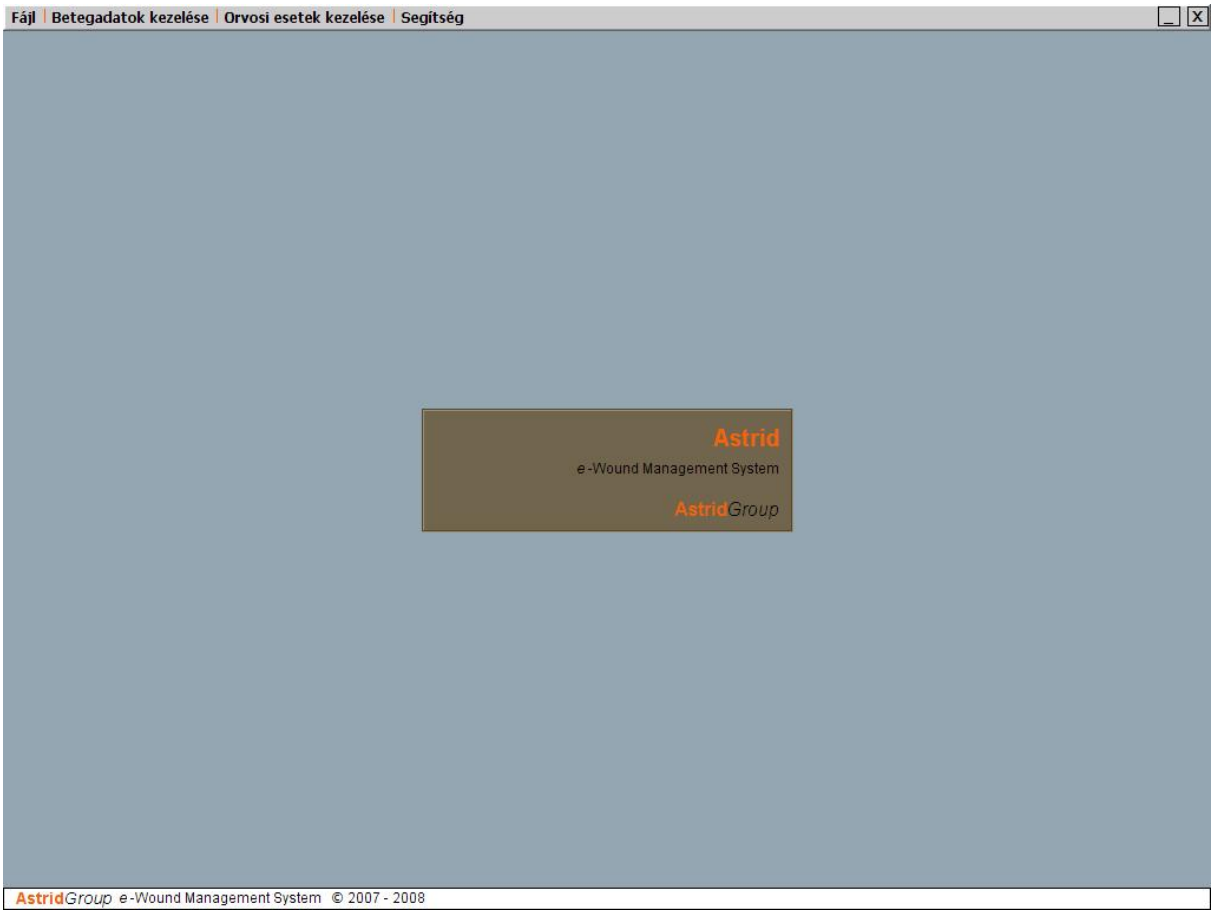
Sikertelen bejelentkezés esetén a hiba okától függően, a hiba üzenet dialógus ablakban, vagy közvetlenül a beviteli mezők alatt jelenik meg.

A bejelentkezési felületen a Kilépés gombra kattintva lehetőség van az alkalmazás bezárására.

## *5.2. Kezdőfelület*

A kezdőfelület az elérhető menüsorral a sikeres belépési autentikáció után válik láthatóvá. A menüsor menüpontjai a bejelentkezett felhasználó jogkörének megfelelően határozódnak meg.

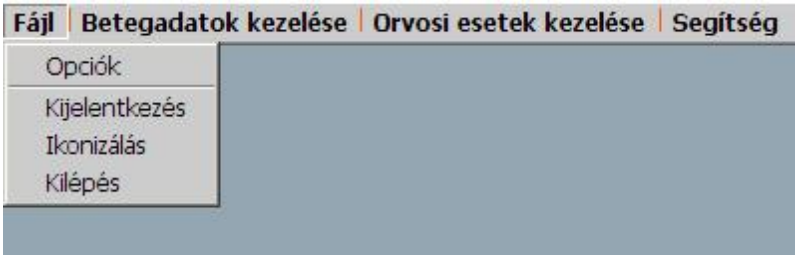
A 9. ábra egy helyi szakorvos jogosultsággal belépő felhasználó kezdőfelületét szemlélteti.




9. ábra: Kezdőfelület (helyi szakorvos esetén)

A menüsor a Fájl, Betegadatok kezelése, Orvosi esetek kezelése, Segítség menüpontokat tartalmazhatja.

## *5.3. Fájl menü*

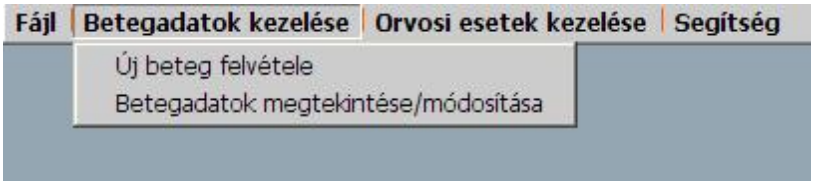



10. ábra: Fájl menü

A Fájl menü mindegyik felhasználótípus számára elérhető. Az Opciók menüpont alatt lehetőség van az aktuális felhasználó jelszavának megváltoztatására, személyes adatinak megtekintésére, az alkalmazás nyelvének megváltoztatásra.

Továbbá lehetőség van a rendszerből való kijelentkezésre a Kijelentkezés menüpont használatával, hatására törlődik a menüsor, és megjelenik a bejelentkező ablak. Ha ideiglenesen nincs szükségünk az alkalmazásra, akkor „leikonizálhatjuk” az Ikonizálás menüpont segítségével. A Kilépés menüpont segítségével kiléphetünk és bezárhatjuk az alkalmazást.

