Hubasky magánkórház

Rendszerterv

(Kidolgozás)

Készítette:

Hazai Péter – Projektvezető

Owczarek Artur – Adminisztrátor

Stricker Balázs – Demonstrátor

Breier Balázs – Kapcsolattartó

Tartalom

A megrendelő igényei ... 3

A rendszer környezete ... 4

A rendszer szerkezete ... 5

HospitalManagement alrendszer ... 6

HospitalManager osztály ... 7

Unit osztály ... 7

Hospital osztály ... 8

Department osztály ... 8

Ward osztály ... 9

Person osztály ... 9

Employee osztály ... 9

IManageableUnit interfész ... 10

IManageableEmployee interfész ... 12

PatientManagement alrendszer ... 18

PatientManager osztály ... 19

Patient osztály (Person osztály), MedicalRecord osztály ... 19

Procedure osztály ... 20

Betegfelvétel ... 21

Egészségügyi eljárás eredményének rögzítése ... 21

Egészségügyi eljárás kiírása ... 22

Eszköz törlése ... 23

Kórtörténet szerkesztése ... 23

Számla kiadása ... 24

Interface: IInventoryManagement ... 25

Interface: IPatientManager ... 25

Normálistól *eltérő használati esetek* ... 27

InventoryManagement alrendszer ... 28

IInventoryManager interfész ... 28

InventoryManager osztály ... 28

InventoryItem osztály ... 28

InventoryType enumeráció ... 28

ApplicationManagement alrendszer ... 31

IApplicationManagement interfész ... 31

ApplicationManager osztály 31
ApplicationUser osztály 31
Role enumeráció 31
WebDataManagement alrendszer 33
WebDataManagement 33
Web felület 34
1. melléklet – Használati eset modell 38
Páciens 39
Saját kórtörténet megtekintése: 39
Adminisztrátor 40
Alkalmazott kezelése: 40
Szervezeti hierarchia kezelése: 40
Adatrögzítő 41
Betegfelvétel 41
Beteg elbocsátás 41
Labor technikus 42
Eljárás kezelése 42
Orvos 43
Eljárás kezelése 43
Ápoló 44
Eljárás kezelése 44
2. melléklet – Analízis modell 45
Csomagdiagram 45
Kommunikációs diagramok 46
Osztálydiagramok 51
3. melléklet – Jegyzőkönyvek 54

A megrendelő igényei

A projekt célja egy olyan grafikus felhasználói felülettel rendelkező asztali alkalmazás elkészítése, amely egy magánkórház ügyviteli és adminisztrációs folyamatait támogatja.

A magánkórház ügyfeleinek egészségügyi ellátást biztosít, melynek keretében az ügyfél által választott, vagy éppen a szakorvos által ajánlott kezeléseket, terápiákat az ügyfél az állami egészségügyi rendszeren kívül igénybe veheti. Az ügyfelek megfelelő kiszolgálása érdekében ezért szükség van egy egyszerűen kezelhető, jól működő adminisztrációs eszközre, amelyben a betegek, a kezelésükhöz szükséges berendezések, tárgyi eszköz nyilvántartására lehetőség van, valamint a kórház részleteit is menedzselni lehet.

Mivel betegellátással kapcsolatos bizalmas adatokról és nagyfokú felelősséggel járó eljárások adminisztrálásáról van szó, ezért fontos szempont a biztonságos adattárolás és jogosultságok felhasználói szerepkörök szerinti kezelése. Kritikus műveletek kezdeményezése esetén (pl. műtét) szükséges lehet az ismételt azonosítás. Az adatok egy központi adatbázis szerverről érhetőek el, ez az adatok jogosultságok mentén történő differenciált hozzáférését is lehetővé teszi.

A szoftver moduláris felépítésű, cél, hogy a kezdeti modulok által nyújtott funkcionalitás a jövőben további modulok hozzáadásával egyszerűen bővíthető legyen. A megrendelő négy modul elkészítését várja el, sikeres szállítás esetén pedig lehetőség szerint további modulokat is rendelne.

A szoftverrendszer által kezelt adatbázist mind grafikus, mint parancssoros felületen lehet kezelni, az adatbázis jellege miatt azonban bizonyos adminisztrációs és lekérdező parancsok megvalósításának grafikus implementációját nem végezzük el (az iparági sztenderd szerint ezek parancssorban kezelt funkciók (pl. View létrehozása stb.)). A szoftver grafikus kezelői felülettel rendelkezik, mely lehetőséget nyújt az alábbi tevékenységek elvégzésére:

A betegnyilvántartó felületen új beteg felvétele, adatváltoztatás lehetősége valamint a kórtörténet megjelenítése lehetséges. Minden szerepkör használja, első alkalommal a betegirányító betegfelvétel eljárást ír ki, amivel hozzárendeli a beteget egy szakosztályhoz. A szakorvosok előjegyezhetik a pácienseket különböző eljárásokra (vizsgálat, kezelés), ezek eredményei ide kerülnek feltöltésre az egyéb betegellátással kapcsolatos dokumentumokkal egyetemben. Minden eljárásnak meghatározott díja van, mely az eljárás után hozzáíródik a beteg egyenlegéhez. A zárójelentés kézhezvételekor a beteg megkapja a kezelési költségek összesítő számláját, melyet köteles kiegyenlíteni.

Az eszköznyilvántartó felületen lehet a kórház eszköz parkját listázni (pl. általános eszközök, mint gézlap, gyógyszerek, orvosi eszközök, ruházat stb.). Bizonyos kezelések eszközigényének kezelésére ad lehetőséget.

A kórházi hierarchia adminisztrációs felületen lehet a kórház struktúráját felépíteni, osztályokat, részlegeket létrehozni, törölni, alkalmazottak adatait elérni, létrehozni, módosítani, törölni, alkalmazottakat szervezeti egységekhez rendelni, vezetőket kijelölni. Kiegészül még a kórházépület logikai modelljével, az egyes kórtermek, laborok adataival is.

A regisztrációval rendelkező ügyfelek egy weblapon keresztül autentikáció után hozzáférnek a kezelési adataikhoz, a kezelésük során keletkezett dokumentumokhoz.

A rendszer környezete

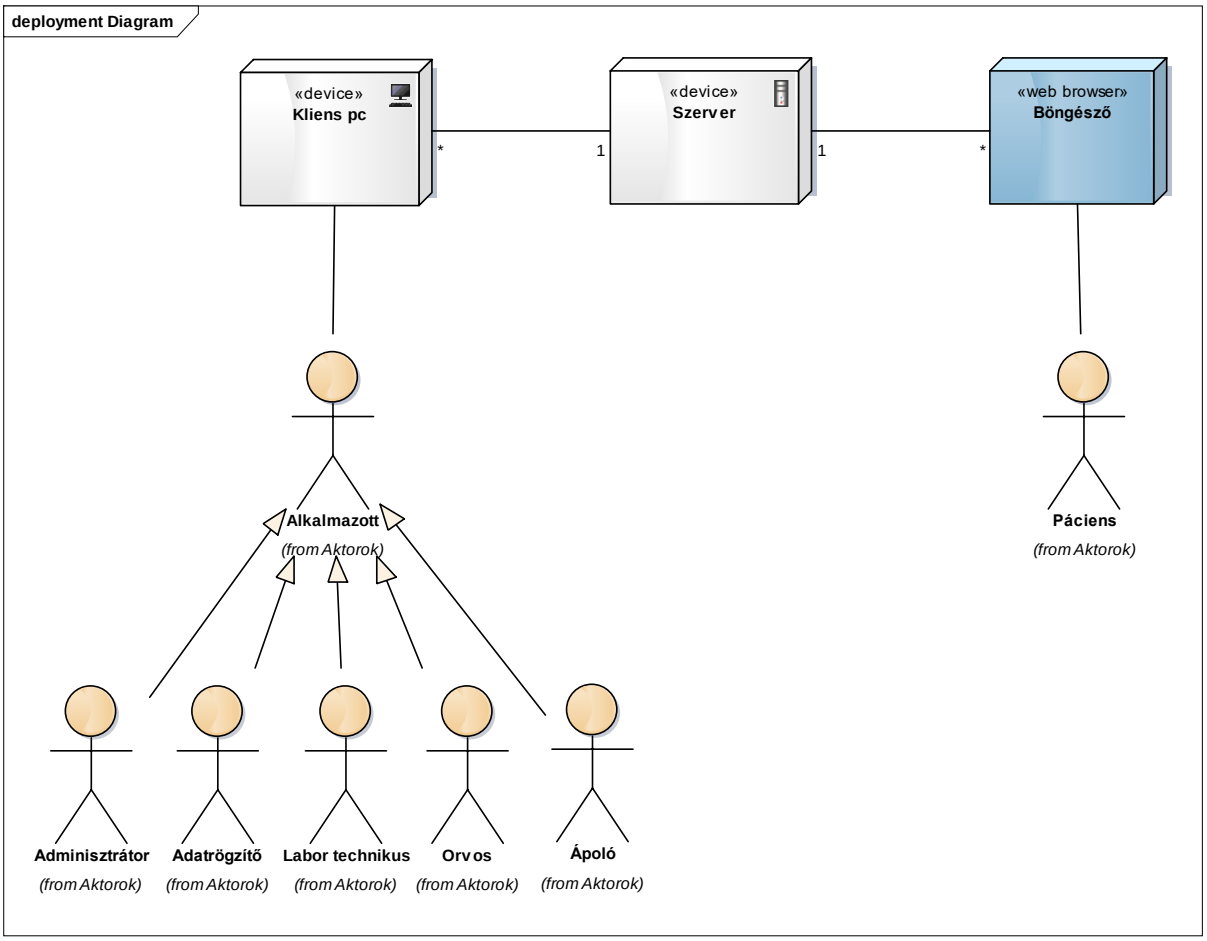
A kórházban kialakított rendszer magját egy központi szerveren futó adatbázis szolgáltatás (MySQL) adja, melyet különböző jogosultsági szinteken lehet elérni belső hálózaton (intraneten) a kliensalkalmazással vagy böngésző segítségével interneten keresztül.

A központi szerver operációs rendszere tulajdonképpen tetszőleges lehet, az alábbi funkcionalitásokat kell kiszolgálnia:

- MySQL adatbázis szerver
- Webserver szolgáltatás (Java – Glassfish, php)
- Tűzfal

Az igényeket figyelembe véve az ajánlott operációs rendszer az Ubuntu Server 14.04.4 LTS (64 bit).

A kórház különböző osztályain, orvosi szobák, laborok, irodák és a recepció is egy-egy kliens számítógéppel rendelkeznek, melyeken Windows 7, vagy ennél újabb Windows operációs rendszer üzemel (8, 8.1, 10 stb).

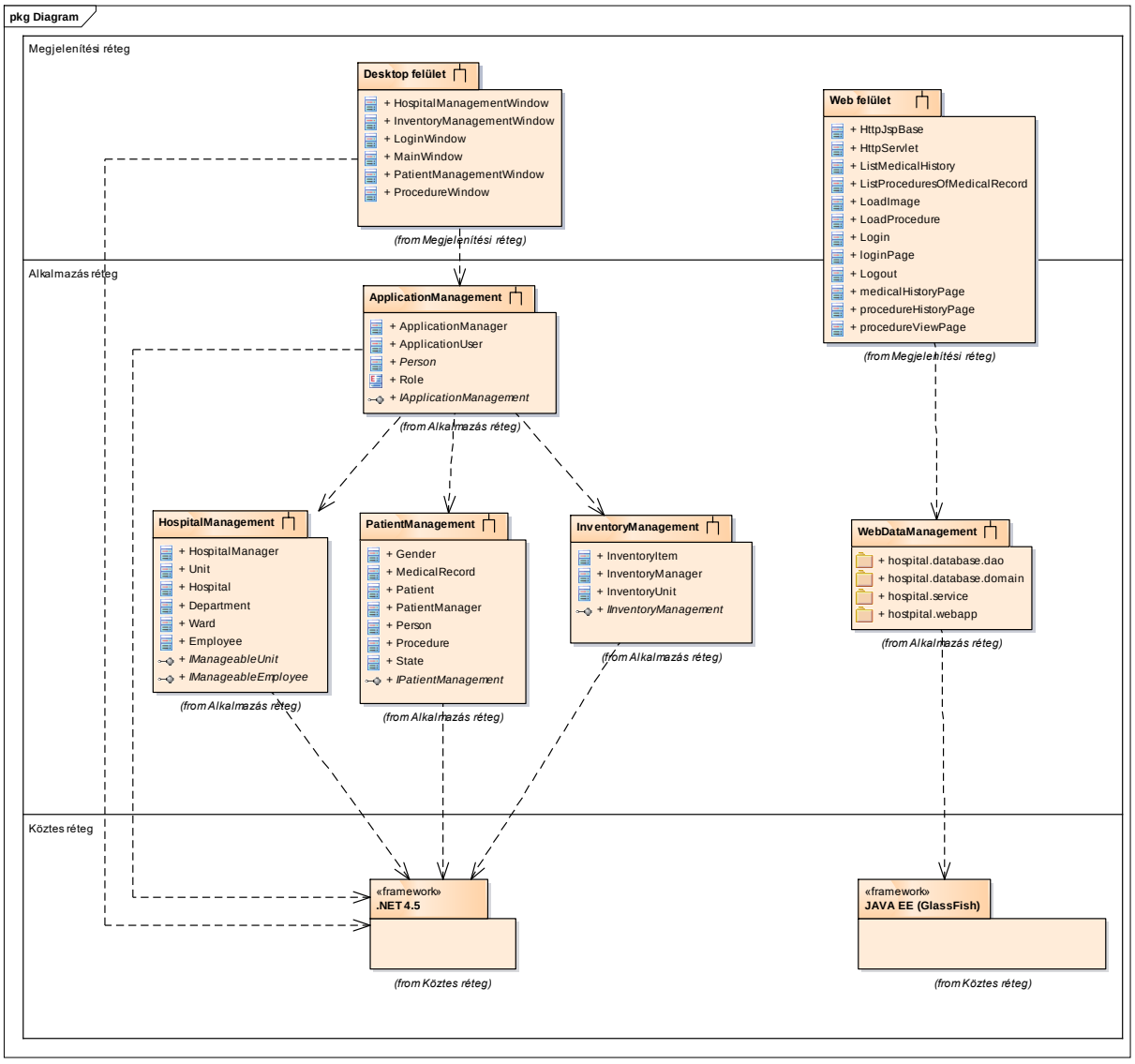


1. ábra - A rendszer környezete

A rendszer felépítését figyelembe véve a kliensalkalmazást a Microsoft .NET 4.5 keretrendszerre építve fejlesztjük, készítjük el.

A rendszer szerkezete

A rendszert a kívánt moduláris felépítést figyelembe véve alrendszerekre osztottuk fel, meghatároztuk a rendszer környezetét alkotó azon elemeket melyek az implementáláshoz és a későbbi működéshez szükségesek.



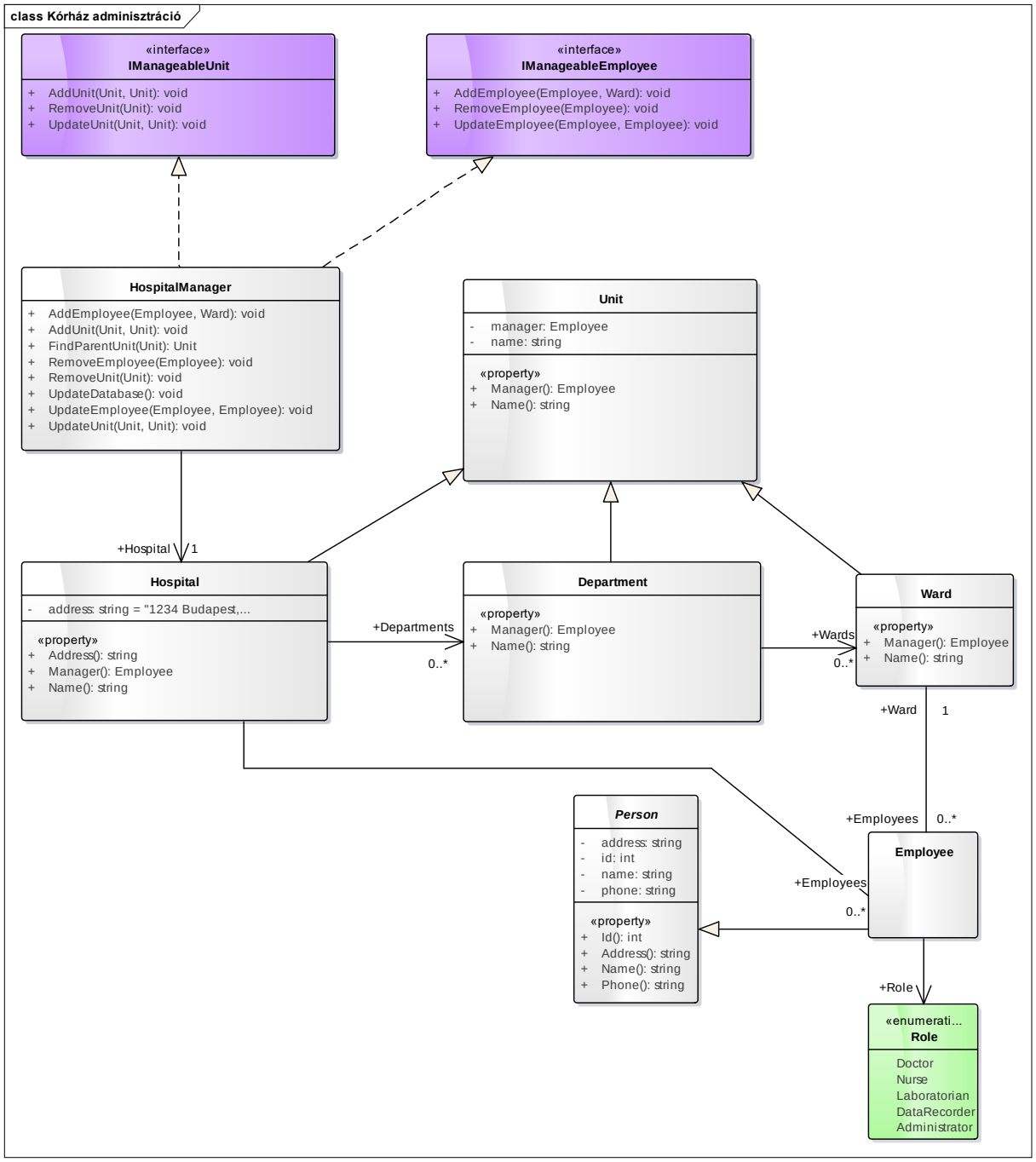
A megjelenítési szinten két alrendszer található, ezek tartalmazzák az asztali és webes felhasználói felületek megjelenítéséhez szükséges osztályokat.

Az alkalmazási szinten a HospitalManagement, PatientManagement és InventoryManagement alrendszerek biztosítják az asztali alkalmazás fentiekben már részletezett moduljainak működését. Az ApplicationManagement alrendszer kezeli a bejelentkezett felhasználót, annak jogosultságát és az alrendszer többi moduljának használatához szükséges adatokat. Itt található a WebDataManagement alrendszer, mely a webes felületnek biztosít hasonló funkcionalitást.

A köztes szinten található a .NET keretrendszert szimbolizáló csomag, a rendszerelemek közül közvetlenül nem használ fel az alkalmazás.

HospitalManagement alrendszer

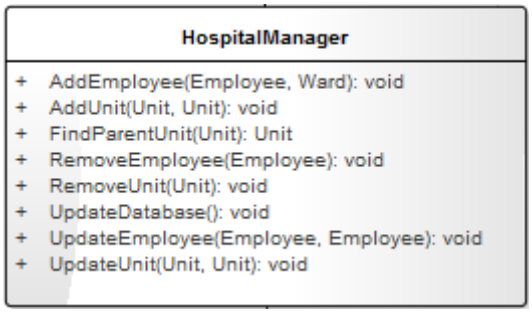
A kórházi intézmény hierarchia kezelésére szolgáló modul, melyben két fő részre oszthatók az elvégezhető műveletek. Az egyik az intézeti egységek, másik pedig az alkalmazottak kezelésére szolgáló funkciókat tartalmazza.