## *5.4. Betegadatok kezelése menü*



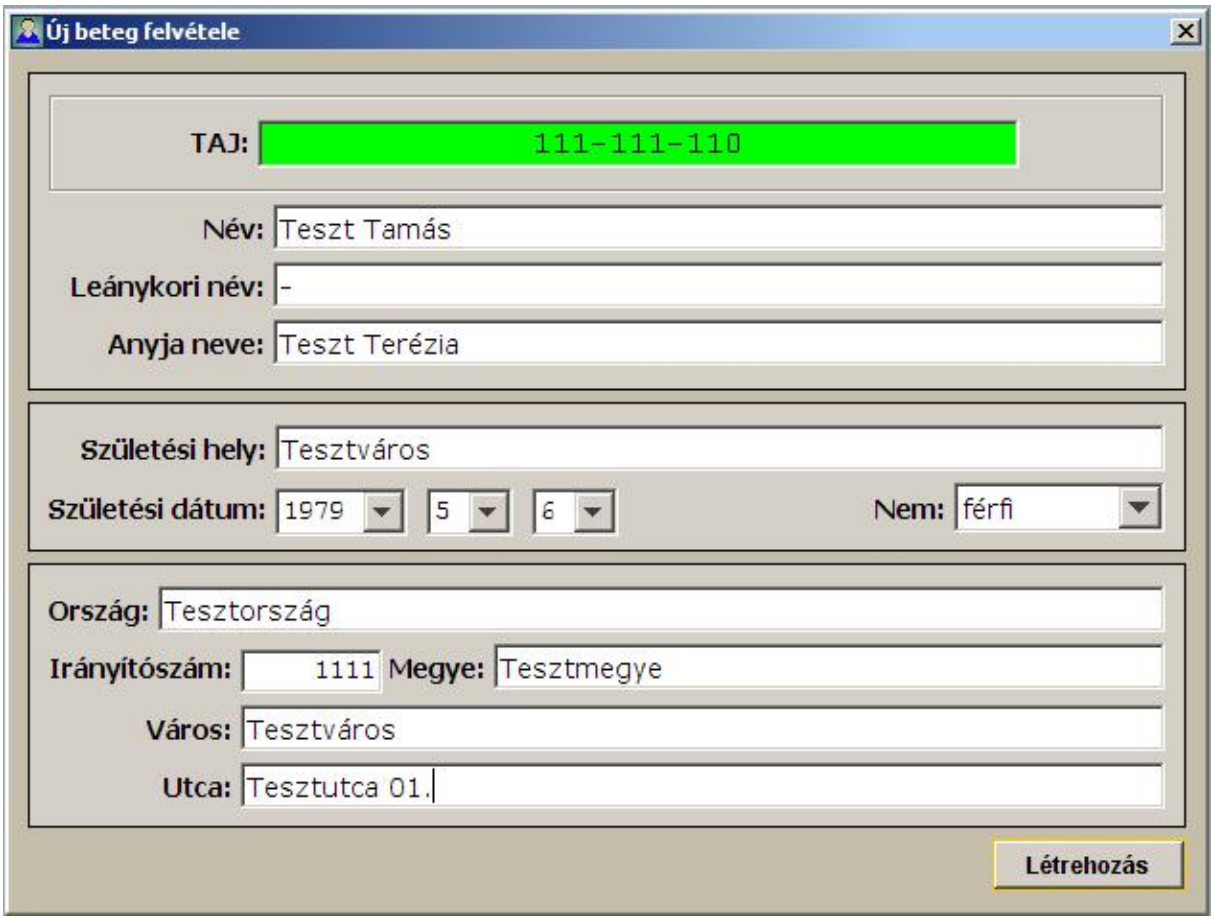

11. ábra: Betegadatok kezelése menü

Lehetősége van új páciens felvételére (Új beteg felvétele), és a rendszerbe felvett betegek adatainak megtekintésére, módosítására (Betegadatok megtekintése/módosítása).

### *5.4.1. Új beteg felvétele*

Lehetőség van új beteg felvételére a rendszerbe, ami kizárólag a helyi szakorvos jogosultsággal rendelkező felhasználók számára elérhető. A beteg felvétele a TAJ szám megadásával kezdődik. A TAJ szám elhagyása esetén, helytelen, vagy 9 számjegynél rövidebb TAJ szám megadása esetén, vagy ha a megadott TAJ szám már létezik a rendszerben, akkor hibaüzenet jelenik meg.

Helyes TAJ szám és a beteg személyazonosító adatainak megadása után a Létrehozás gombra kattintva lehetséges a páciens létrehozása. A művelet végeztével az ablak automatikusan bezáródik.

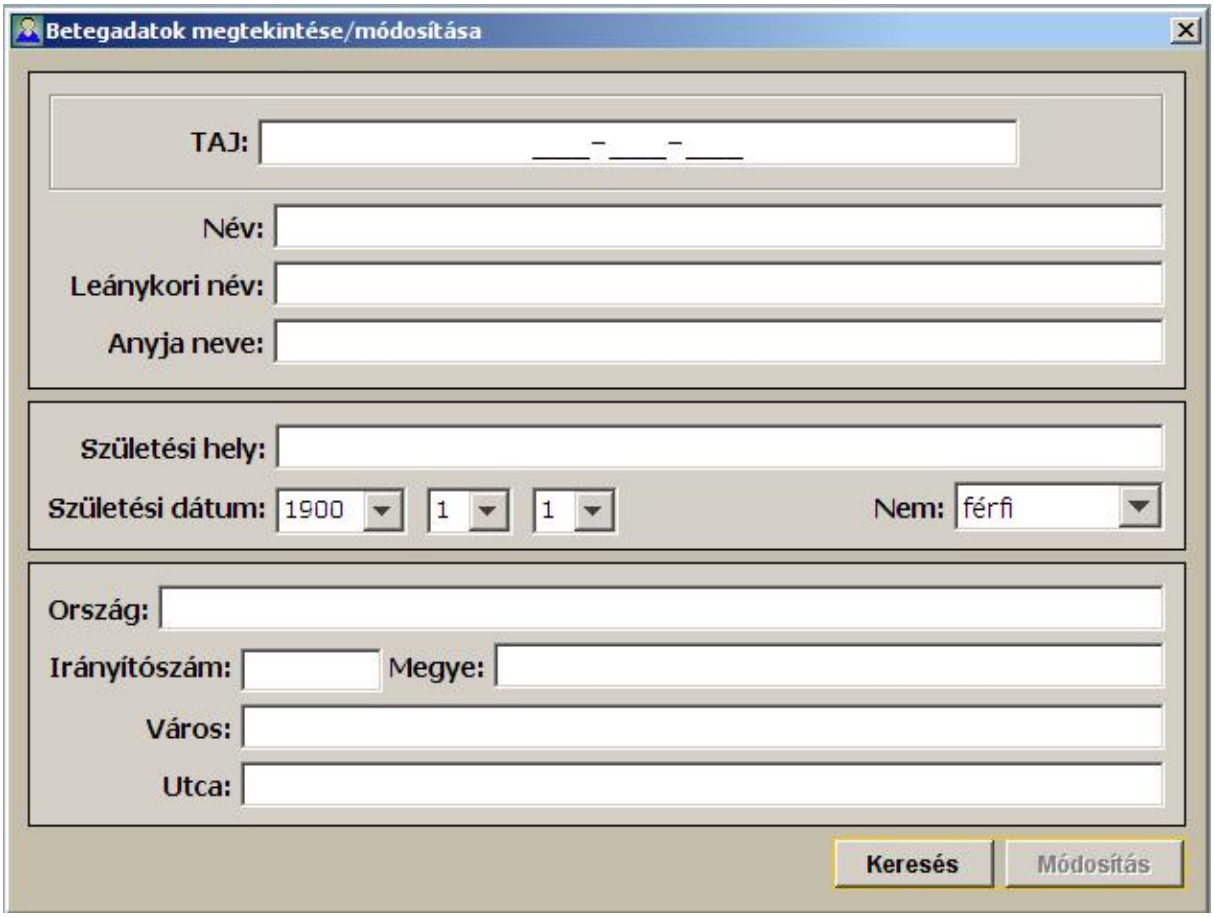



12. ábra: Új beteg felvétele

### *5.4.2. Betegadatok megtekintése/módosítása*

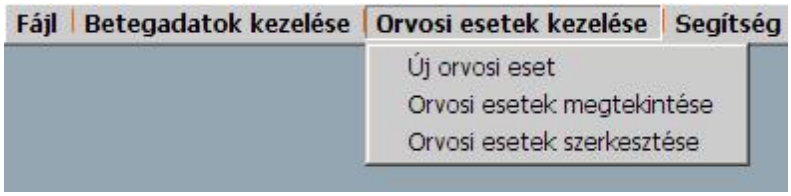
Lehetőség van a rendszerbe felvett betegek személyazonosító adatainak megtekintésére, módosítására, kizárólag a helyi szakorvos jogosultsággal rendelkező felhasználók számára elérhető. A keresés TAJ szám alapján történik. A TAJ szám elhagyása esetén, helytelen, vagy 9 számjegynél rövidebb TAJ szám megadása esetén, vagy ha a megadott TAJ nem létezik a rendszerben, akkor hibaüzenet jelenik meg.