2. ábra - Kórház adminisztráció alrendszer osztályai

HospitalManager osztály

Az alrendszer aktív osztálya, melyből az alkalmazás futása során egyetlen példány készül és végigköveti az alkalmazás élettartamát.



3. ábra - HospitalManager osztály

A 2. ábrán látható, hogy a HospitalManager osztálynak egy Hospital típusú (és megnevezésű) adattagja van, tulajdonképpen ezen keresztül érhető el a teljes hierarchiai szerkezet. Az adattag elérését a tulajdonságoknál megszokott módon a „get” művelettel végezzük el.

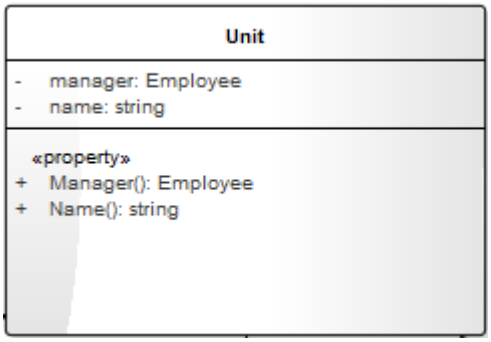
Hat alap funkciót ellátó metódust implementál az osztály, melyek interfészen keresztül hívhatók meg. Ezek segítségével lehet a kórházi rendszer hierarchiáját szerkeszteni. A két fő csoportja a műveleteknek: intézményi egység kezelés valamint alkalmazott kezelés. Mindkét csoportban három metódus van, Add, Remove és Update. Ezek sorban, ahogy a nevük is utal rá, a hozzáadás, eltávolítás és a módosítás.

Mivel az egyes hierarchiai szintek lista szerkezetekben kerülnek tárolásra, egy segéd metódusra is szükség van, nevezetesen a FindParentUnit()-re, hogy intézeti egység eltávolítása során az törölhető legyen a lista szerkezetből.

Az UpdateDatabase() egy segéd metódus, mely az egyes elemek módosításának érvényesítésekor kerül meghívásra. Használatával a módosított értékek beírásra kerülnek az adatbázisba és a változtatás visszavonhatatlan lesz.

Unit osztály

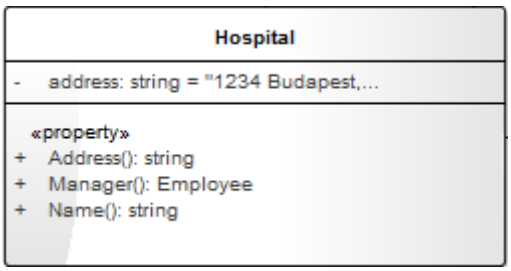
Ez az osztály szolgál ősosztályként az összes intézményi egység számára. Tulajdonképpen csak két adattagot tartalmaz, melyeket tulajdonságokon keresztül érhetünk el. Ezek a manager, amely egy Employee típusú adattag, referenciát tartalmaz arra az alkalmazottra, aki az intézeti egységet vezeti, másik pedig a name, ami string típusú, az osztály megnevezését tartalmazza.



4. ábra - Unit osztály

Hospital osztály

A Unit osztály egy leszármazott osztálya, az alkalmazás futási ideje során egyetlen objektumpéldány készül belőle a program indításakor.



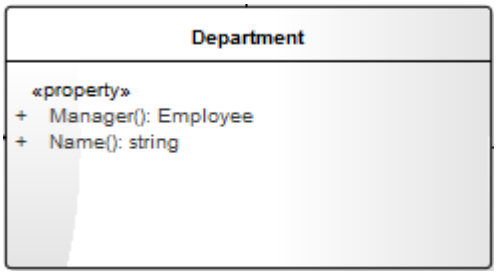
5. ábra - Hospital osztály

Az örökölt adattagok mellett egy address megnevezésű, string típusú adattaggal specializálja az ősosztályt, melyben a kórház címét tárolja. A 2. ábrán látható, hogy egy Departments nevű lista adatszerkezetben tárolja a kórház osztályait.

Van még egy lista adatszerkezete az osztálynak, mely az Employees névre hallgat és Employee típusú elemekből áll. Ez a lista egy referencia lista a kórház összes alkalmazottjának (beleértve a managereket is) objektumára.

Department osztály

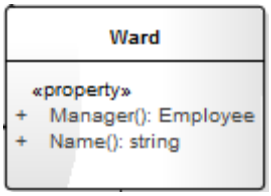
Szintén egy leszármazott osztálya a Unit osztálynak, és ez az osztály is tartalmaz egy lista adatszerkezetet, melynek megnevezése Wards. Ebben tárolja az osztály alá tartozó intézeti egységeket.



6. ábra - Department osztály

Ward osztály

Unit osztály leszármazottja. Ez az intézeti hierarchia alsó eleme, ez az osztály tárolja Employees nevű lista adatszerkezetben az alkalmazottakat (Employee típus).

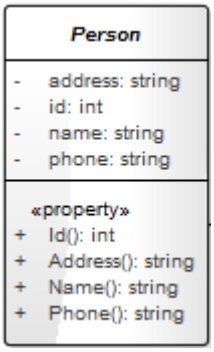


7. ábra - Ward osztály

Minden alkalmazott valamelyik Ward példány objektumhoz van hozzárendelve. A program működésének kell biztosítania azt, hogy egy alkalmazott egyszerre csak egy Wardhoz tartozzon. Az intézeti egység vezetők kivételek ez alól, ők ebben a listában nem szerepelnek, az ő példány objektumukra az egyes intézeti egység objektumának manager adattagja tartalmaz referenciát.

Person osztály

Ősosztályként szolgál az alkalmazásban minden személy számára (alkalmazott vagy páciens), és ennek megfelelően a szükséges alapvető információk tárolásáért felelős. Ezek az address acím, id mint azonosító, name a személy neve és phone mely a személy telefonszámát tartalmazza string típusként. Az adattagokat tulajdonságokon keresztül lehet elérni, módosítani.



8. ábra - Person osztály

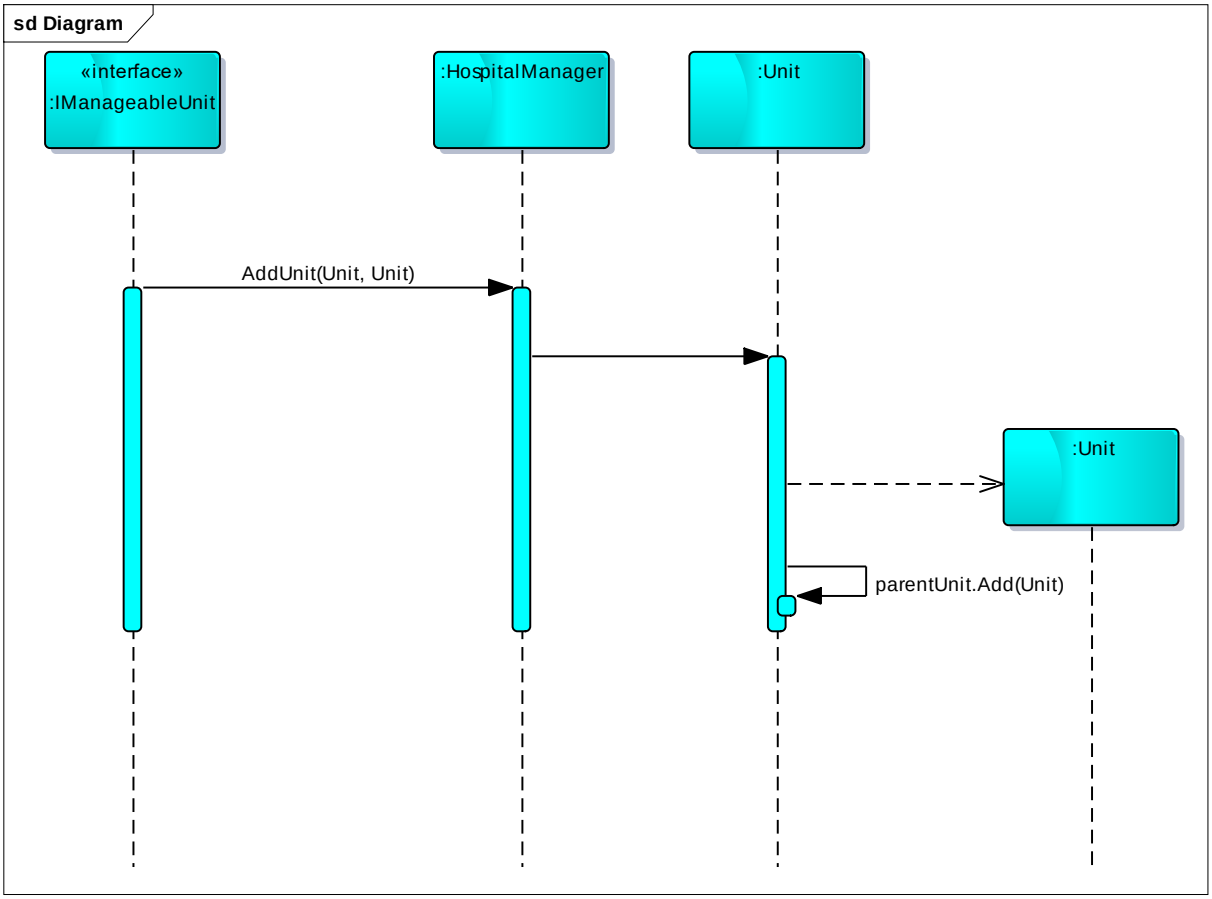
Employee osztály

Leszármazottja a Person osztálynak, mely további két adattaggal specializálja ősosztályát: Role és Ward. Előbbi egy Role típusú Enum, mely az alkalmazottak típusait tárolja (orvos, ápoló, labor technikus, adatrögzítő, adminisztrátor). Utóbbi pedig egy referencia, mely azt mutatja meg, hogy az adott alkalmazott melyik intézeti egységhez tartozik.

IManageableUnit interfész

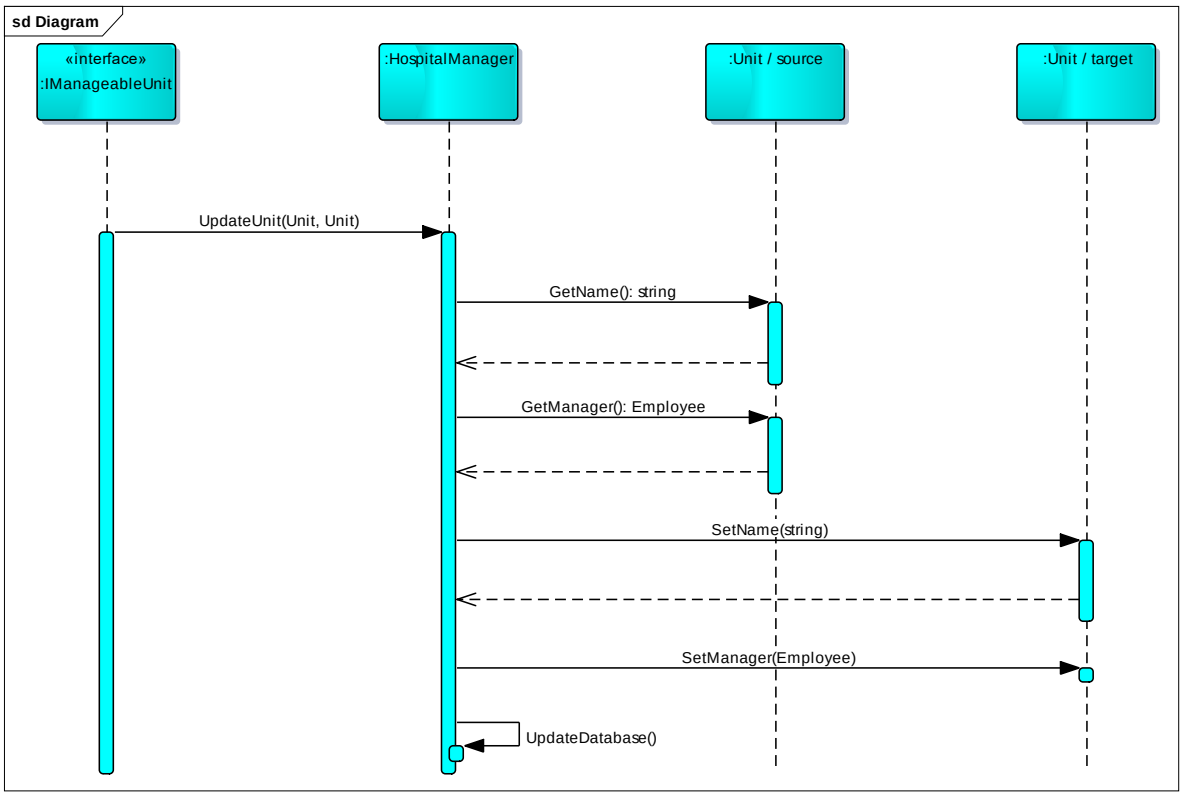
A HospitalManager osztály két interfészt implementál, és az egyik az IManageableUnit interfész. Három metódust tartalmaz, melyek az intézeti egységek kezelésére szolgálnak (AddUnit(), RemoveUnit(), ModifyUnit()).

Az AddUnit() metódussal paraméterként átadott Unit típusú objektum példányunkat a szintén paraméterként megadott, szintén Unit típusú objektum példányunk alá sorolhatjuk be. Bár a metódus Unit típusú referenciákat vár bemenő paraméterekként, a működéséhez nagyon fontos, hogy a specializált osztályokat használjuk azok hierarchiai eloszlása szerint. Ward típus csak Department típus alatt helyezkedhet el, Department típus pedig csak Hospital típus alatt. Általánosságban jellemző az alkalmazás működésére, hogy amennyiben hibás, rossz paraméterezés történik, az esetnek megfelelő kivételt dob a művelet végrehajtása során.



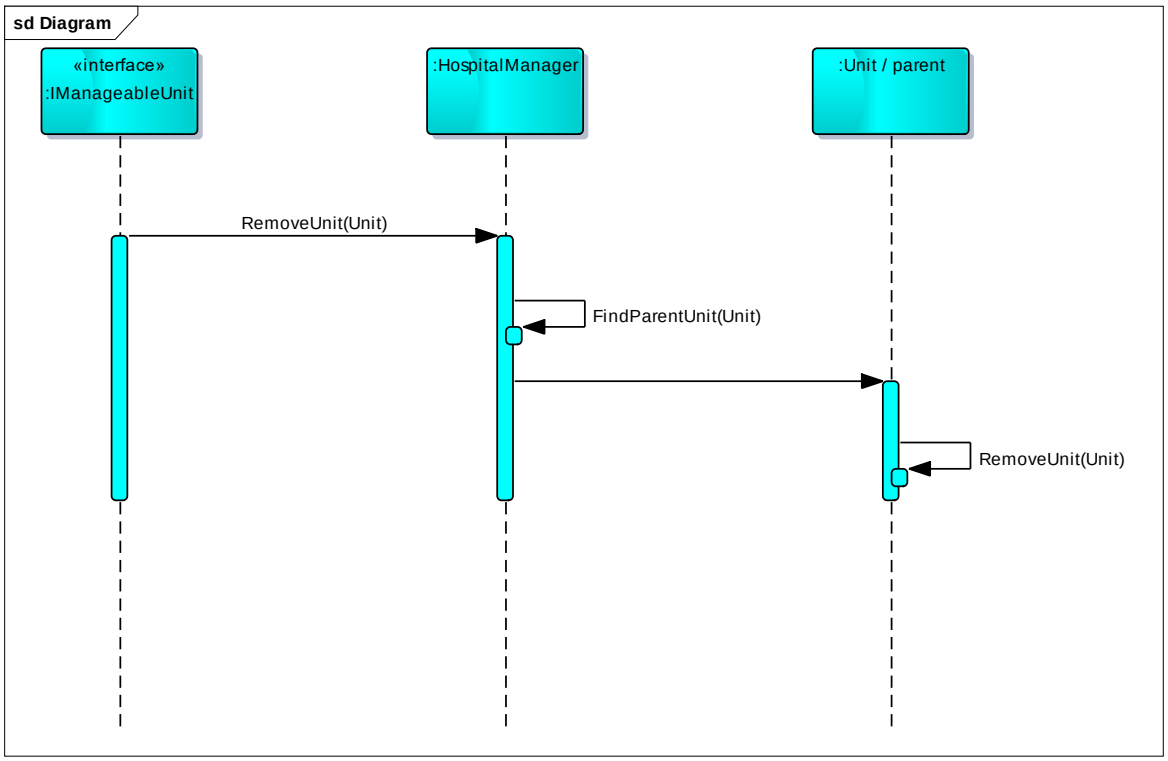
9. ábra - AddUnit() metódus

UpdateUnit() metódus paraméterként átadott Unit típusú referenciákat másolja. Az egyik paraméter a másolandó a másik pedig a cél referencia. A művelet elvégzése után az UpdateDatabase() metódus kerül meghívásra, mely módosítja az adatbázis tartalmat.



10. ábra - UpdateUnit() metódus

RemoveUnit() metódus a paraméterként átadott Unit típusú referenciát távolítja el az ő szülő objektum lista adatszerkezetéből. A szülő objektum példány keresésére a FindParentUnit() segédmetódus meghívásával van lehetőség.