Helyes TAJ szám megadása után a Keresés gombra kattintva a beviteli mezők feltöltődnek a TAJ számhoz tartozó beteg adataival, és a Módosítás gomb aktívvá változik. Az adatok megváltoztatása után a Módosítás gombra kattintva lehetséges a módosítások végrehajtása. A művelet végeztével az ablak automatikusan bezáródik.




13. ábra: Beteg adatok megtekintése, módosítása

## 5.5. Orvosi esetek kezelése menü




14. ábra: Orvosi esetek kezelése menü

Mindegyik felhasználótípus számára elérhető, de az almenüpontok az aktuális felhasználó jogosultságának megfelelően változnak.

Az Új orvosi eset funkció csak a helyi szakorvos, Orvosi esetek megtekintése funkciót a helyi szakorvos, és a távoli szakorvos, Orvosi esetek szerkesztése funkciót pedig a helyi, a távoli szakorvos és a specialista számára elérhető.

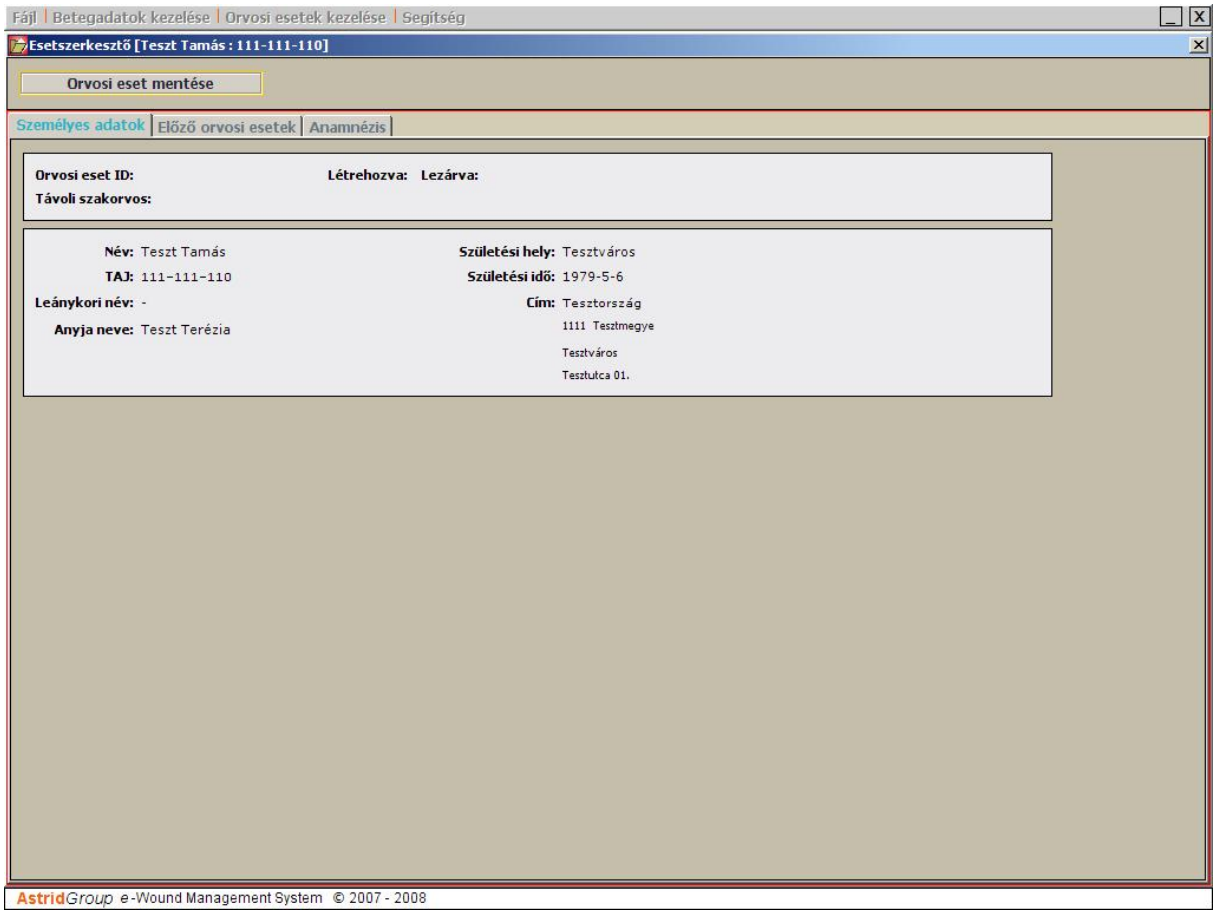
A 14. ábra a helyi szakorvos által elérhető Orvosi esetek kezelése menü almenüpontjait szemlélteti.

### 5.5.1. Új orvosi eset létrehozása

Új orvosi esetet egy már korában a rendszerbe felvett beteghez van lehetőség létrehozni. Az Új orvosi eset menüpontot választva a megjelenő dialógus ablakban az adott beteg TAJ számának helyes megadása után az Új eset létrehozása gombot választva megnyílik az esetszerkesztő ablak.