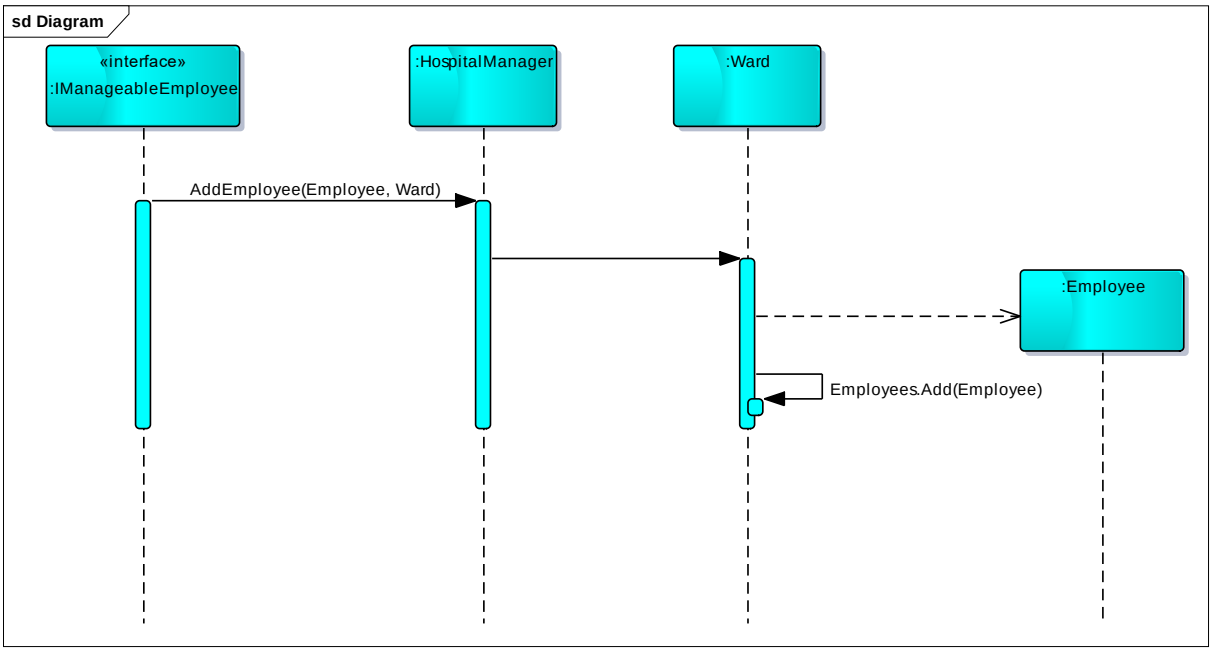


11. ábra - RemoveUnit()

IManageableEmployee interfész

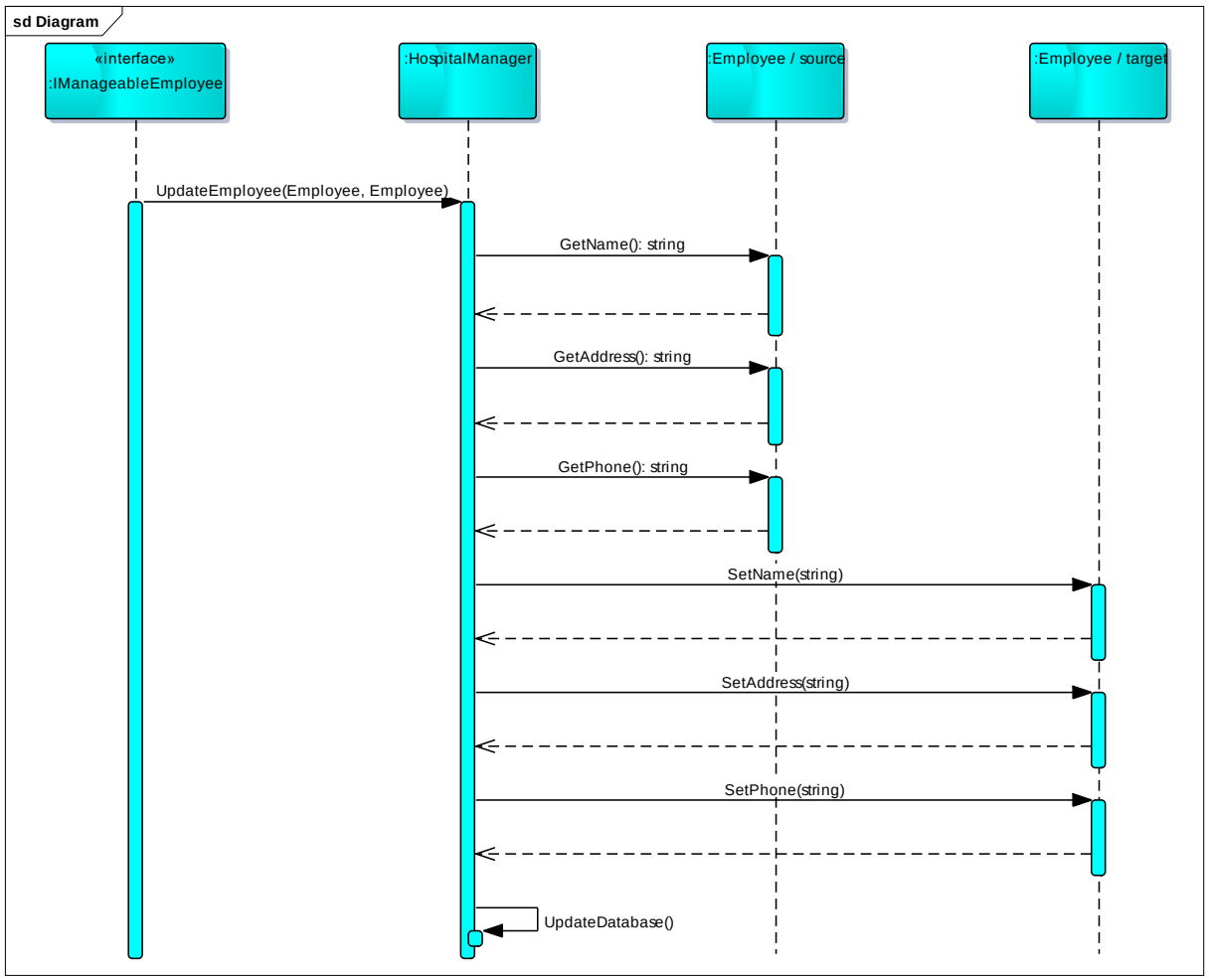
Ez az interfész is három metódust tartalmaz, hasonlatosan az intézeti egységek kezeléséhez itt is van egy hozzáadó, módosító valamint eltávolító művelet.

AddEmployee() metódussal lehet új alkalmazottat felvenni a kórházi kötelékbe. A létrehozott Employee objektum példány a művelet végrehajtása során beszúrásra kerül a megfelelő Ward típusú intézeti egység Employees lista adatszerkezetébe. Az alkalmazott id nevű adattagja a beillesztés során kerül meghatározásra.



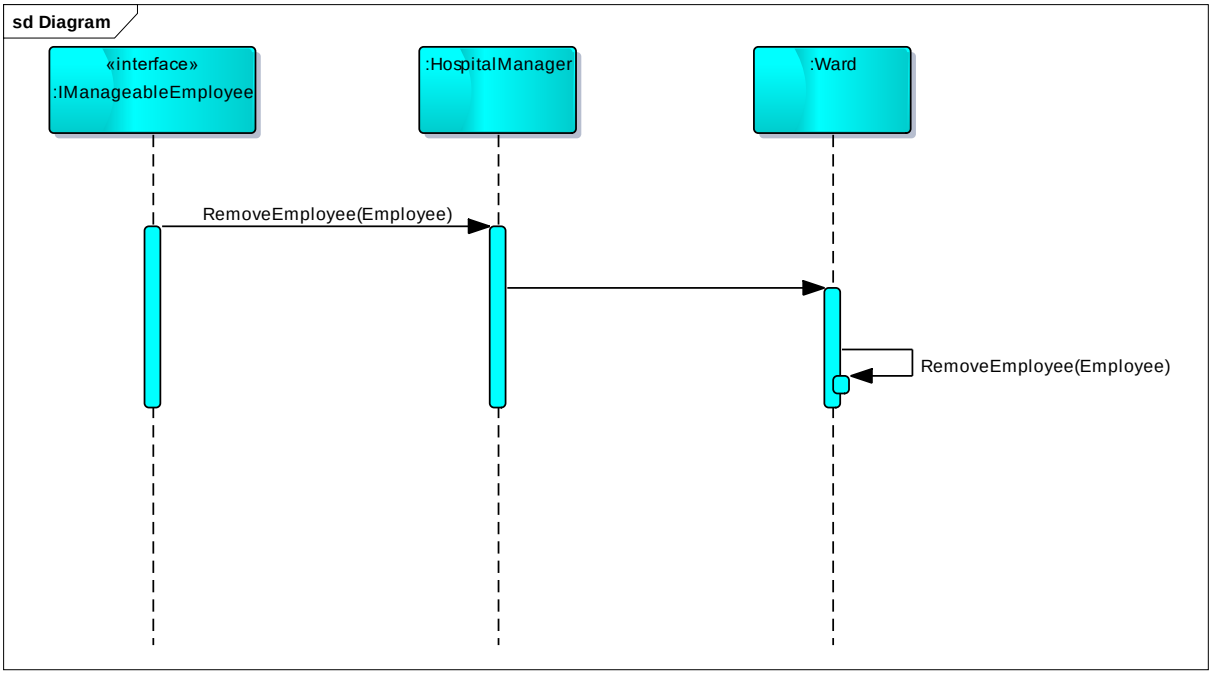
12. ábra - AddEmployee() metódus

UpdateEmployee() metódus működése szintén hasonló az UpdateUnit()-éhoz. Paraméterül egy kiinduló és egy cél alkalmazott objektum példány referenciát kér, majd a kiinduló alkalmazott adatait átmásolja a cél alkalmazott példányba, felülírva azt. A művelet végén az adatbázisban történő rögzítéshez a segédmetódus, UpdateDatabase() metódus kerül meghívásra.

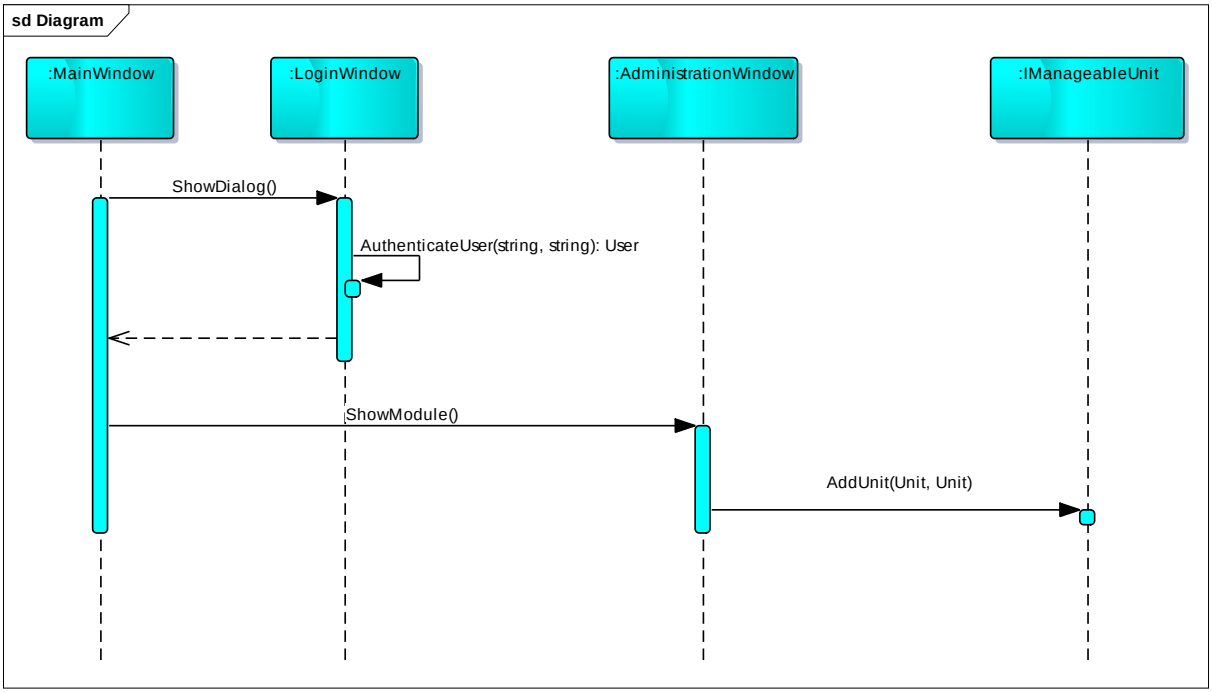


13. ábra - ModifyEmployee() metódus

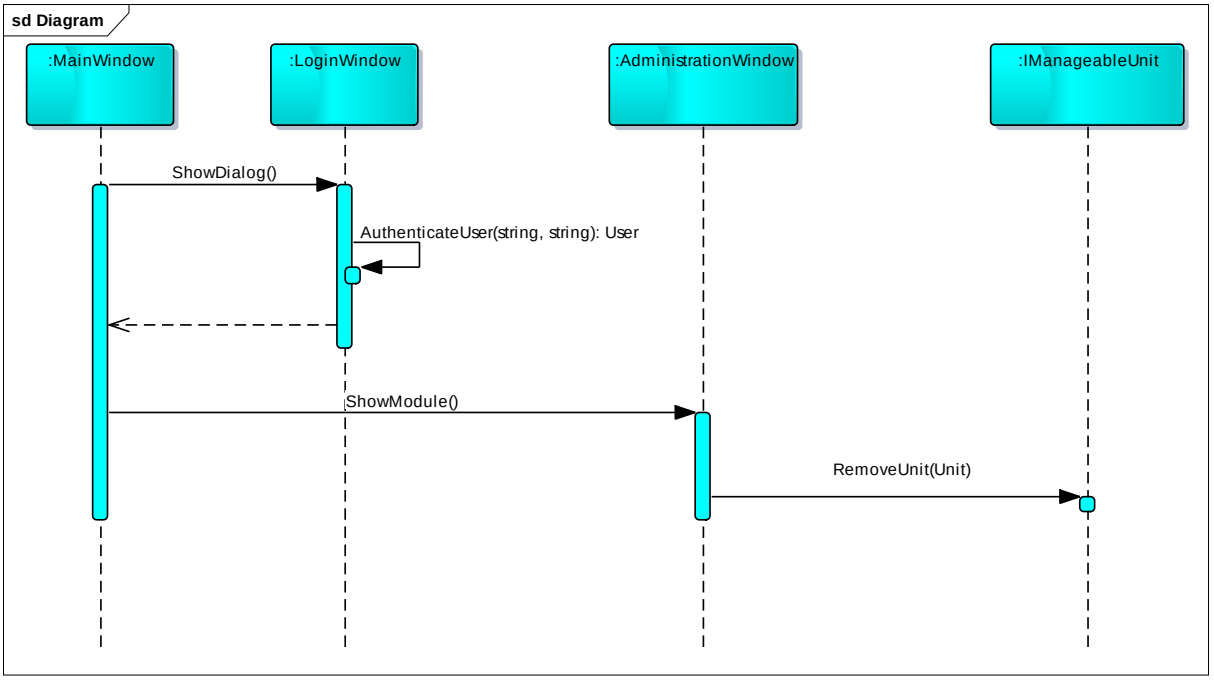
RemoveEmployee() metódus segítségével lehet egy alkalmazottat eltávolítani a hozzá tartozó Wardból. A metódus működése során az alkalmazottat minden listából eltávolítja, így az alkalmazott többé nem lesz tagja a kórházi dolgozóknak.



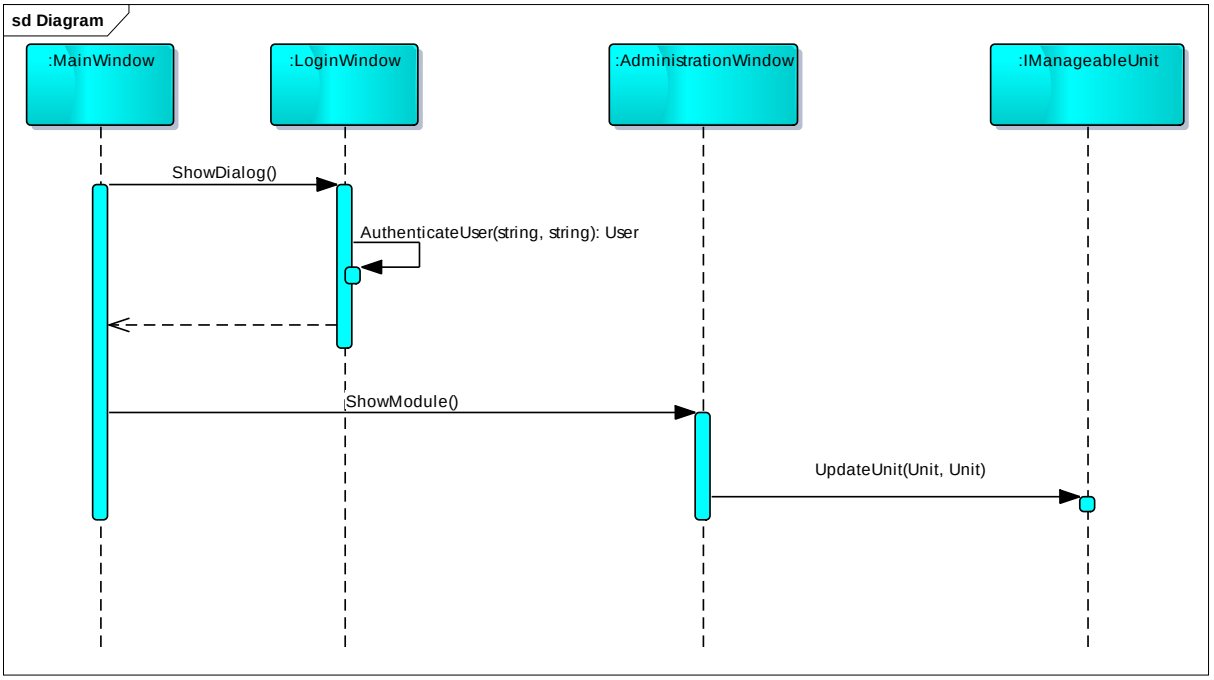
14. ábra - RemoveEmployee() metódus



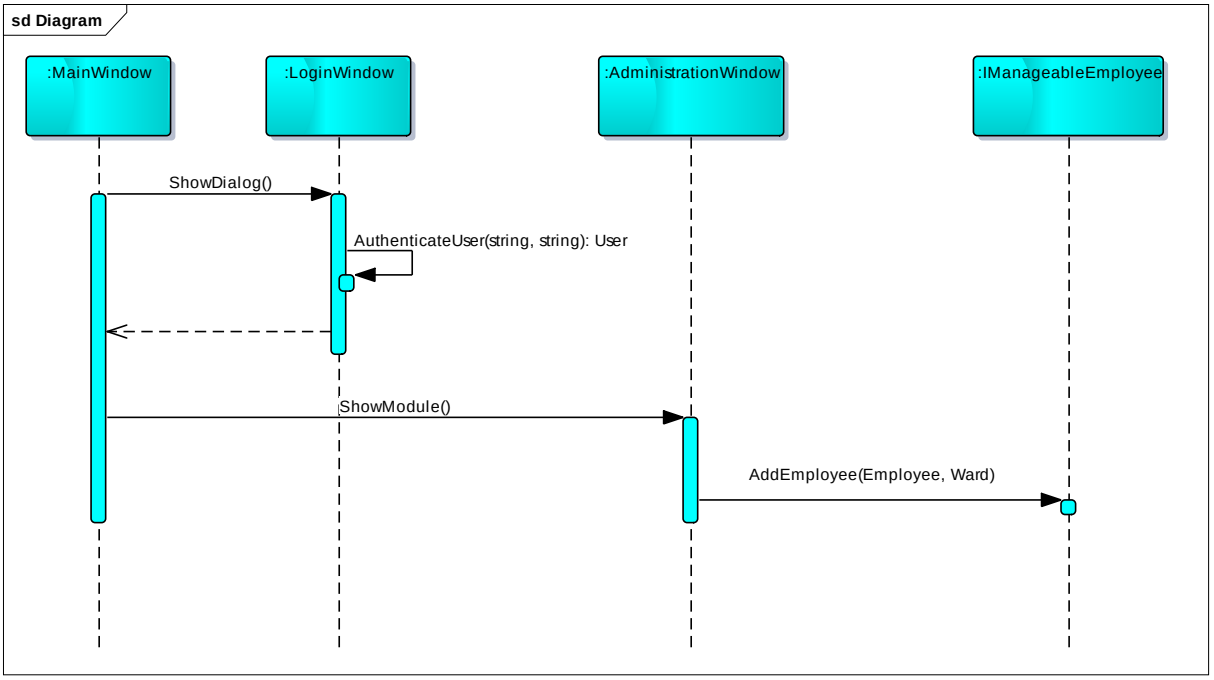
15. ábra - AddUnit() metódus hívása kezelőfelületről



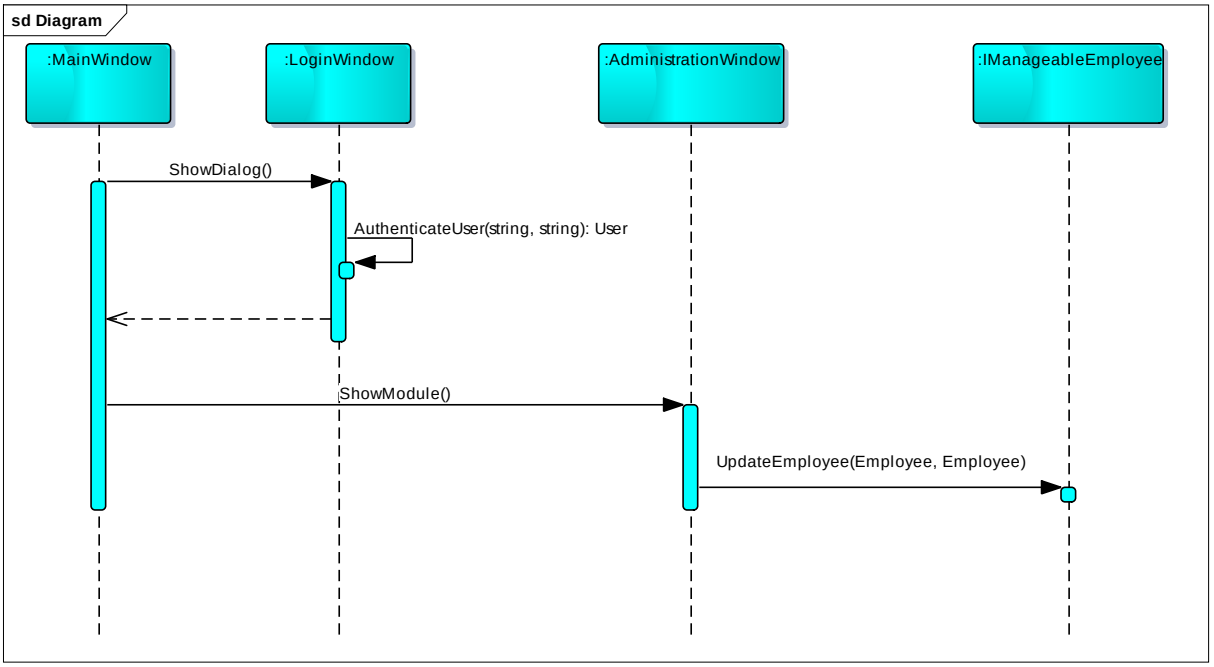
16. ábra - RemoveUnit() metódus hívása kezelőfelületről



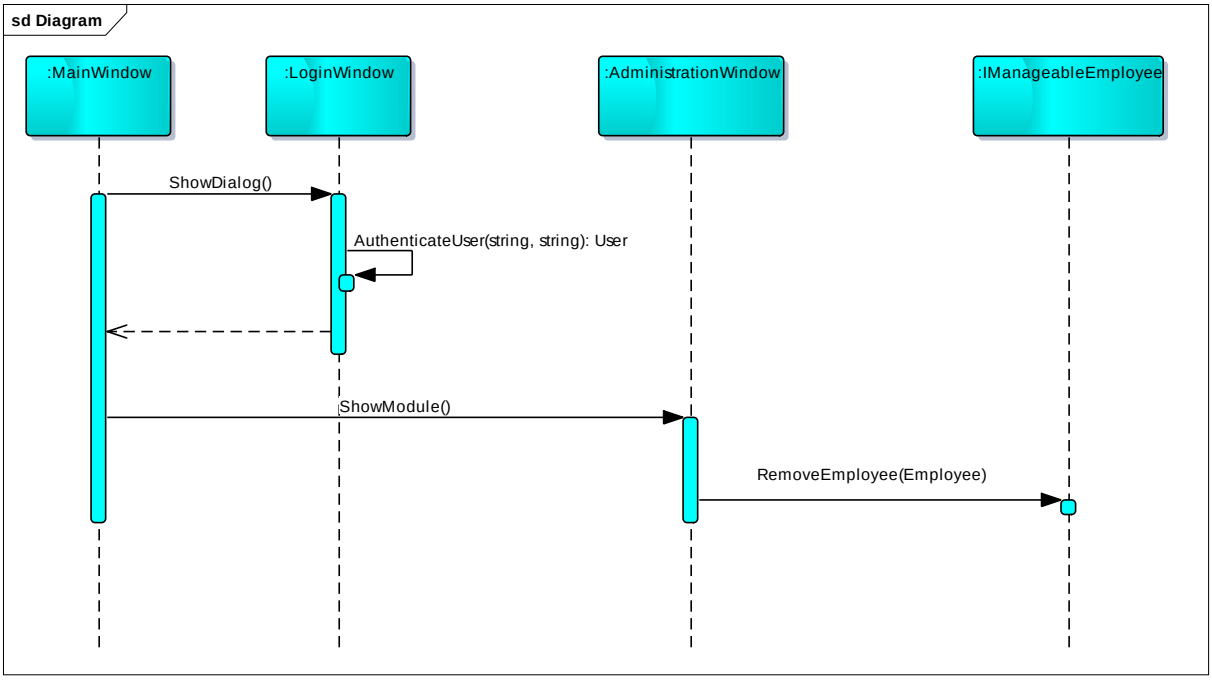
17. ábra - UpdateUnit() metódus hívása kezelőfelületről



18. ábra - AddEmployee() metódus hívása kezelőfelületről



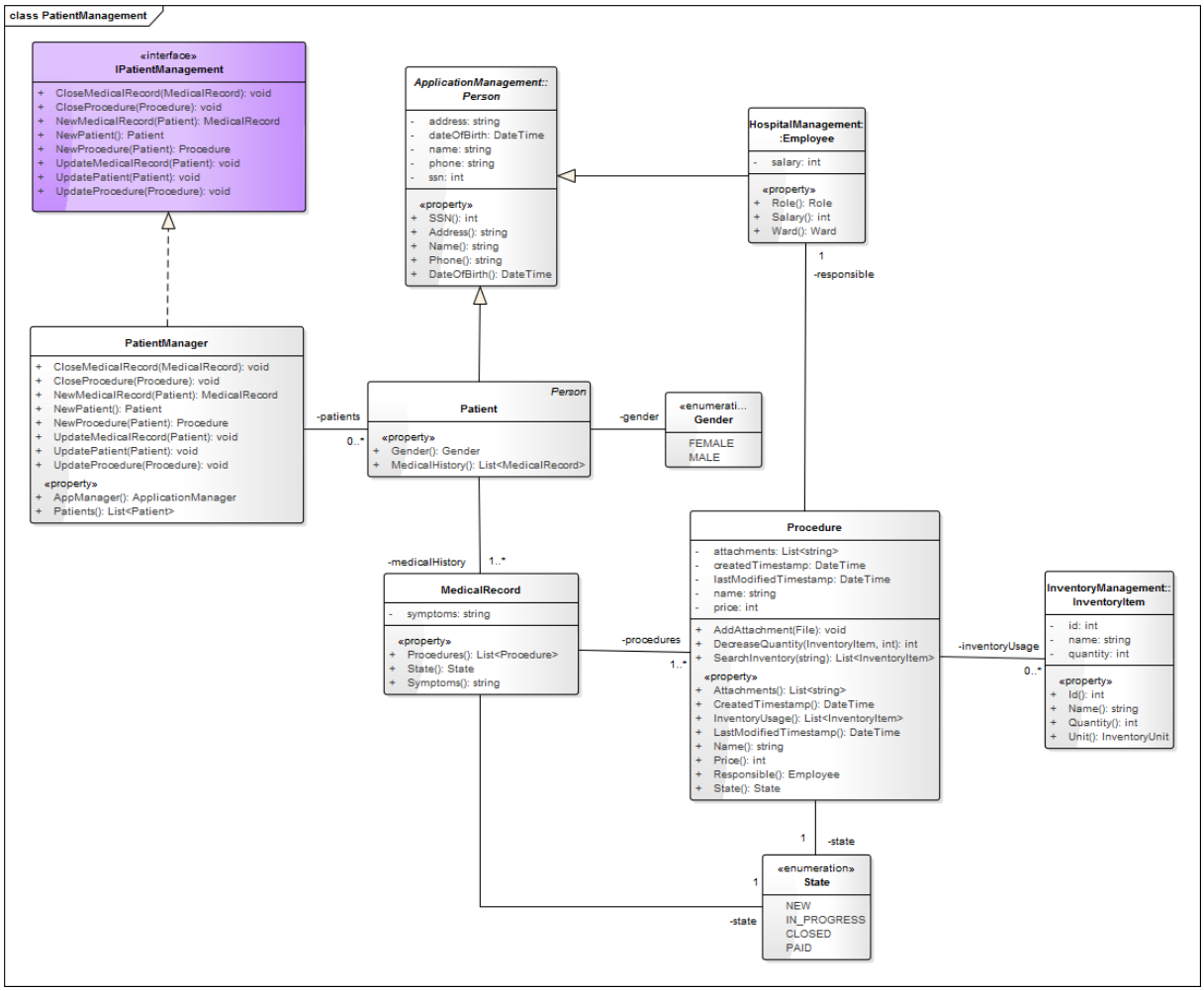
19. ábra- UpdateEmployee() metódus hívása kezelőfelületről



20. ábra - RemoveEmployee() metódus hívása kezelőfelületről

PatientManagement alrendszer

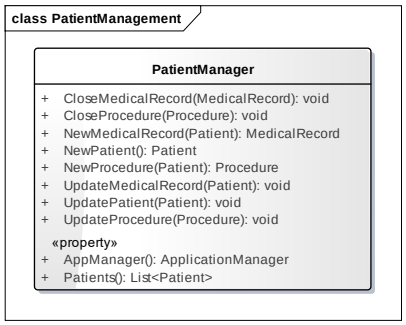
A kórház betegeinek adminisztrálására, regisztrálására, felvételére szolgáló modul, itt történik a kezelések betegekhez rendelése is. A betegek keresésére is mód van, illetve a számla összegének, tételes listájának előállítására is itt van kialakítva a funkcionalitás.



21. ábra - Kórház adminisztráció alrendszer osztályai

PatientManager osztály

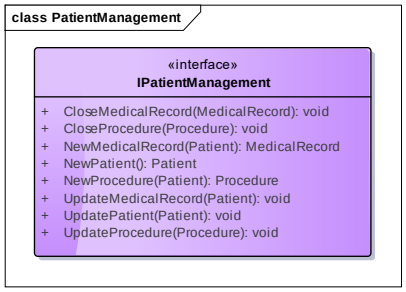
Az alrendszer aktív osztálya, az alkalmazás futása során a HospitalManager osztályhoz hasonlóan ebből is egyetlen példány készül, és végigköveti az alkalmazás élettartamát.



22. ábra – PatientManager osztály

A PatientRegistryWindow ablakban jelenik meg a betegek listája, a PatientManager osztály a listához hozzáadás tényleges megtörténte előtt ellenőrzi, hogy az adott TAJ (SSN) számmal szerepel-e már valaki az adatbázisban, illetve ez alapján lehet célzottan is keresni, amennyiben a beteg már fel van véve kezelésre. A betegekhez kezelést lehet (adott esetben kell) hozzáadni, ennek oka a beteg osztálynál kerül kifejtésre. A kezelések kórtörténetet alkotnak (ez automatikusan létrejön az első kezeléssel), és eljárásokból állnak. Az első eljárás a beteg felvétele, illetve kivizsgálása. Jogosultságtól függően az eljárások eredményét be lehet jegyezni, illetve az eljárásokat, kórtörténetet frissíteni lehet. A beteg részére a számlakiállítást és a zárójelentés elkészítését is ez az osztály végzi a MedicalRecord lezárásakor.

Ezen metódusok meghívását a PatientManager által implementált interfészen keresztül lehet megtenni:



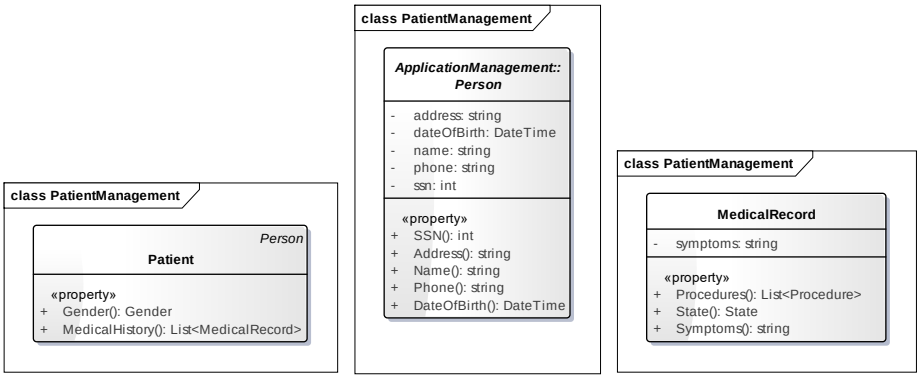
23. ábra – PatientManager osztály implementált interfészei

Ezeken kívül (mint később látható lesz) az adattagok beépített metódusait alkalmaztuk, illetve az adatkötés lehetőségeivel éltünk.

Patient osztály (Person osztály), MedicalRecord osztály

A Patient osztály a betegek adatainak tárolására szolgál. Ősosztálya a Person osztály, amely egyben a dolgozóknak is ősosztálya (lásd még HospitalManager osztály). A Patient osztálynak létrehozásakor van egy MedicalHistory nevű List<MedicalRecord> adattagja, egy anamnézis, mely kezdetben rövid ideig üres. A beteg felvételekor a létrejön az első MedicalRecord, benne egy Procedure (pl. Orvosi kivizsgálás, konzultáció) objektummal (bővebben lásd lejjebb). A MedicalRecord objektumok tehát a

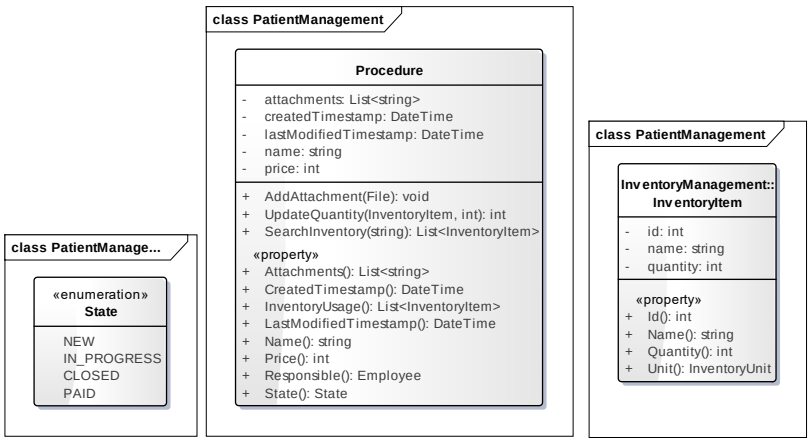
Procedure objektumok olyan konténerei, amelyek egy egybefüggő tünetegyüttes egyhuzamban történő gyógykezelését igyekeznek lefedni, melyben az első mozzanat az orvosi kivizsgálás és kezdeti diagnózis, az utolsó mozzanat pedig az utolsó kezelés, amely után a beteg (remélhetőleg) gyógyultan távozik a kórházból, ennek megfelelő diagnózissal.



24. ábra – Patient (Person) osztály, MedicalRecord osztály

Procedure osztály

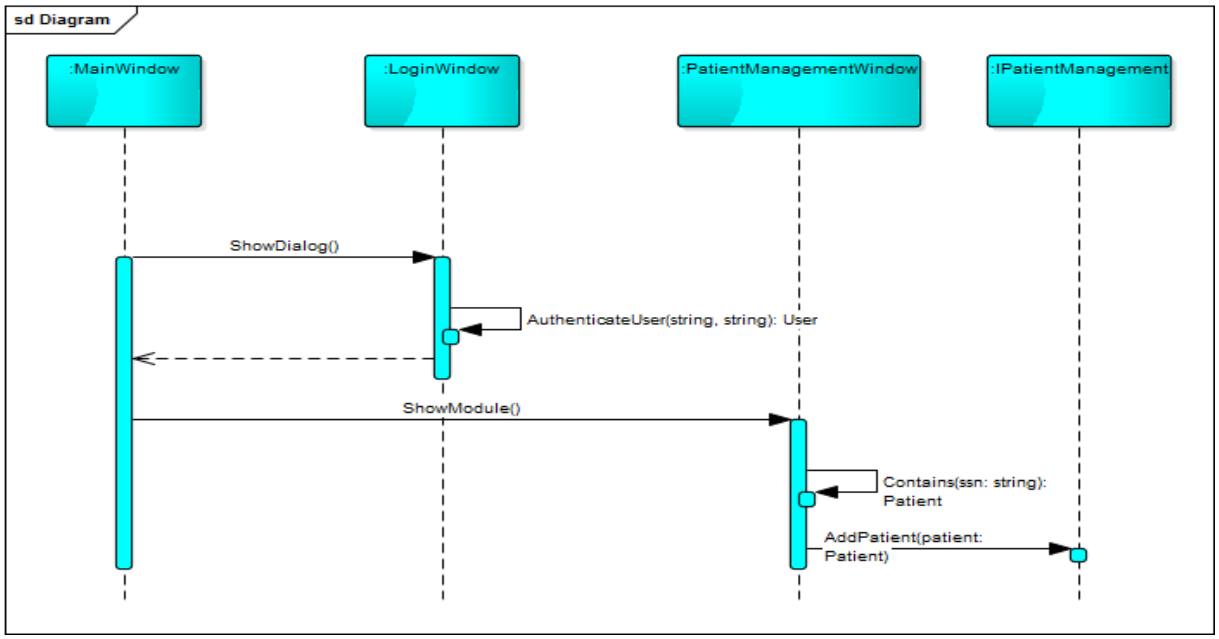
A MedicalRecord osztályoknak kötelezően van egy List<Procedures> adattagja, amelynek első eleme általában a kezdeti orvosi vizsgálat, vagy konzultáció. Minden Procedure-höz tartozik egy orvos, és a konvencionális adattagokon kívül csatolmányokat is tartalmazhat, pl. Röntgen, CT vizsgálat képei. Minden Procedure egyedi azonosítóval rendelkezik, amely a kiírás pillanatában hozzá rendelt TimeStamp. Az IInventoryManagement interfészen keresztül emellett rendelkezik olyan metódusokkal, amelyek a felszerelés-szükségletnek (pl. szike, gézlapok) megfelelő InventoryItem-ek számát vátoztatják, és segítik azok keresését az eszközadatbázisban (lásd InventoryManagement modul).



25. ábra – Patient (Person) osztály, MedicalRecord osztály

A Procedure és a MedicalRecord osztály ezen kívül egy állapotváltozóval rendelkezik, amely a számla kiállítása, és a kezelések nyomon követése során játszik szerepet.

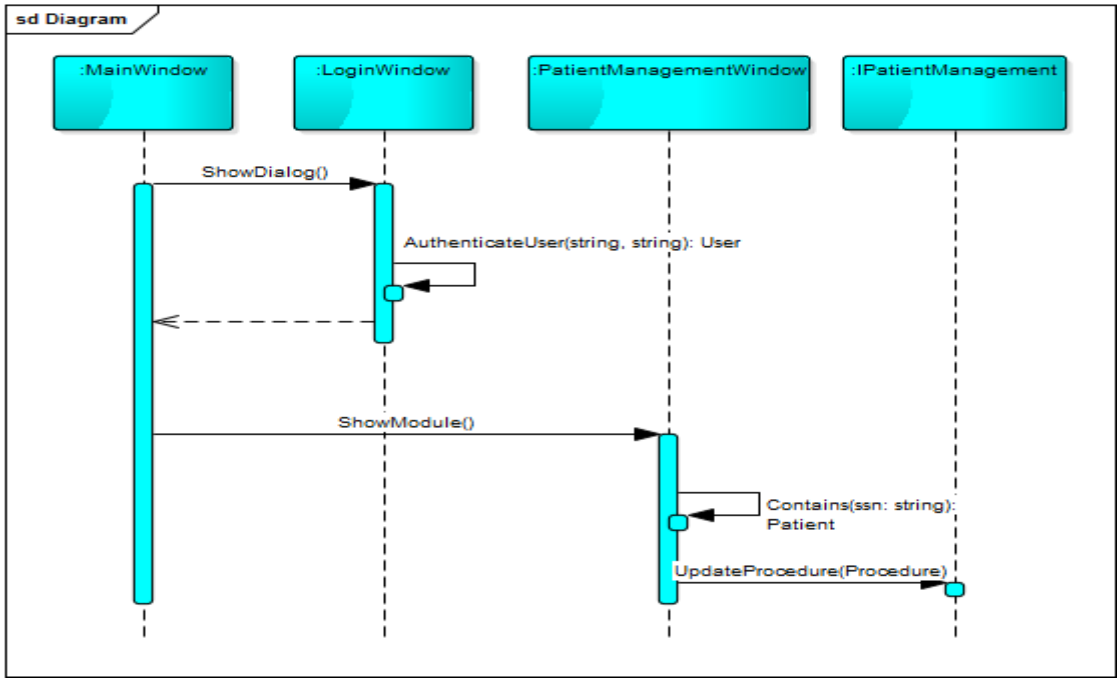
Betegfelvétel



26. ábra – AddPatient() metódus

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A beteg felvételekor annak TAJ-száma alapján jelzi a rendszer, ha a betegek listájában már megtalálható a szám, így nem hoz létre új beteg objektumot, hanem a meglevőhöz lehet hozzáférni (jelen esetben ténylegesen létrejön új felhasználó).