15. ábra: TAJ szám beviteli dialógus ablak

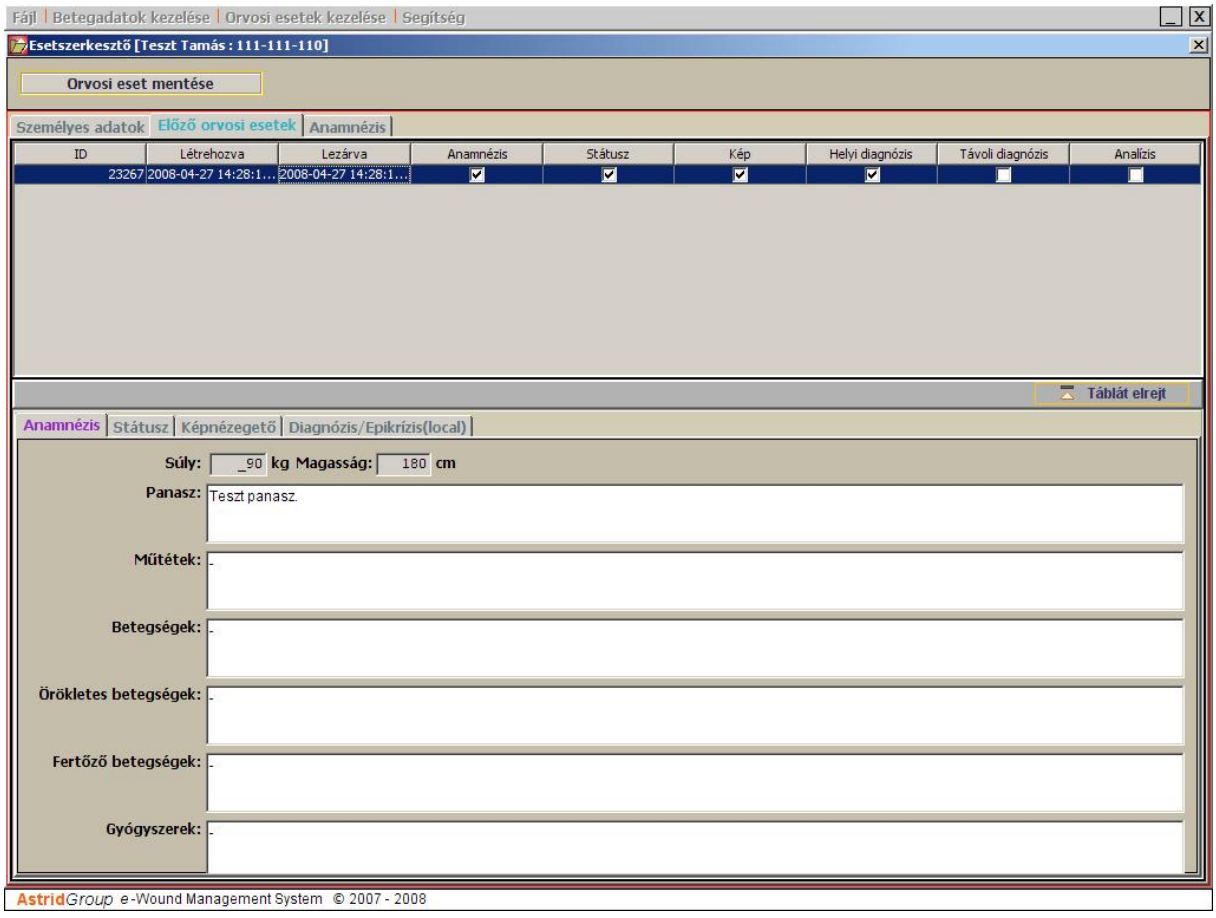
A Személyes adatok fülön az orvosi eset azonosító, a létrehozás-, lezárás dátuma, az orvosi esethez rendelt távoli szakorvos azonosítója, és az adott beteg személyazonosító adatai láthatók. A személyazonosító adatok módosítására itt nincs lehetőség.




16. ábra: Beteg személyazonosító adatai

Előző orvosi esetek fülön az adott beteghez tartozó korábban létrehozott orvosi eseteket lehet megtekinteni (17. ábra). Az orvosi esetek táblázatba rendezve jelennek meg, melynek egy sora egy az adott beteghez létrehozott orvosi esetet reprezentál. Tartalmazza az orvosi eset azonosítóját, a létrehozás és a lezárás dátumát, és tájékoztatást az orvosi dokumentumok állapotáról. A táblázat egy sorára kattintva az alsó részben megtekinthető a kiválasztott orvosi esethez tartozó orvosi dokumentum(ok).

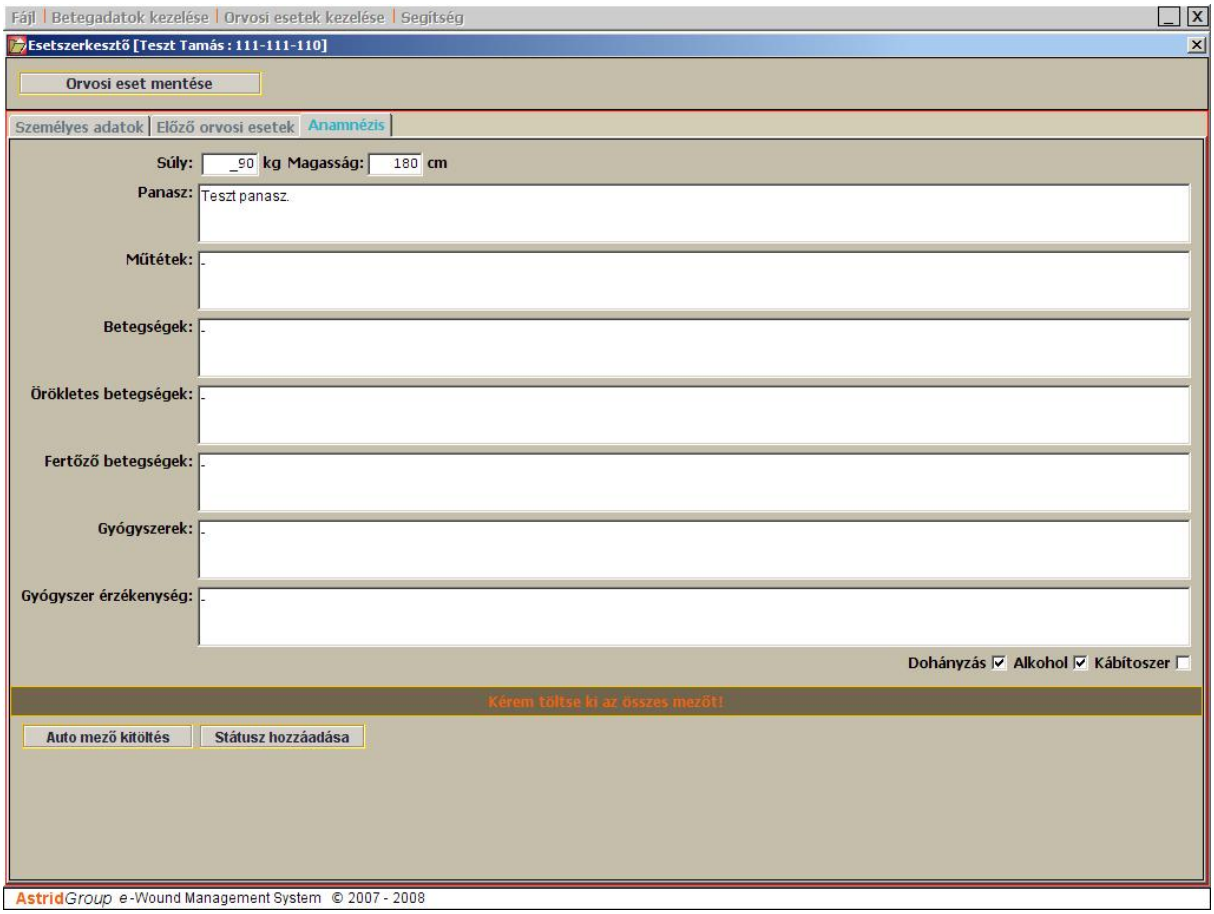
A táblázatot és a beteg orvosi dokumentációját kettéválasztó elem jobb szélén található Táblát elrejt funkciót választva az orvosi eseteket tartalmazó táblázat összecsukható, így a megtekinteni kívánt orvosi dokumentum(ok) alsó része is láthatóvá válik. A táblázat ismételt megjelenítéséhez a Táblát mutat funkciót kell választani.