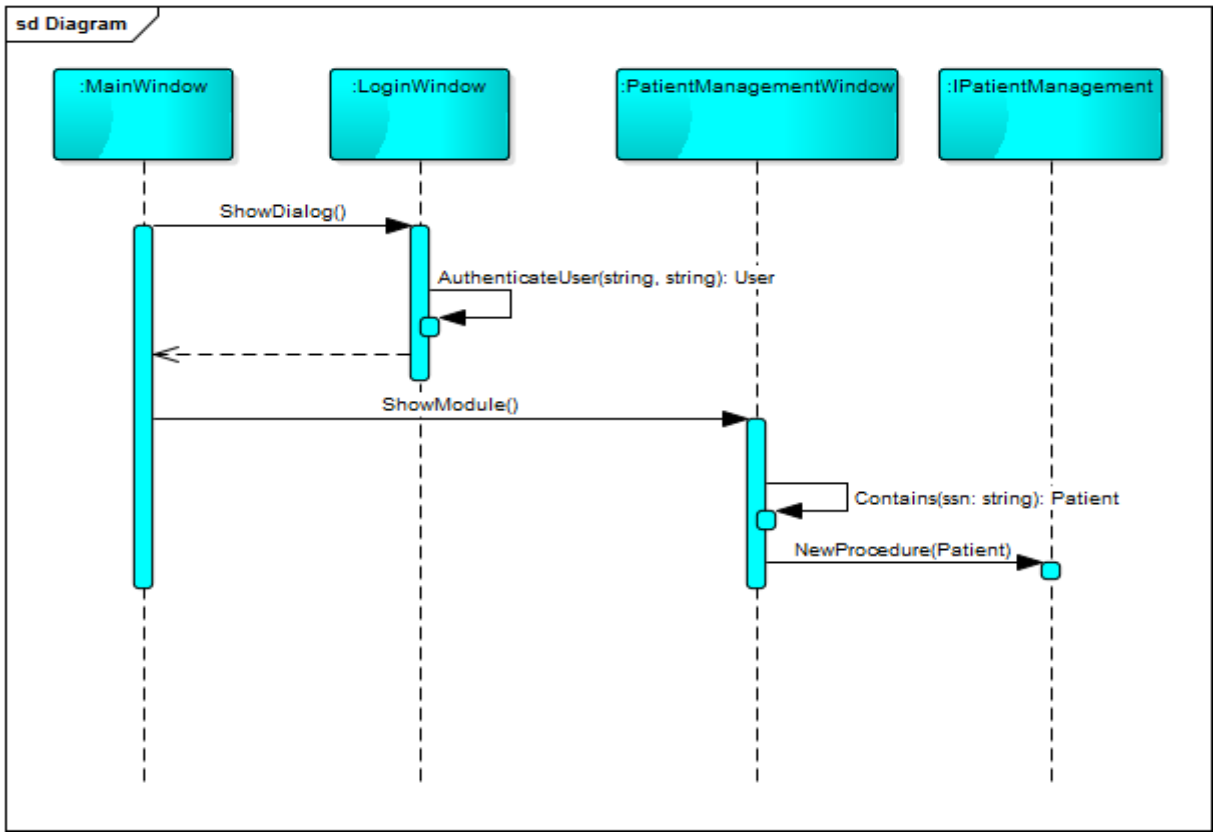
Egészségügyi eljárás eredményének rögzítése



27. ábra UpdateProcedure() metódus

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A TAJ szám alapján keresve a beteg lezáratlan MedicalRecord-jában, megkeresve az adott eljárást frissíteni lehet pl. annak diagnózisát.

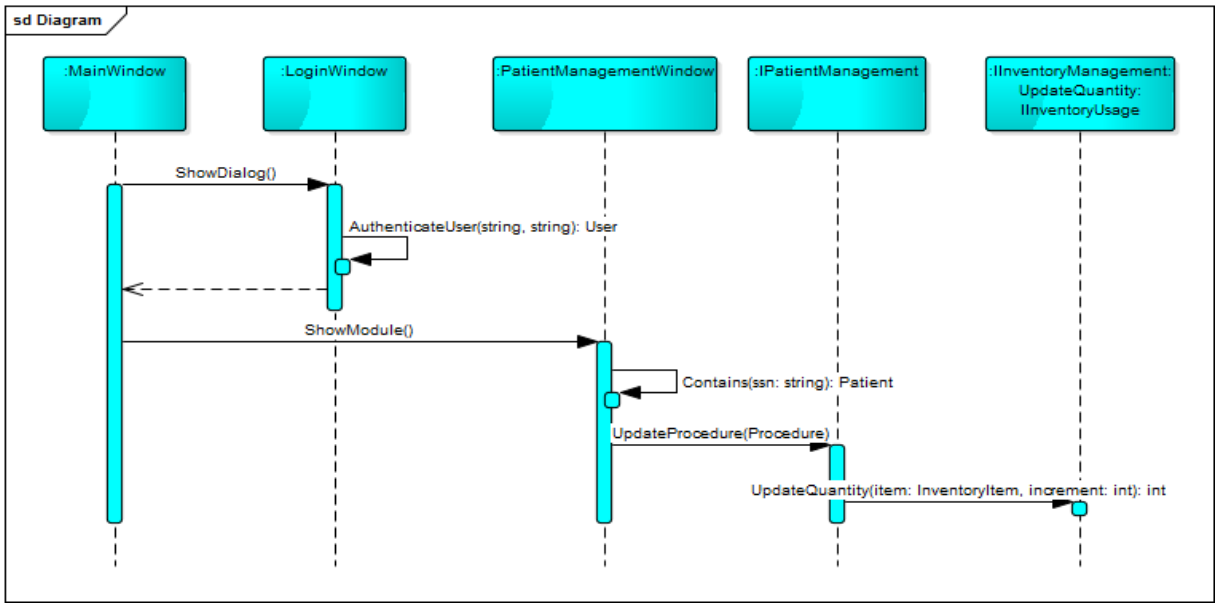
Egészségügyi eljárás kiírása



28. ábra - NewProcedure() metódus

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A beteg felvételekor annak TAJ-száma alapján jelzi a rendszer, ha a betegek listájában már megtalálható a szám, így a meglevőhöz hozzá lehet rendelni egy új eljárást.

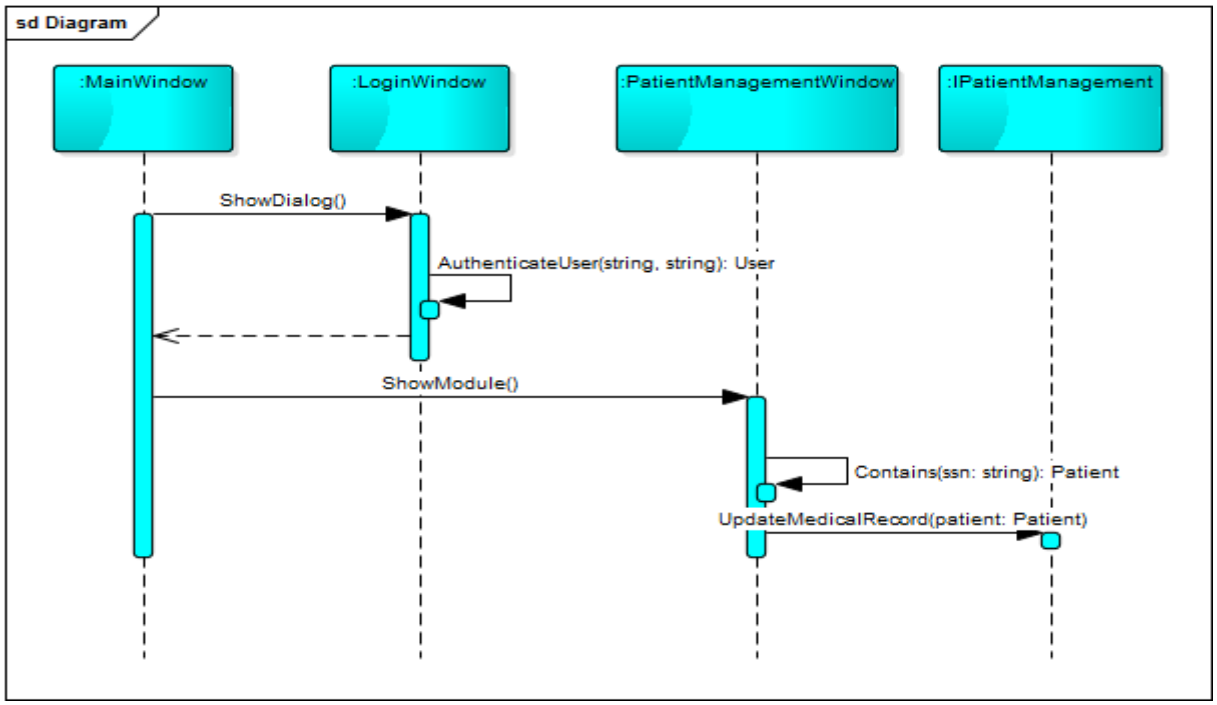
Eszköz törlése



29. ábra - UpdateQuantity

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A beteg TAJ-száma alapján, a beteg vizsgálatainak listájában már az eljárás eszközigényét az ItemManager által implementált interfészen keresztül meg lehet változtatni.

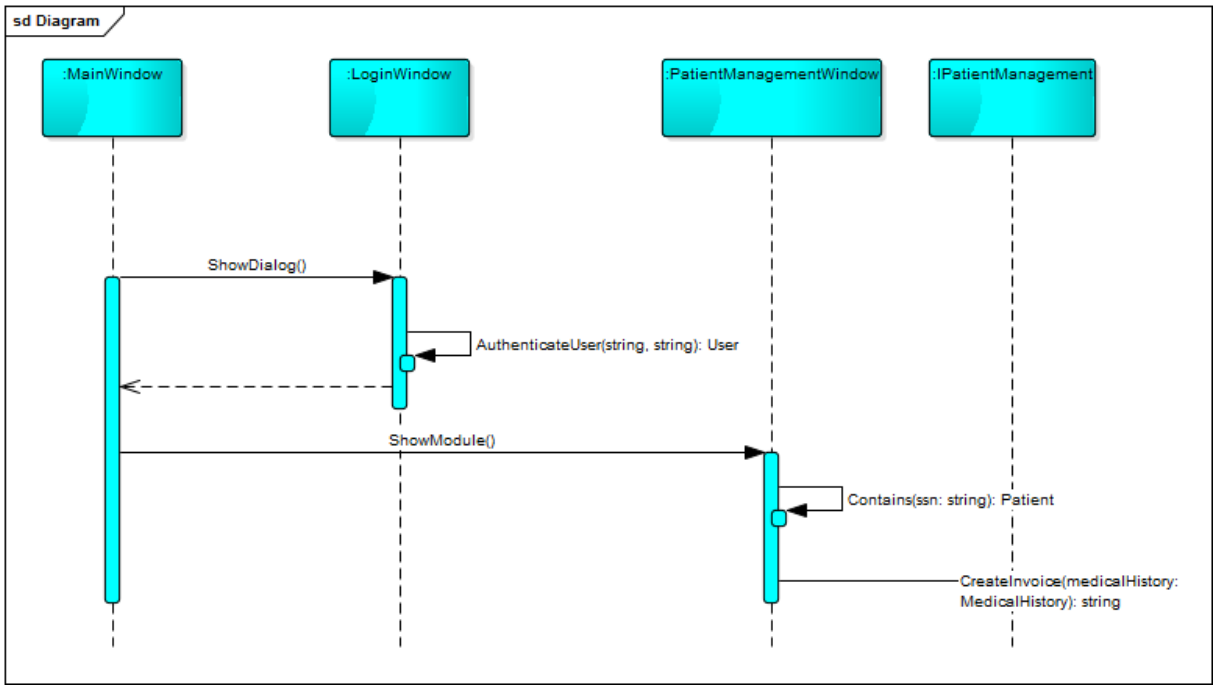
Kórtörténet szerkesztése



30. ábra - UpdateMedicalRecord() metódus

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A beteg felvételekor annak TAJ-száma alapján jelzi a rendszer, ha a betegek listájában már megtalálható a szám, a meglevő objektum kórtörténetét szerkeszteni lehet.

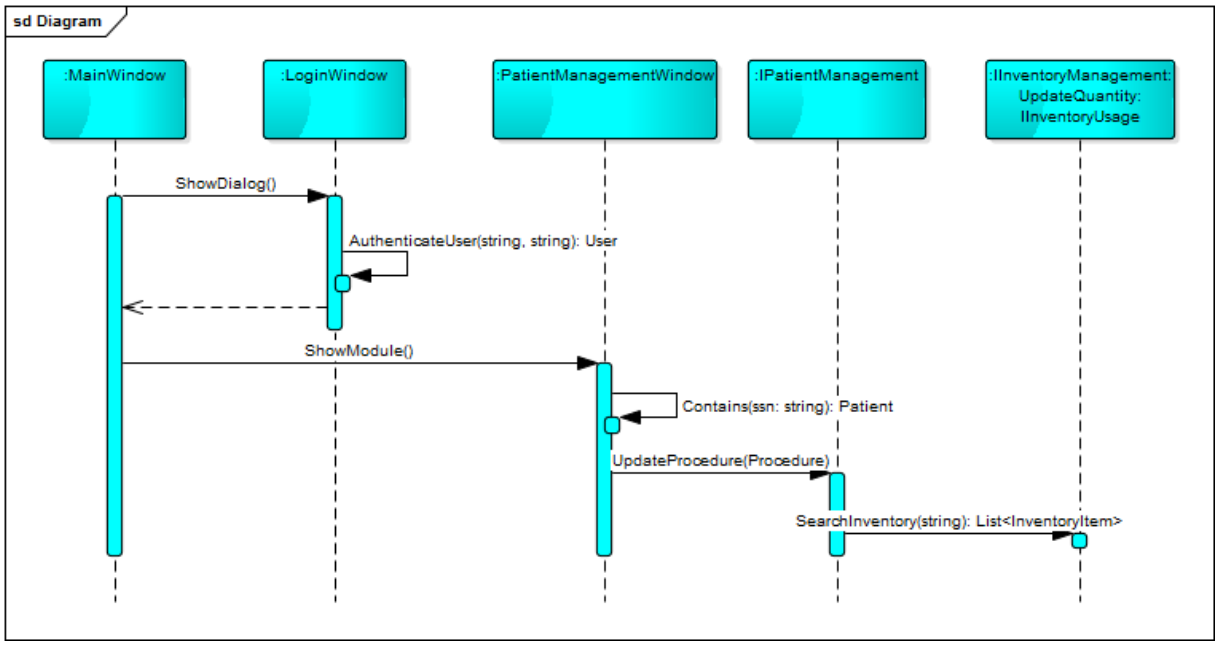
Számla kiadása



31. ábra - CreateInvoice() metódus

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A TAJ-szám alapján megtalált beteg nyitott MedicalHistory objektumában a még nem kifizetett eljárásokról tételes listát készít.

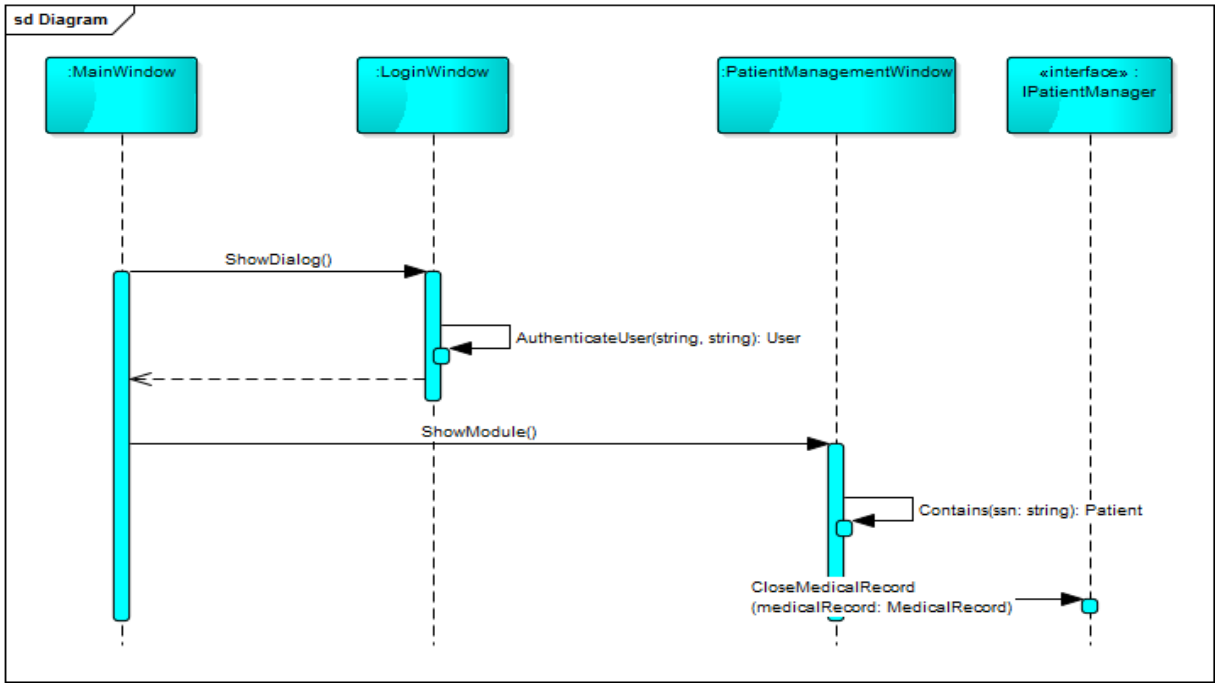
Interface: IInventoryManagement



32. ábra - IInventoryManagement () metódus

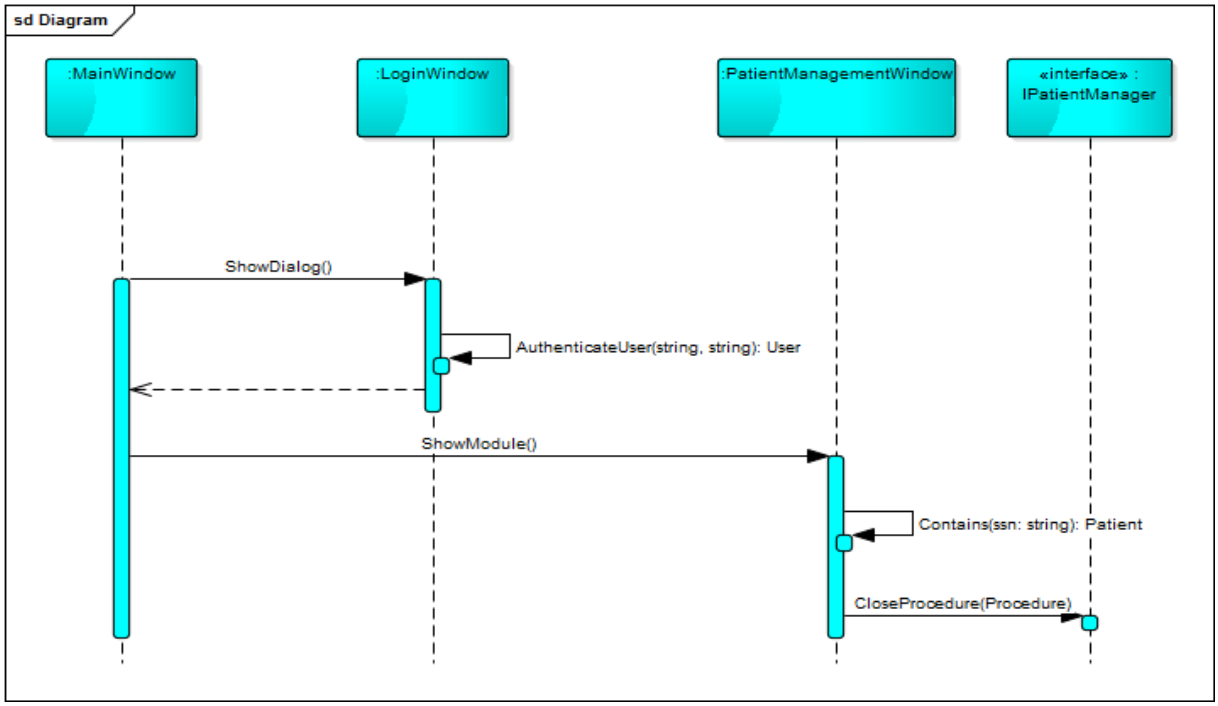
Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A TAJ-szám alapján megtalált beteg nyitott MedicalHistory objektumához hozzárendelt orvosi elárások számára szükséges eszközöket meg lehet keresni az IInventoryManagement interfész SearchInventory metódusának segítségével.

Interface: IPatientManager



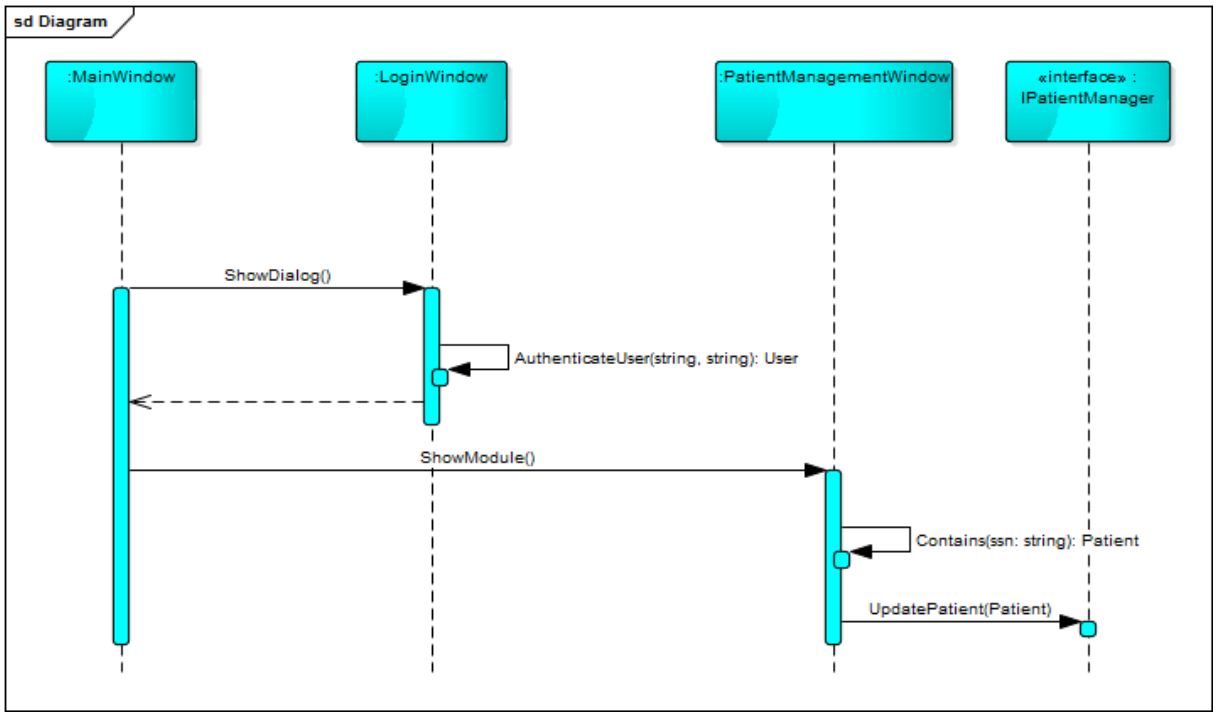
33. ábra - CloseMedicalRecord() metódus

Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A TAJ-szám alapján megtalált beteg nyitott MedicalHistory objektumát – amennyiben például a gyógykezelés véget ért – le lehet zárni.



34. ábra - CloseProcedure() metódus

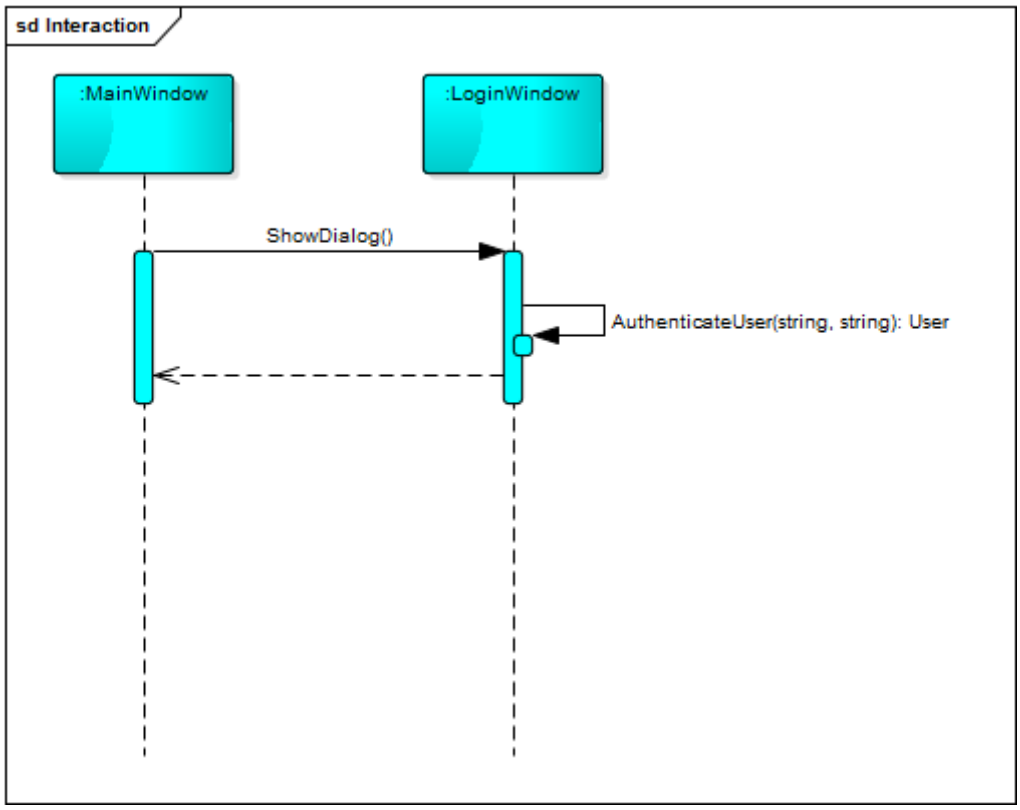
Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A TAJ-szám alapján megtalált beteg nyitott MedicalHistory gyógykezelés objektumában a lefolytatottnak tekintett eljárást le lehet zárni, például a diagnózis felállítása után.



35. ábra - UpdatePatient() metódus

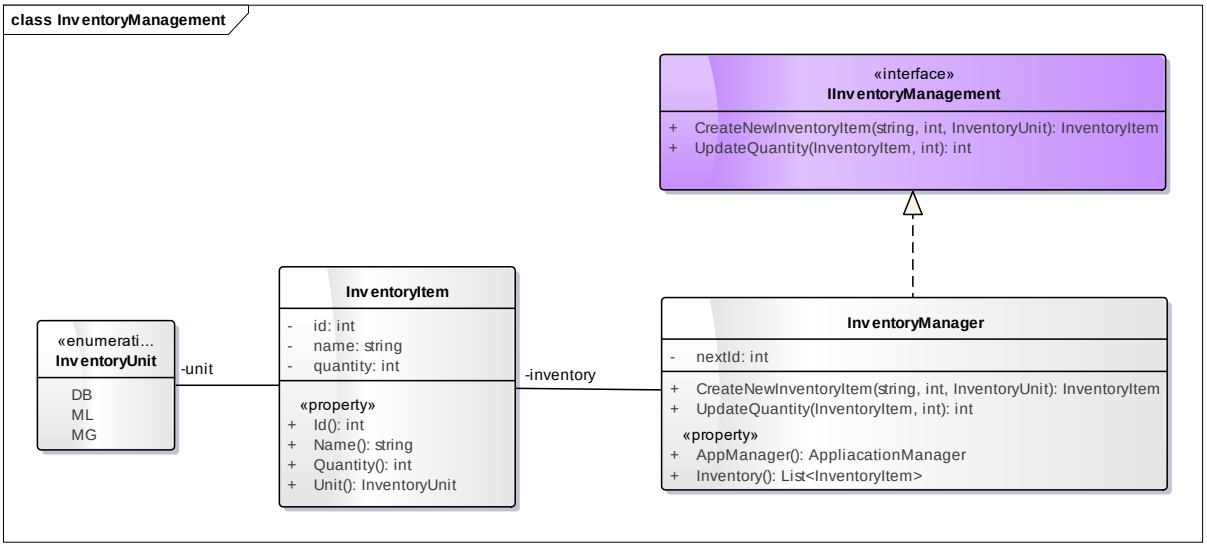
Sikeres azonosítás után a felhasználó visszakerül a főablak felületére, ahonnan a betegek kezelését választja. A TAJ-szám alapján megtalált beteg adatait azok valamilyen módosítása után el lehet menteni, és felülírni az adatbázisban tárolt adatokat az adatkötés által felülírt objektum adataival.

Normálistól *eltérő használati esetek*



36. ábra – Sikertelen azonosítás

InventoryManagement alrendszer



37. ábra InventoryManagement alrendszer

IInventoryManager interfész

Ez az interfész biztosítja a kapcsolatot a megjelenítési réteg és az alkalmazásréteg eszközkezelő logikája között. Két metódust definiál, melyek az InventoryManager osztályban vannak implementálva. Ezek felelnek új eszközbejegyzés létrehozásáért (CreateNewInventoryItem()) és egy eszköz szabad mennyiségének változtatásáért (UpdateQuantity()).

InventoryManager osztály

Az alrendszer aktív osztálya, amely az alkalmazás futása során egyetlen példányban létezik és végigköveti annak élettartamát.

Az osztály adattagjai mind privát elérésűek melyek publikus tulajdonságokon keresztül érhetőek el. Az AppManager tulajdonságon keresztül érhető el az alkalmazás objektum. Az Inventory tulajdonságon keresztül manipulálható az eszköznyilvántartás, amely InventoryItem objektumok listája. A nextId egész típusú mező a következő szabad sorszámot tárolja, amit egy InventoryItem objektum egyedi azonosítója felvehet.

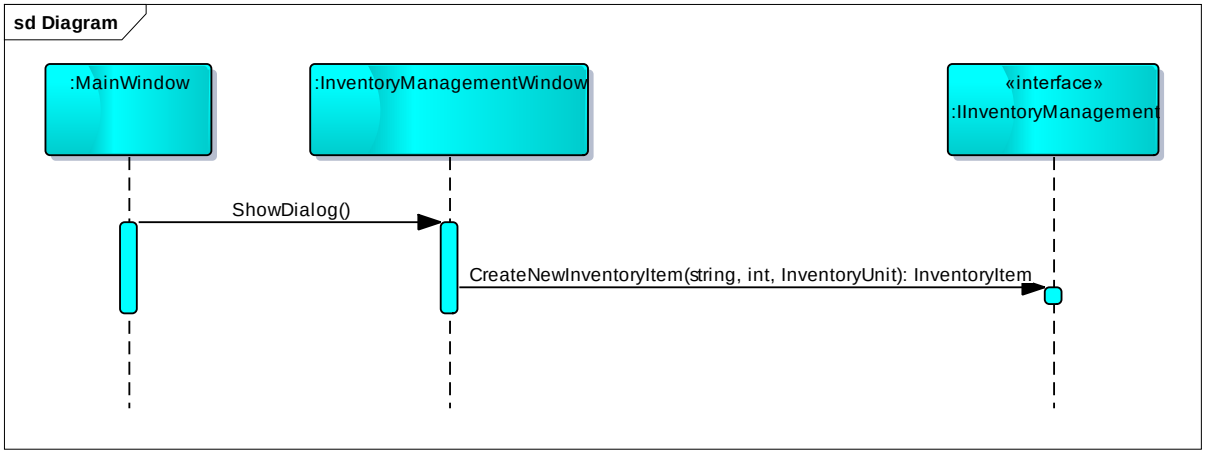
Az osztály implementálja az IInventoryManager interfészben definiált metódusokat.

InventoryItem osztály

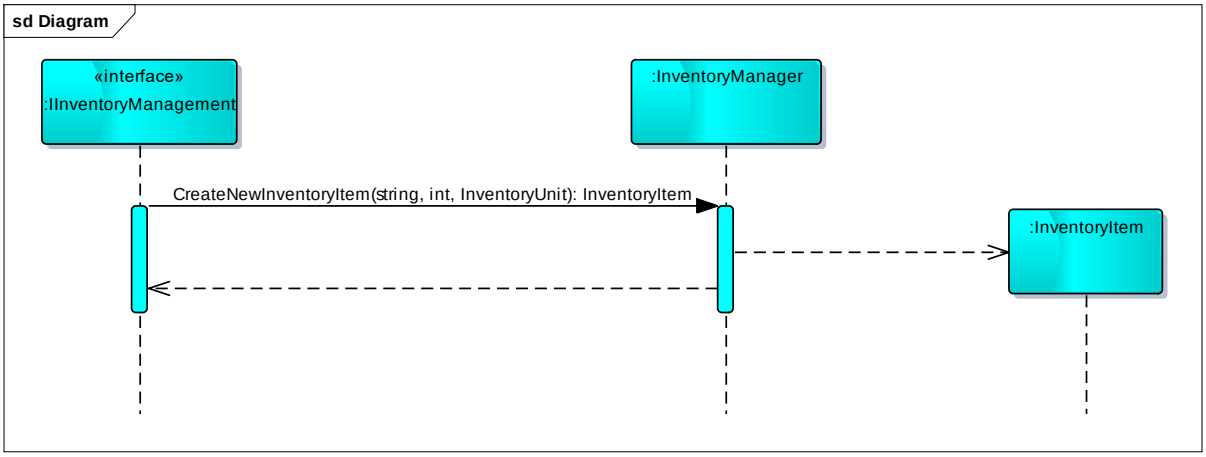
Ez az osztály reprezentálja az eszköznyilvántartás egy bejegyzését. Minden bejegyzés egész típusú egyedi azonosítóval (id), string típusú névvel (name), egész típusú mennyiséggel (quantity) és felsorolás típusú mennyiségjelzővel (unit) rendelkezik. Az adattagok privát elérésűek, tulajdonságokon keresztül manipulálhatóak.

InventoryType enumeráció

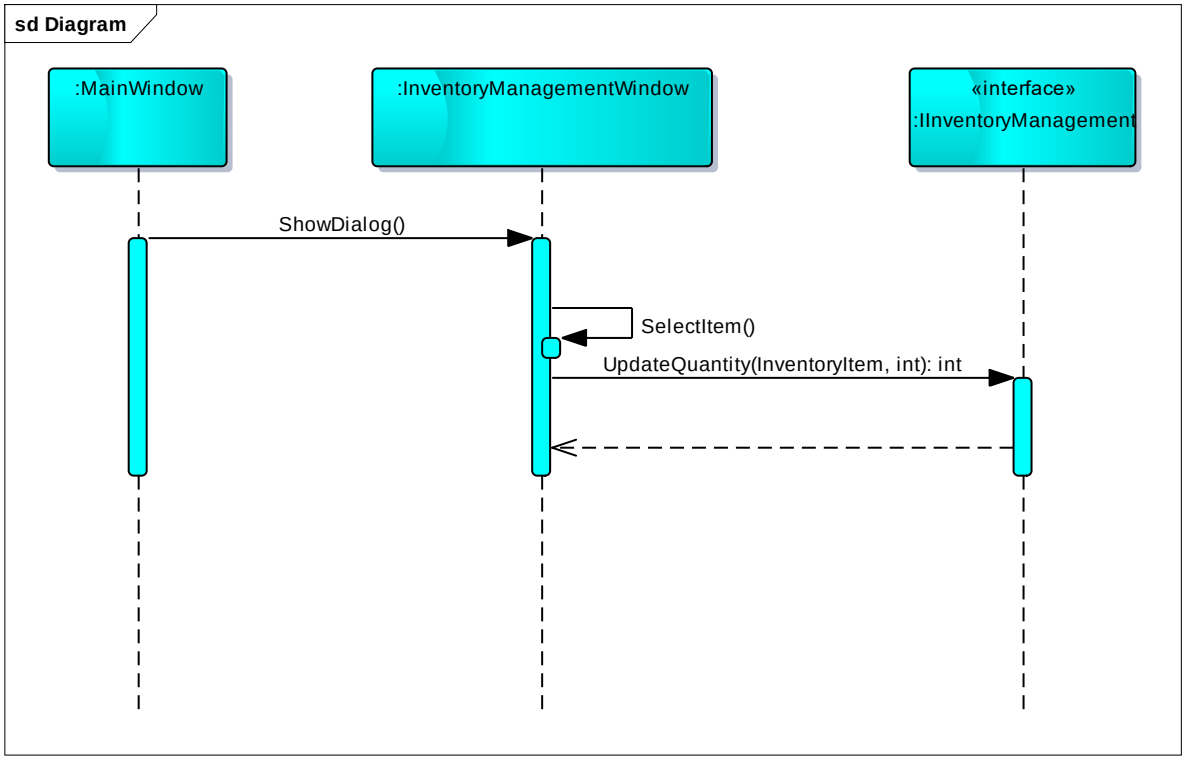
Ez a felsorolás típus a lehetséges mennyiségjelzőket reprezentálja.