17. ábra: Előző orvosi esetek megtekintése

Az Anamnézis fülön lehetőség van az adott orvosi esethez anamnézis felvitelére (18. ábra). Auto mező kitöltés gombra kattintással lehetőség van az anamnézishez tartozó mezők automatikus kitöltésére, az adott beteghez tartozó Előző orvosi esetek fülön kiválasztott orvosi eset anamnézis adataival. A Státusz hozzáadása gombra kattintva lehetőség van státusz adatok felvitelére.

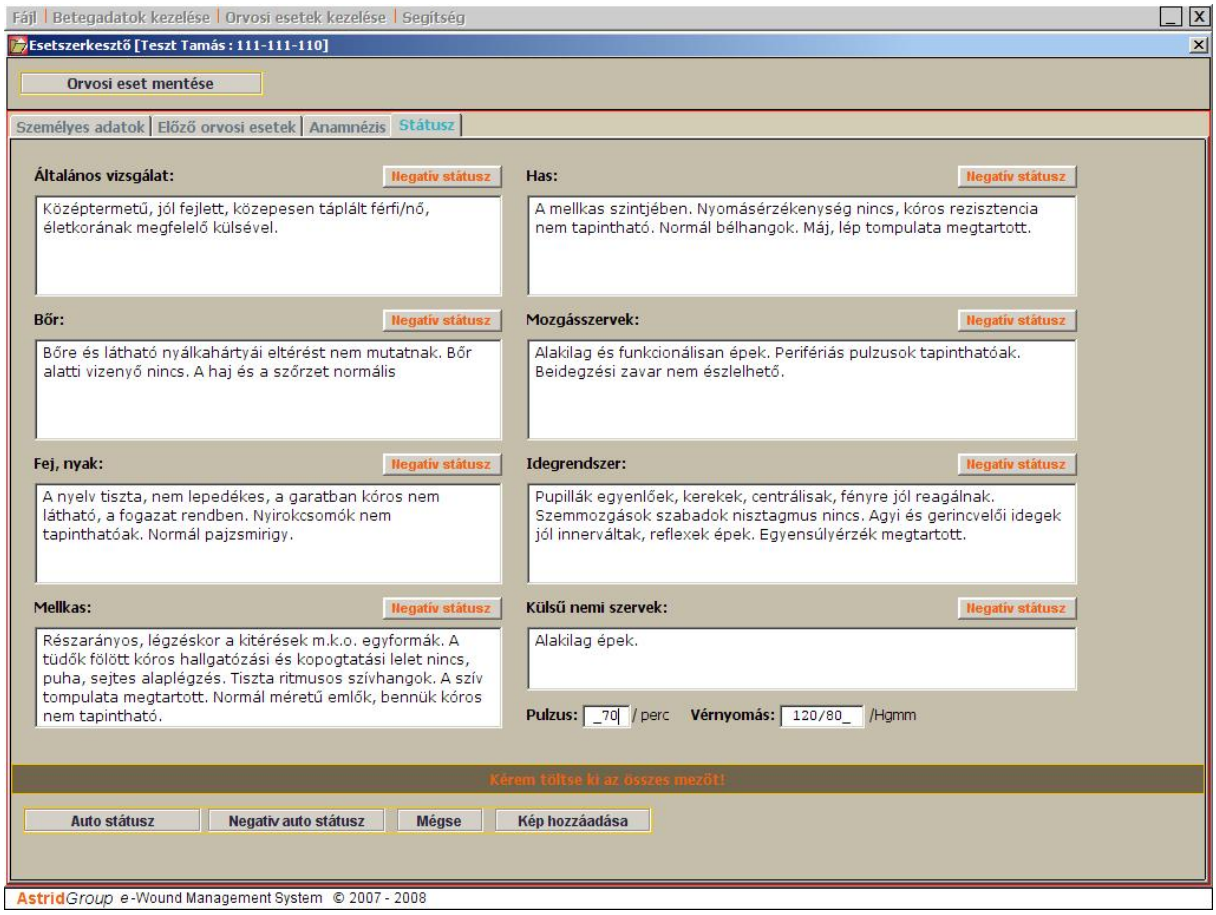
Az Anamnézis fülön mindegyik mező kitöltése kötelező, csak a jelölőnégyzetek opcionálisak.




18. ábra: Anamnézis fül

A Státusz fülön van lehetősé státusz adatok felvitelére (19. ábra). Az Auto státusz gombra kattintva a mezők az Előző orvosi esetek fülön kiválasztott orvosi eset státusz adataival töltődnek fel. A Negatív auto státusz gombra kattintva lehetőség van az összes mező kitöltésére úgynevezett negatív státusszal. Lehetőség van továbbá csak bizonyos mezők kitöltésére negatív státusszal, a mezők felett elhelyezett Negatív státusz gomb használatával.

Továbbá visszavonható a státusz a Mégse gombra kattintva, a Kép hozzáadása gombbal pedig lehetőség van képet csatolni a készülő orvosi esethez.