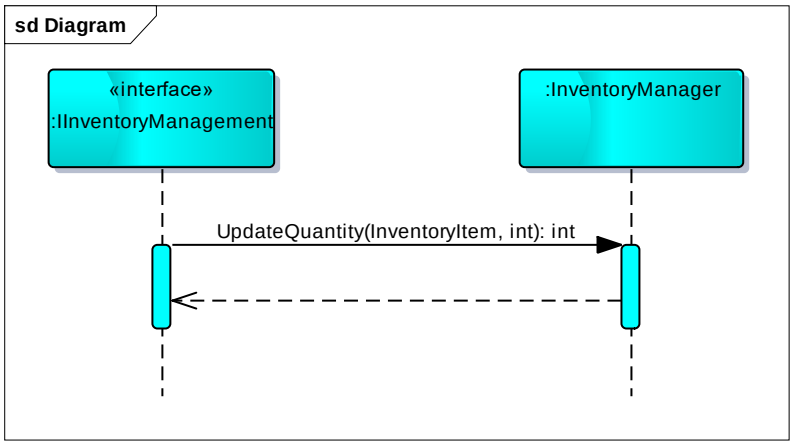
38. ábra CreateNewInventory() metódus használati eset realizáció



39. ábra CreateNewInventory() metódus interfész művelet

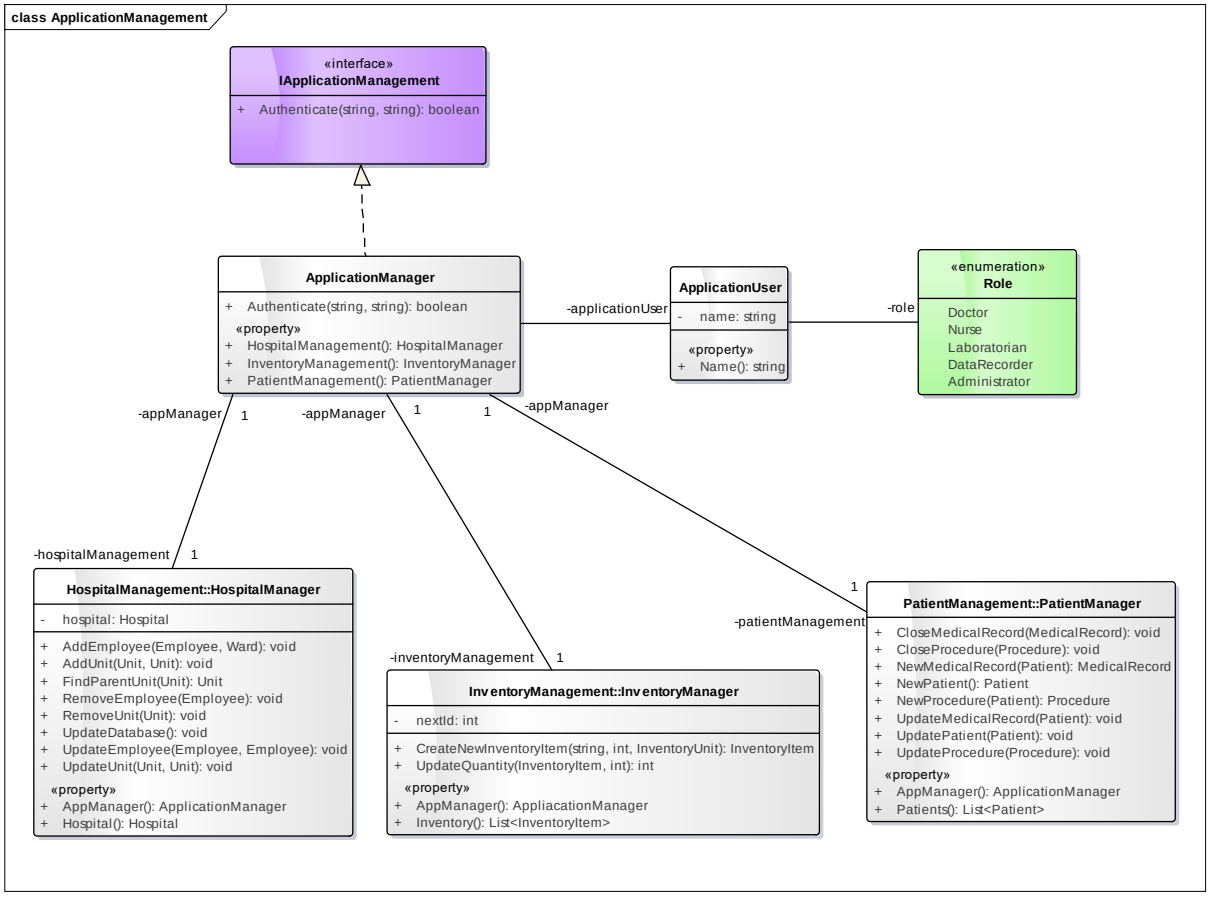


40. ábra UpdateQuantity() metódus használati eset realizáció



41. ábra UpdateQuantity() metódus interfész művelet

ApplicationManagement alrendszer



42. ábra ApplicationManagement alrendszer

IApplicationManagement interfész

Ez az interfész biztosítja a kapcsolatot a megjelenítési réteg és az alkalmazásréteg alkalmazás logikája között. Egyetlen metódust definiál amely a felhasználó autentikációjáért felel (Authenticate())

ApplicationManager osztály

Az alrendszer aktív osztálya, mely az alkalmazás futása során egyetlen példányban létezik és végigköveti annak élettartamát.

Az osztály referenciákat tartalmaz a rendszer alrendszereit megvalósító osztályok 1-1 példányára, ezek privát adattagok, melyek tulajdonságokon keresztül érhetők el.

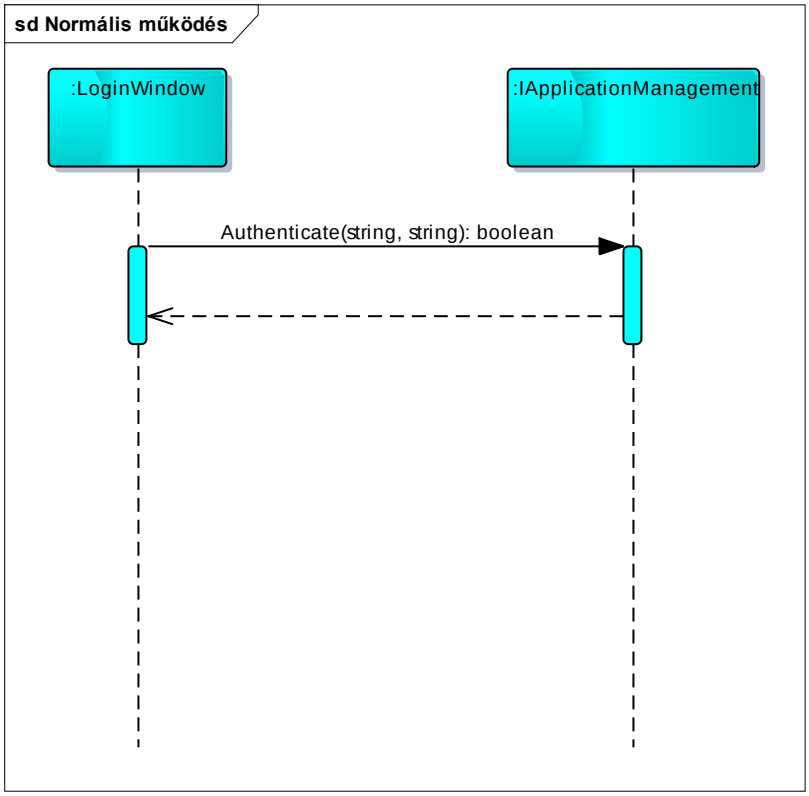
Egyetlen metódusa az IApplicationManager interdészben definiált metódust implementálja.

ApplicationUser osztály

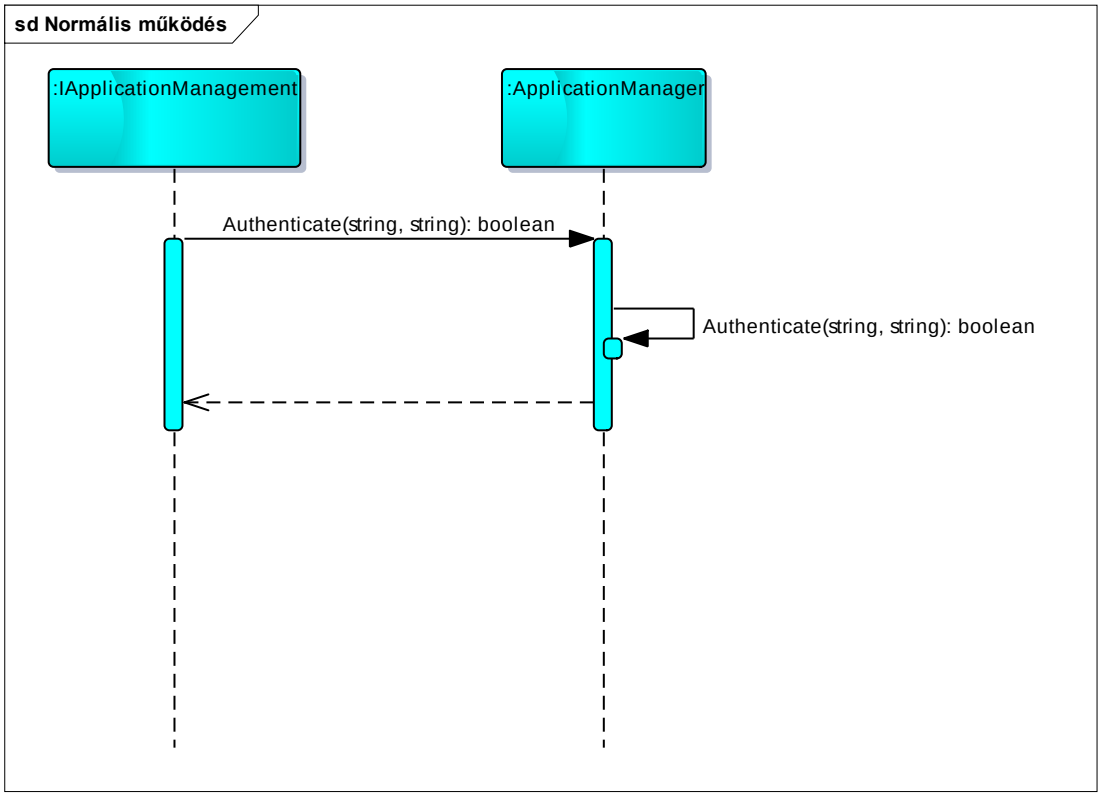
Az alkalmazás felhasználóját reprezentálja annak nevét és szerepét tárolja, előbbit string, utóbbit felsorolás típusú privát változóban.

Role enumeráció

A rendszer felhasználóinak szerepköreit tárolja.



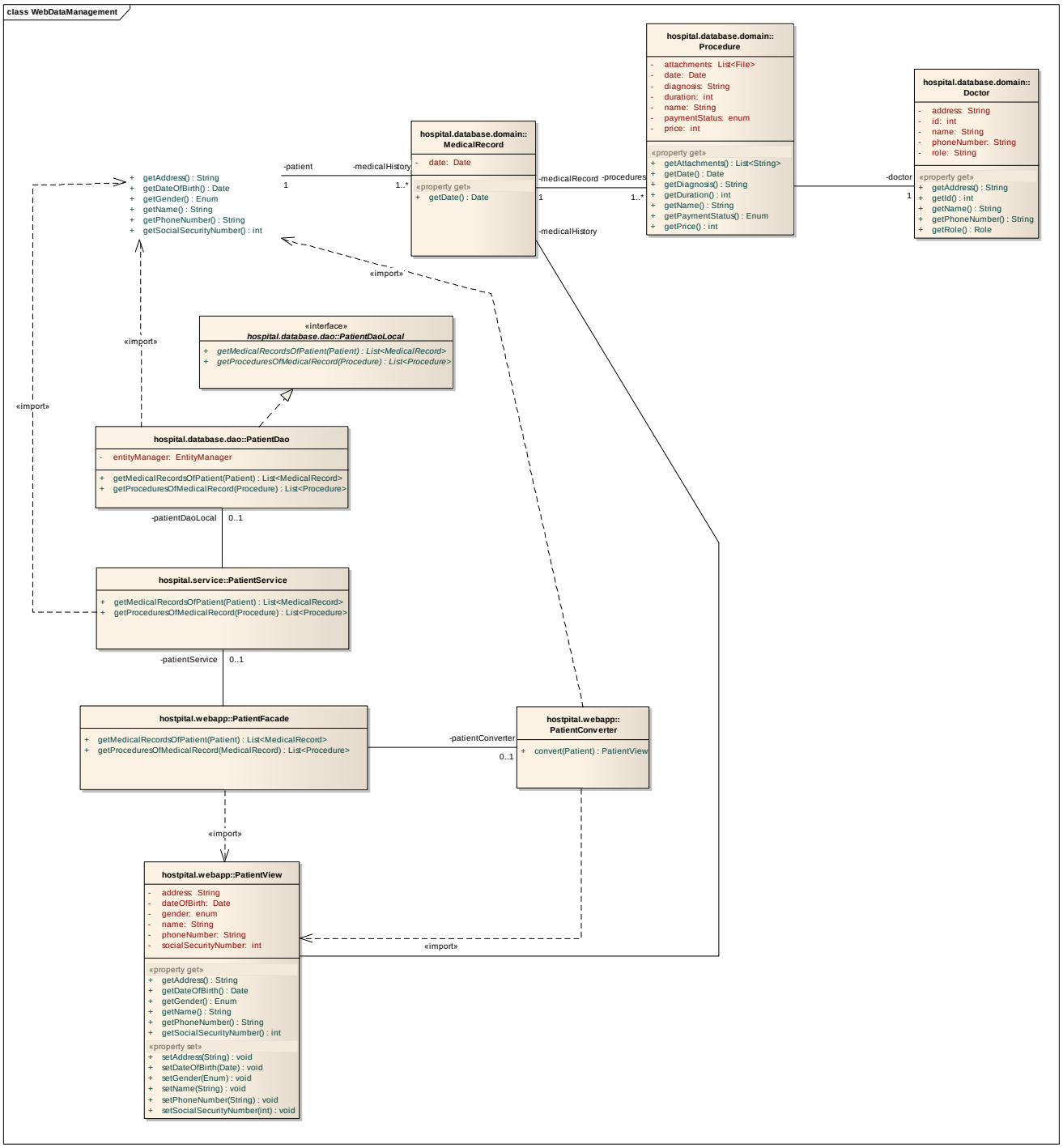
43. ábra Authenticate metódus használati eset realizáció



44. ábra Authenticate metódus interfész művelet

WebDataManagement alrendszer

WebDataManagement



45. ábra WebDataManagement alrendszer

A web alkalmazás alapvetően 3 rétegre oszlik, legalul helyezkedik el az egyetlen – a funkciók száma miatt nem indokolt tovább bontani – *DataAccessObject* (**PatientDao**) osztály, mely a

JDBC-re épülő *Java Persistence Api*n keresztül *Entity* osztályok (**Patient**, **MedicalRecord**, **Procedure**, **Doctor**) felhasználásával a lekért tartalmat betölti.

Ezt továbbítja a Service rétegnek, jelen esetben **PatientService**nek, amire tovább hív, a **PatientFacade** osztály, amely gyakorlatilag a kinyert adatok megjelenítésre alkalmas formába konvertálja a **PatientView** és a **PatientConverter** osztályok segítségével.

A példányosítást, szálkezelést stb. teljes egészében a keretrendszerre bízzuk, így a fenti adatbázis entitás osztályokat leszámítva az összes többi állapot nélküli (@Stateless) *EJB* osztály.

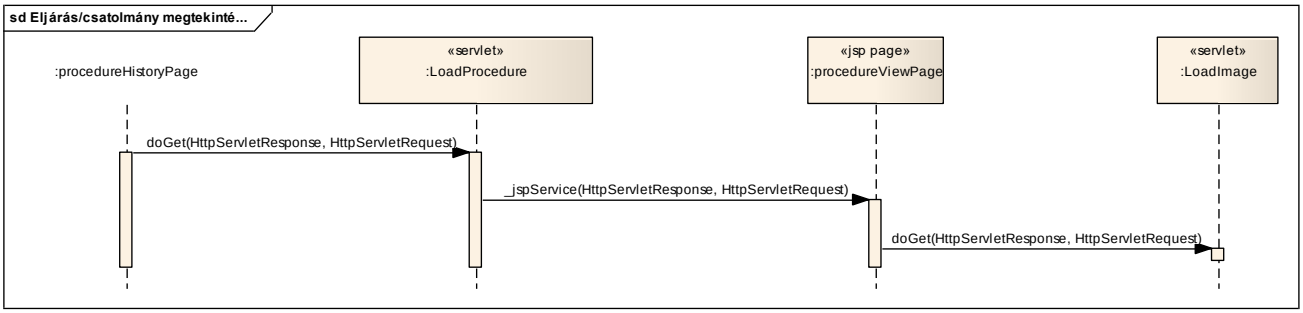
Web felület

46. ábra Web felület

A webfelületet *jsp* oldalak és *servletek* segítségével valósul meg.

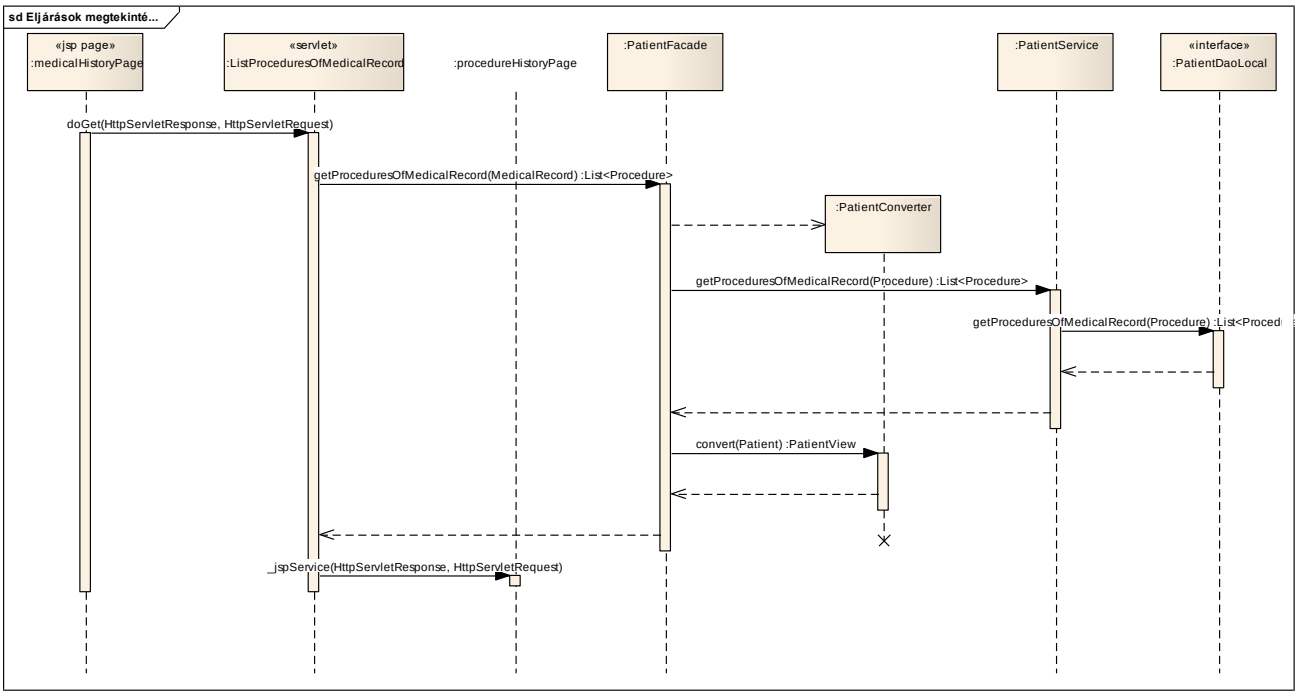
Első körben a felhasználó belép a rendszerbe *FORM* alapon (**Login**, **loginPage**), megadja a belépési kódját, ami jelen esetben a TAJ száma és a hozzá tartozó jelszavát, ami a születési dátum. Autentikációért a *j_security_check* felel, leellenőrizzük, hogy a felhasználó létezik-e, amennyiben igen és a megadott belépési adatok is helyesek továbbítjuk a Kórtörténeti oldalra (**ListMedicalHistory** , **medicalHistoryPage**), ahol kilistázásra kerül az összes eddigi kezelése.