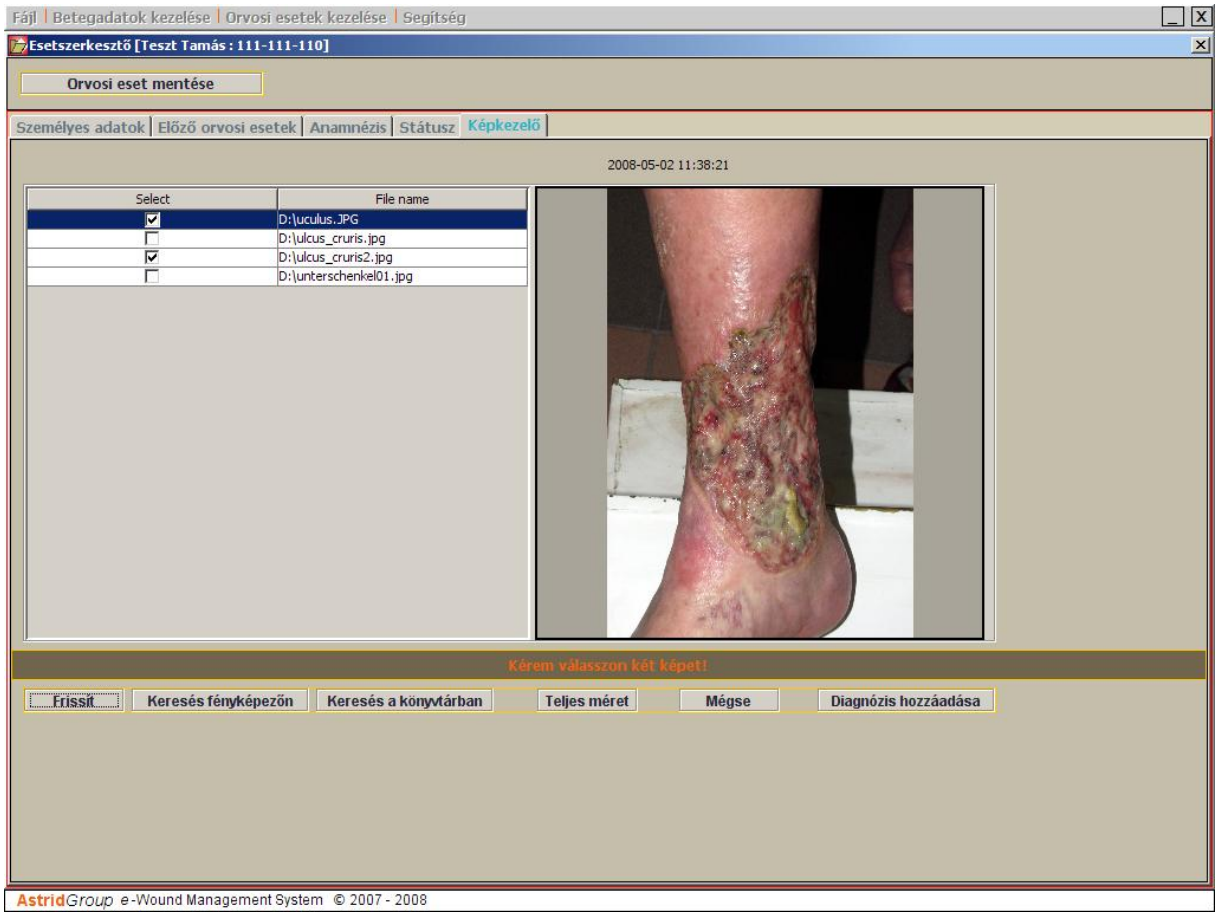

19. ábra: Státusz fül

A Képkezelő fülön (20. ábra) lehetőség van kép csatolására az orvosi eset dokumentációjához. Amennyiben van csatlakoztatva speciálisan formázott digitális fényképezőgép a munkaállomáshoz, akkor a Keresés fényképezőn gomb megnyomásával a fényképezőn található képék betöltődnek a táblázatba.

Keresés a könyvtárba gomb használatával pedig a számítógépen található tetszőleges könyvtár képeinek a betöltése lehetséges.

A táblázat egy sorára kattintva a jobb oldalt található előnézet ablakban jelenik meg a kiválasztott kép. Azok a képek kerülnek csatolásra az orvosi esethez, amelyeknél a táblázat első oszlopában található jelölőnégyzet ki lett jelölve. Maximum kettő darab kép kijelölése lehetséges, és szükséges az orvosi eset mentéséhez.

Visszavonható a kép hozzáadás művelet a Mégse gombra kattintva.



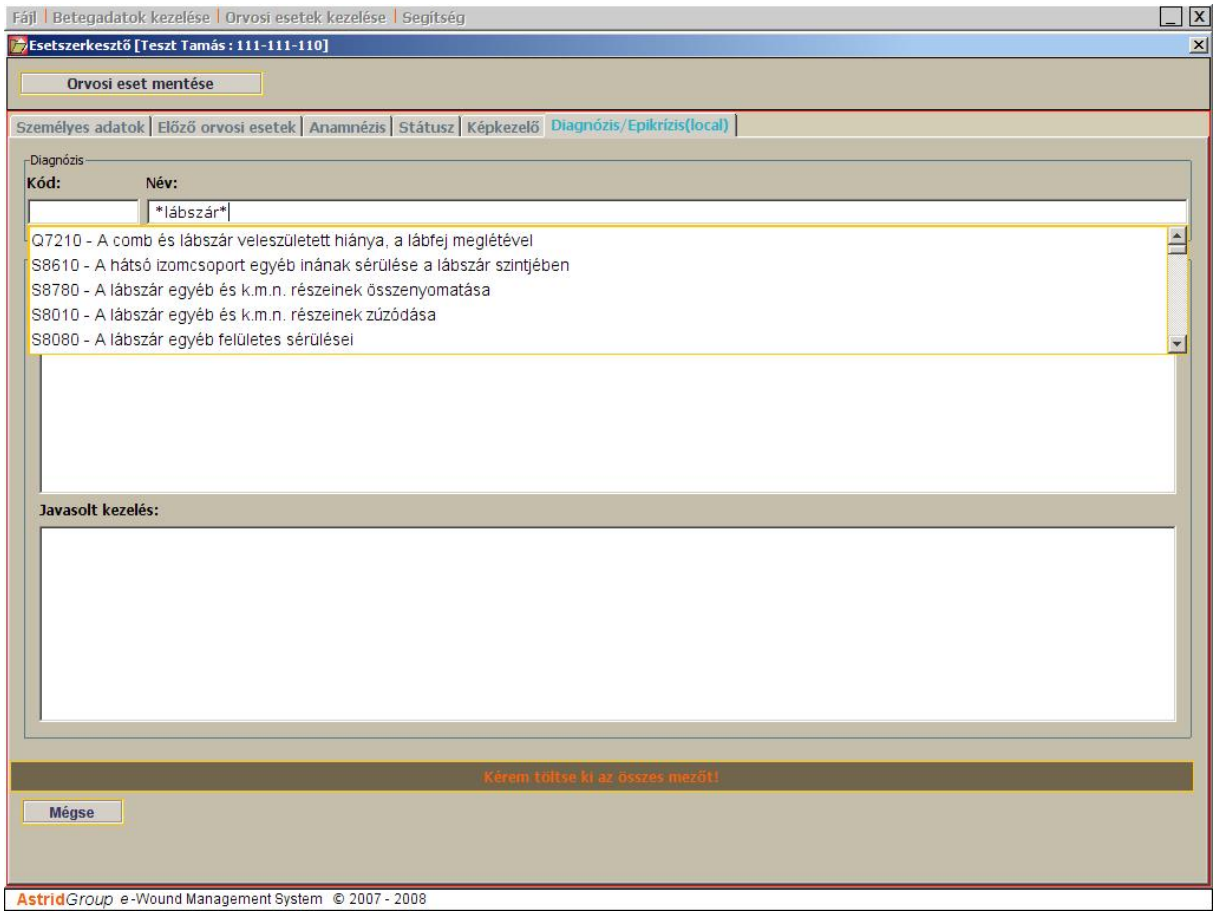

20. ábra. Képkezelő fül

Lehetőség van továbbá: a táblázat adatainak frissítésére, a Frissít gombra kattintva a kiválasztott kép megtekintésére eredeti méretben, az előnézeti képre, vagy a Teljes méret gombra kattintva, diagnózis csatolására a Diagnózis hozzáadása gomb segítségével.

Diagnózis/Epikrízis fülön (21. ábra) lehetőség van a diagnózis, epikrízis csatolásra az orvosi eset dokumentációjához.

A diagnózis megadható BNO kód alapján, vagy név alapján. Lehetőség van kód vagy név részlet alapján történő diagnózis keresésre. A lenyíló diagnózis listából választható ki a megfelelő diagnózis. Az alsó szövegmezőkbe vihető be az epikrízis (szakvélemény és a javasolt kezelés).

Lehetőség van a diagnózis visszavonására a Mégse gomb használatával.



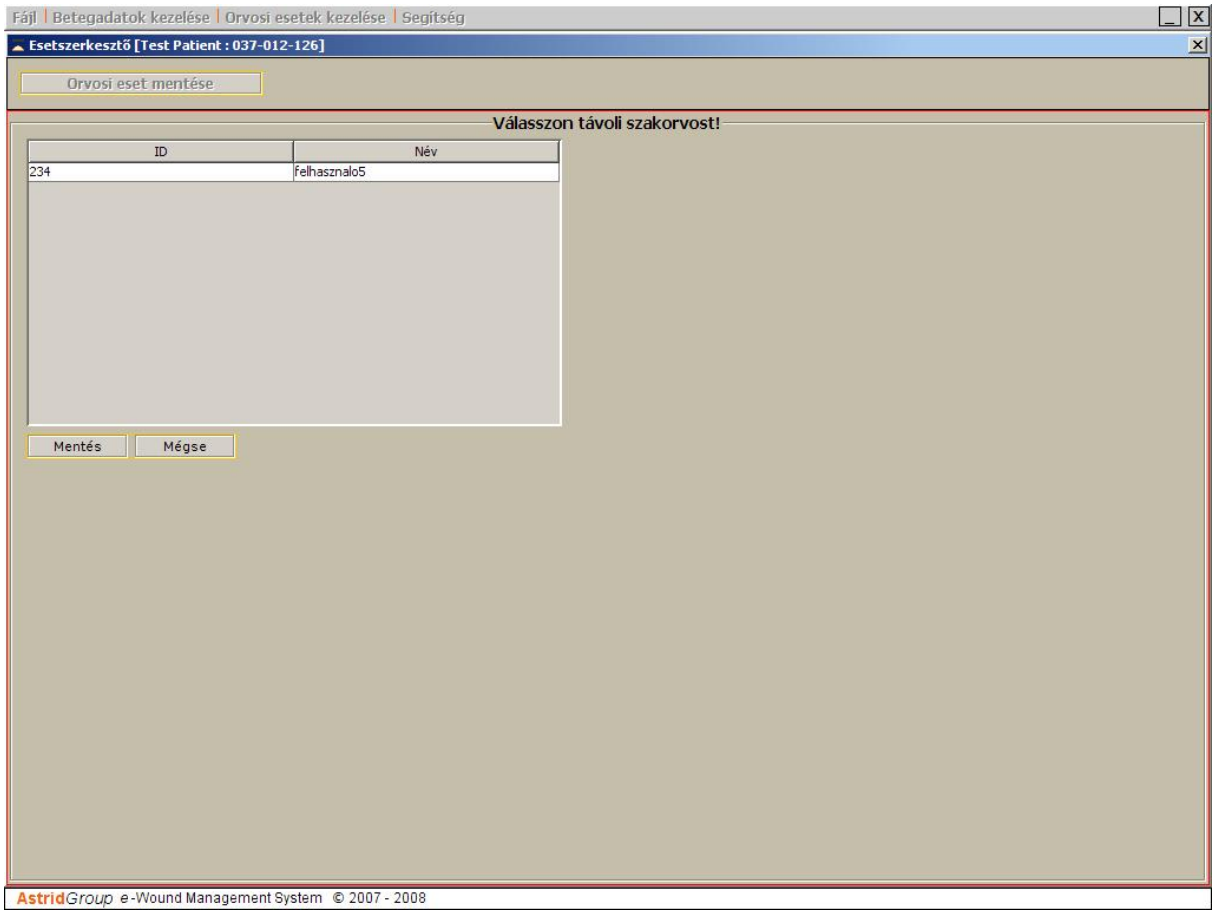

21. ábra. Diagnózis/Epikrízis fül

Az orvosi eset menése az Orvosi eset mentése gombra kattintással történik.

Az új orvosi eset mentése lehetséges, ha a megnyitott füleken ki van töltve minden kötelezően kitöltendő mező, ellenben egy dialógus ablakban figyelmeztetést kapunk arról, hogy melyik fülön hiányos a kitöltés. Az adott fülön a hiányosan kitöltött mező neve pirossal jelenik meg, ezzel is könnyítve a kitöltetlen mező megtalálását.

Ha diagnózis és epikrízis is készült az új orvosi esethez, akkor az Orvosi eset mentése gombra kattintás után lehetőség van az orvosi esethez egy távoli szakorvost társítani (22. ábra). A választható távoli szakorvosok azonosítói és nevei táblázatba rendezve jelennek meg, ezek közül egyet kiválasztva és utána a Mentés gombra kattintva mentésre kerül az orvosi eset.

Lehetőség van a Mégse gomb használatával visszalépni.




22. ábra. Orvosi eset távoli szakorvoshoz rendelése

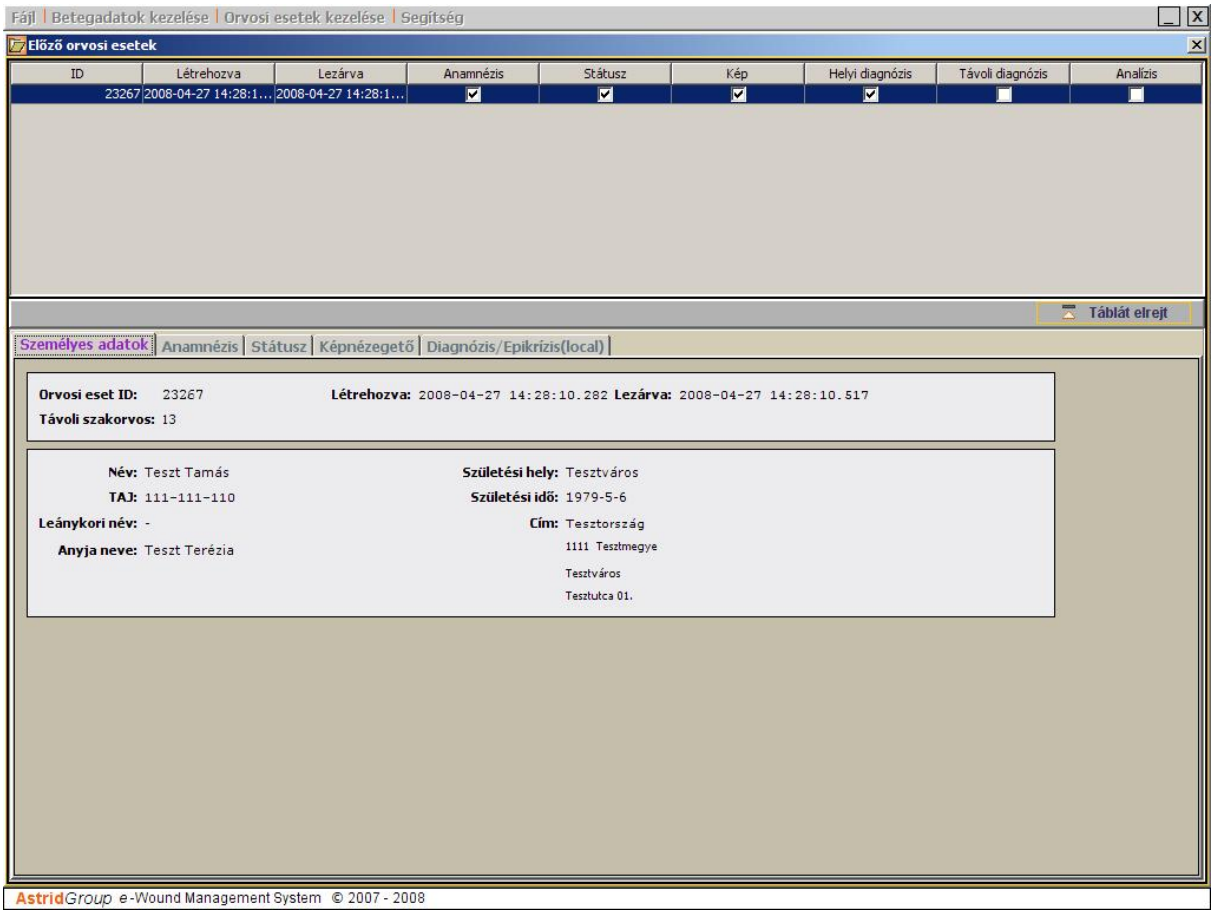
### *5.5.2. Orvosi esetek megtekintése*

Ezt a funkciót egyaránt elérheti a helyi szakorvos és a távoli szakorvos, az Orvosi esetek megtekintése menüpontra kattintással.

A helyi szakorvos esetében egy adott beteghez tartozó orvosi esetek megtekintésére az adott beteg TAJ számának helyes megadása után van lehetőség.

A távoli szakorvos esetében csak a távoli szakorvoshoz rendelt orvosi esetek megtekintésére van lehetőség. A Személyes adatok fül nem elérhető a távoli szakorvos számára, és nem látja a helyi szakorvos által az adott orvosi esethez készített diagnózist.

A megjelenő komponens megjelenésében és funkcionalitásában megegyezik az Előző orvosi esetek fülnél ismertetettekkel.




23. ábra: Orvosi esetek megtekintése

### *5.5.3. Orvosi esetek szerkesztése*

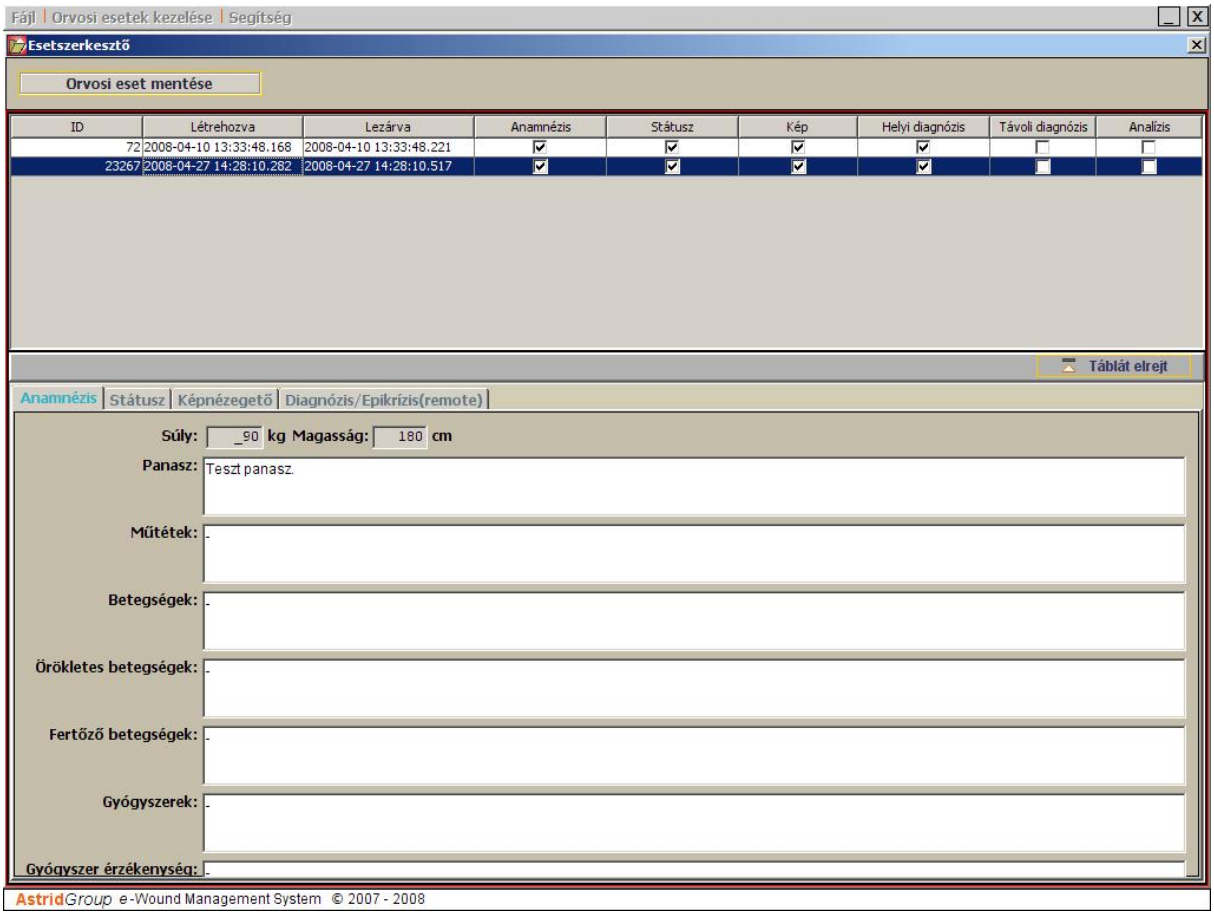
Ez a funkciót a helyi szakorvos, a távoli szakorvos, és a specialista szakorvos érheti el.

A helyi szakorvos esetében egy adott beteghez tartozó nem befejezett orvosi eseteket lehet megnyitni szerkesztésre.

A távoli szakorvos a hozzá rendelt eseteket tudja megnyitni szerkesztésre.

A specialista szakorvos pedig azokat az orvosi az orvosi eseteket tudja megnyitni szerkesztésre, amelyekhez már tartozik helyi és a távoli szakorvos által készített diagnózis és epikrízis.

A 24. ábra egy távoli szakorvos jogosultsággal rendelkező felhasználó esetében megnyíló esetszerkesztő ablakot szemlélteti.




24. ábra: Orvosi esetek szerkesztése (távoli szakorvos esetén)

### *5.6. Segítség menü*

Mindegyik felhasználó típus számára elérhető menüpont. A rendszerről, az egyes menüpontokról és használati esetekről tartalmaz információkat.

# 6. Összefoglalás

Ez volt az első olyan projektem, amit egy szoftverfejlesztéssel foglalkozó cég keretein belül végezhettem. A fejlesztés folyamán elmélyíthettem ismereteimet a Java objektumorientált programozási nyelv terén, a PostgeSQL, a NetBeans IDE, a Swing API nyújtotta lehetőségekben. Új ismeretekre tehettem szert a JBoss alkalmazásszerverrel, a JSP, a Java Servlet, az EJB technológiákkal kapcsolatban.

A rendszerrel kapcsolatban további tervek között szerepel az orvosi dokumentumok kinyomtatásának biztosítása. A hitelesség megvalósítása érdekében az elvégzett műveletek digitális aláírása és időbélyegzése, amire azért van szükség, hogy az adott szakorvos által elvégzett művelet egyértelműen beazonosítható és visszakereshető legyen.

Jelenleg az alkalmazás belső tesztelés alatt áll, remélem a közeljövőben az egészségügyben sikeresen tudják majd használni.

# 7. Irodalomjegyzék


[1] Nyékyné Gaizer Judit, Java2 útikalauz programozóknak, 2001

[2] Hans Bergsten, Java szervletek programozása, 2002

[3] Hans Bergsten, JavaServer Pages, 2001

[4] Bill Burke, Richard Monson-Haefel, Enterprise JavaBeans 3.0

[5] http://www.jboss.org/jbossas/docs/index.html

[6] http://java.sun.com/products/ejb/

[7] http://hu.wikipedia.org/wiki/BNO

[8] http://www.netbeans.org/index_hu.html

[9] http://www.commandprompt.com/ppbook

[10] http://www.eski.hu/new3/adatok/adatok.php#kodok

# 8. Függelék


a.) Az orvosi adatokhoz kapcsolódó Entity Bean-ek osztálydiagramja