Egy adott kezelésre kattintva megjelennek a kezeléshez tartozó eljárások (**ListProceduresOfMedicalRecord, procedureHistoryPage**), majd egy kezelést kiválasztva (**LoadProcedure, procedureViewPage**) megtekinthetjük annak adatait és a hozzá tartozó csatolmányokat (**LoadImage).**



47. ábra Eljárás/csatolmány megtekintése

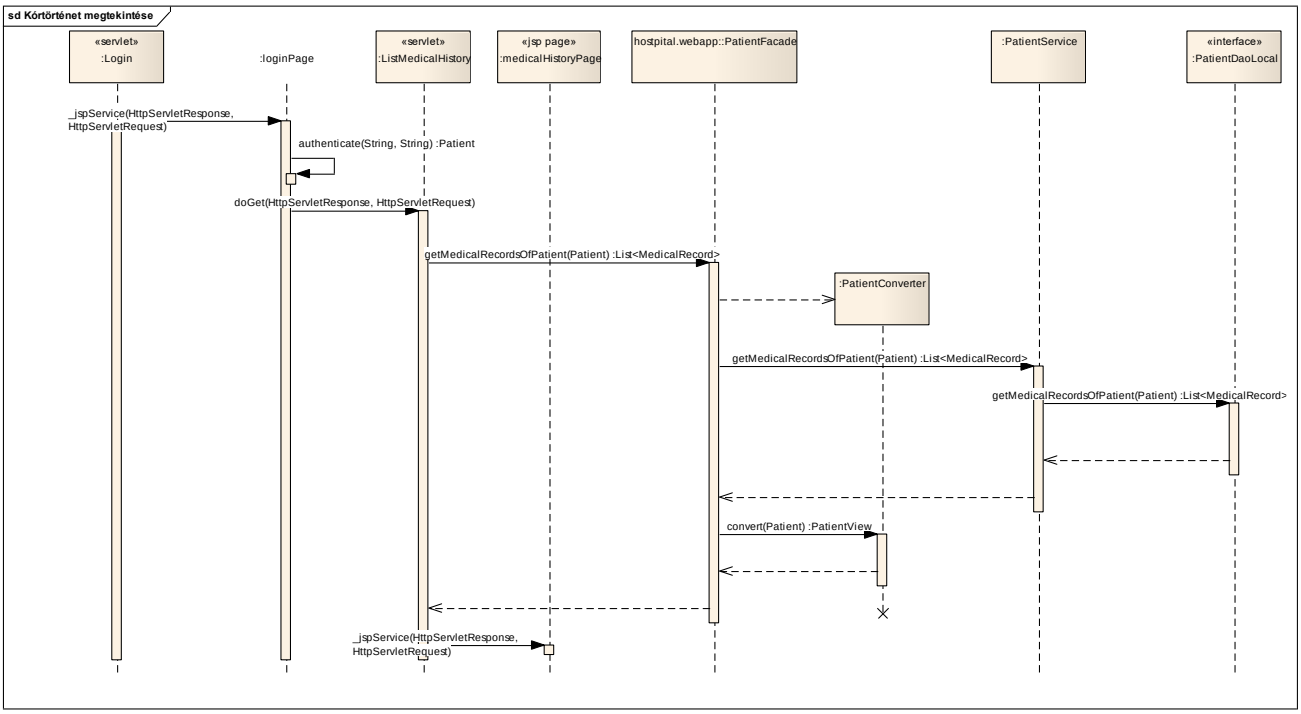
A felhasználó (páciens) megtekinti az adott eljáráshoz tartozó adatokat (**procedureViewPage)**, mely a **LoadProcedure** servlet által betöltésre kerülnek. A csatolmányok nevére kattintva megjelenik azok tartalma (**LoadImage** servlet).



48. ábra Eljárások megtekintése

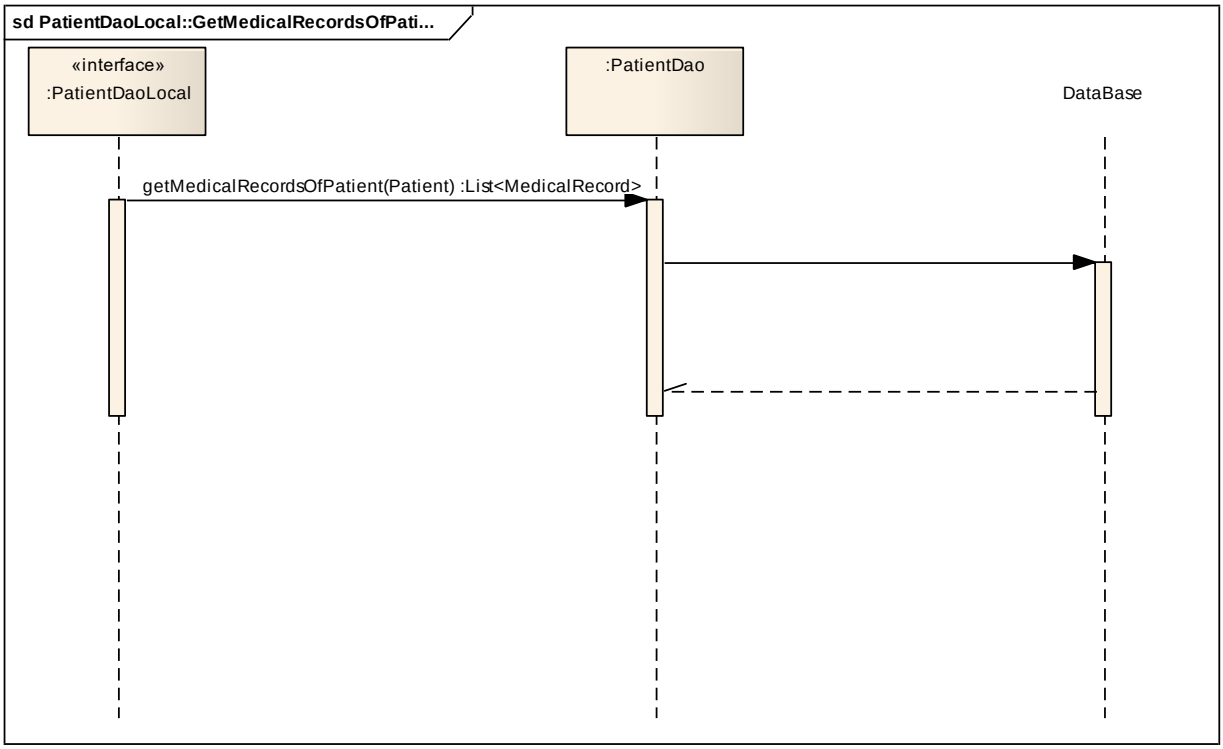
A felhasználó (páciens) kiválaszt egy kezelést, hogy megtekintse az ahhoz tartozó eljárásokat. A **ListProceduresOfMedicalRecord** servlet lekéri a **PatientFacade** osztálytól az adatokat, ezen kérés tovább gyűrűzik a **PatientService** osztályon keresztül a lokális **PatientDaoLocal** interfészig.

Miután a kívánt adat előállt, az megjelenítésre kerül a **procedureHistoryPage**-en.



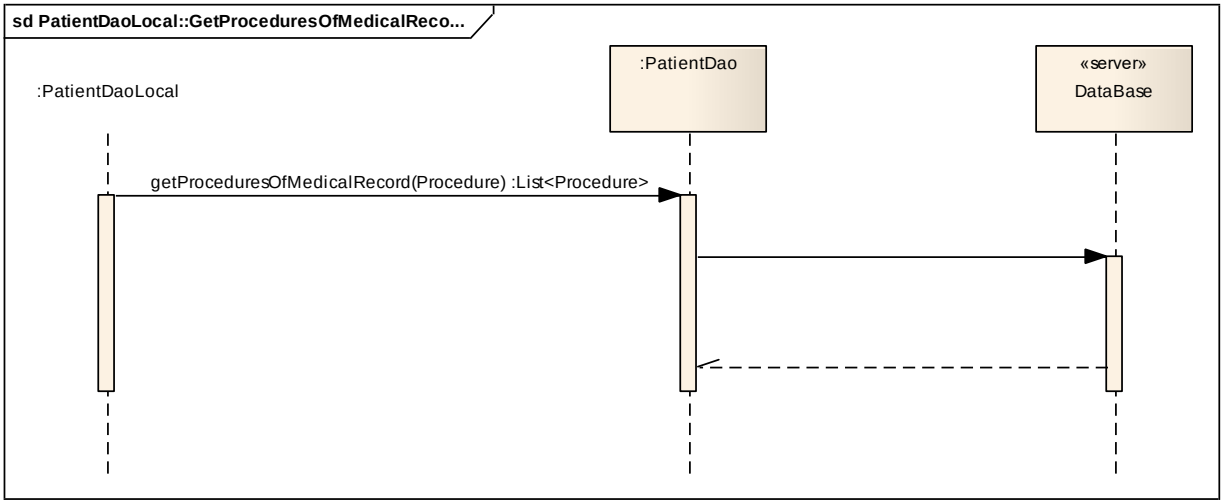
49. ábra Kórtörténet megtekintése

A felhasználó (paciens) bejelentekezik a **loginPage** oldalon. Autentikáció után a rendszer átirányítja a **medicalHistoryPage**-re**,** miután az adatbázisból lekért minden szükséges adatot a **PatientFacade**/**PatientService**/**PatientDaoLocal** osztályokon, illetve interfacen keresztül. Végül a **PatientCoverter** osztály átkonvertálja **PatientView**-ra, azaz a felhasználó számára megjeleníthető formába önti.



50. ábra PatientDaoLocal::GetMedicalRecordsOfPatient

PatientDaoLocal interfész definiálja az összes szükséges metódust, amire a **PatientDao** osztálynak szüksége van. Utóbbi a keretrendszer segítségével lekéri (**getMedicalRecordsOfPatient)** a kívánt adatokat az adatbázisból – a felhasználóhoz (páciens) tartozó összes kezelést –.

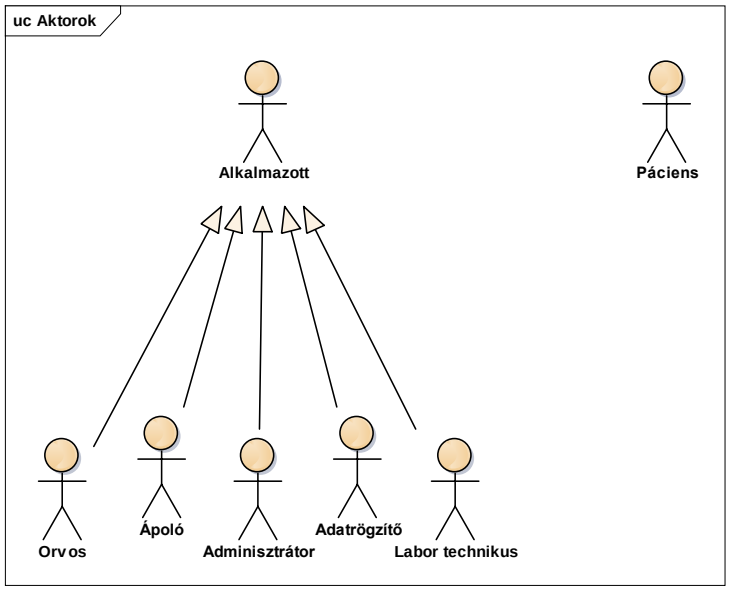


51. ábra PatientDaoLocal::GetProceduresOfMedicalRecord

PatientDaoLocal interfész definiálja az összes szükséges metódust, amire a **PatientDao** osztálynak szüksége van. Utóbbi a keretrendszer segítségével lekéri (**getProceduresOfMedicalRecord)** a kívánt adatokat az adatbázisból. Jelen esetben a kezeléshez tartozó eljárásokat.

1. melléklet – Használati eset modell

A magánkórház menedzselő alkalmazás felhasználói köre alapvetően két csoportra osztható, alkalmazottakra és páciensekre. Az alkalmazottak között megkülönböztetjük tevékenységi körük szerint az alábbi ábrán látható beosztásokat.

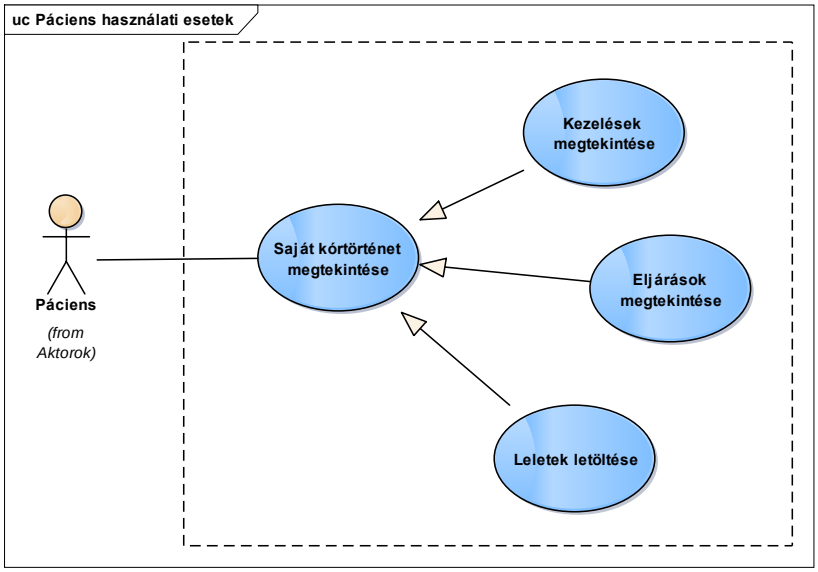


52. ábra aktorok

A különböző felhasználóknak különböző funkciók elérését teszi lehetővé a rendszer. Mivel az egyes funkciók jogosultsági körhöz (szerepkörhöz) lehet kötve, ezért némelyik felhasználói csoport nem ér el bizonyos funkciókat, csak azok egy részét.

Páciens

A páciens funkciói gyakorlatilag csak a web felületen érhetők el.



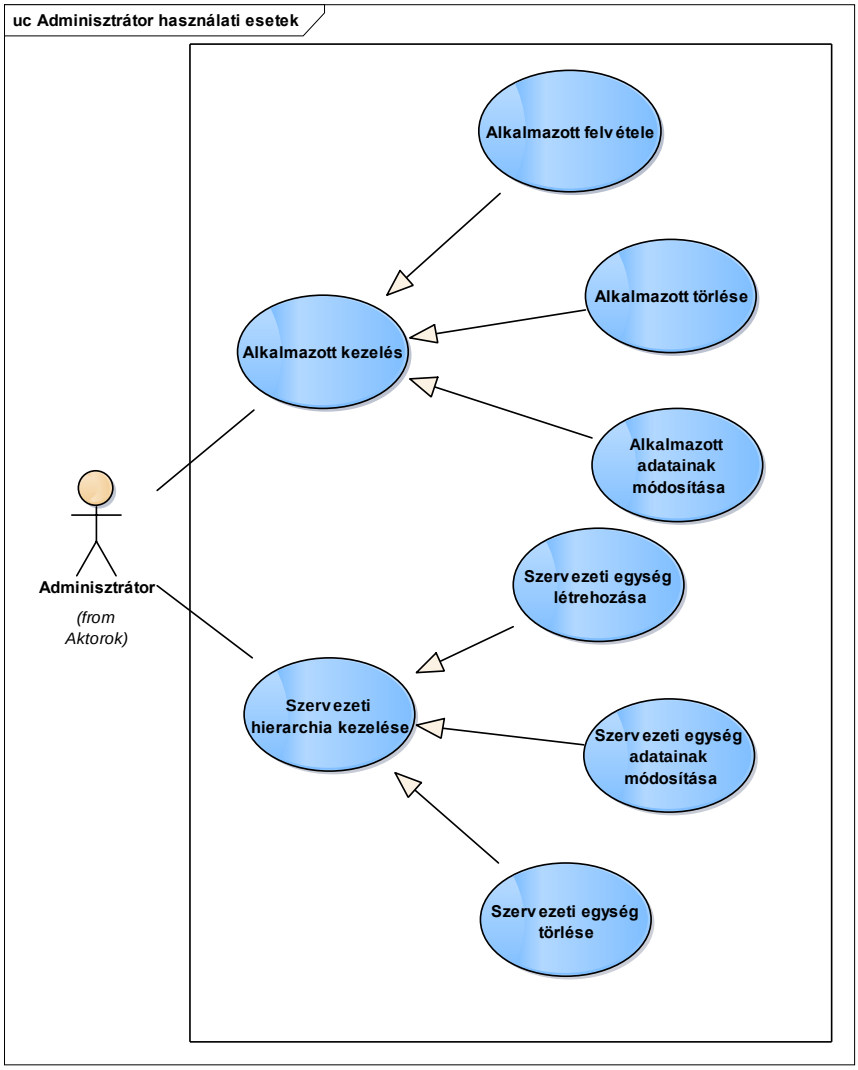
53. ábra páciens

Saját kórtörténet megtekintése:

1. Bejelentkezés a web felületre tajszám és születési dátummal
2. Kezelés kiválasztása
3. Eljárás kiválasztása
4. Leletek letöltése, diagnózis és egyéb információk megtekintése

Adminisztrátor

Ez az alkalmazotti szerepkör felel a kórházi hierarchia kezeléséért. Alkalmazottak és az intézeti egységek kezelésére egy-egy használati eset létezik, melyekből származnak az egyéb, specializáló használati esetek.



54. ábra Adminisztrátor

Alkalmazott kezelése:

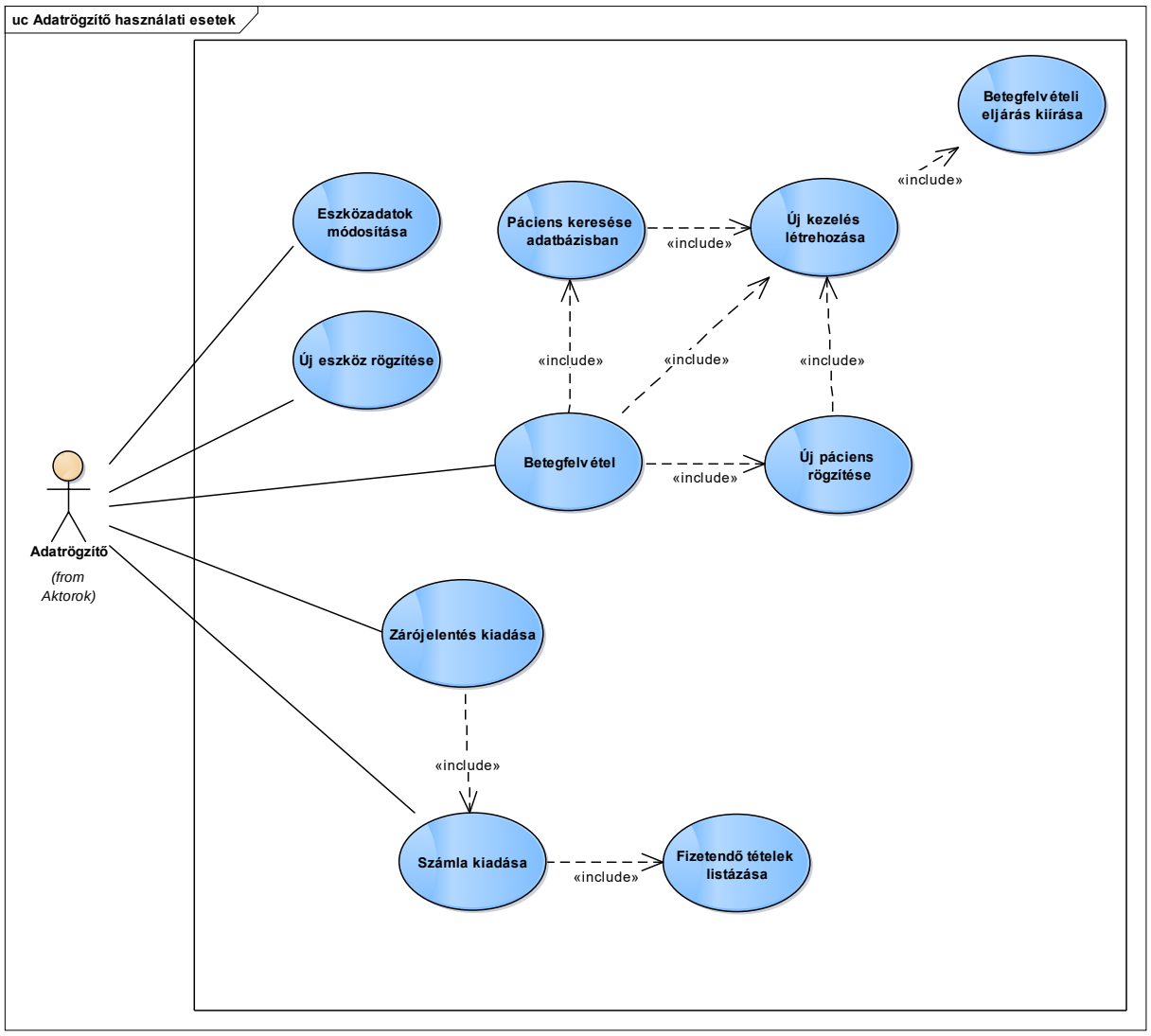
1. Alkalmazott keresése a rendszerben
2. Új alkalmazott felvétele, megtalált alkalmazott adatainak módosítása, alkalmazott törlése

Szervezeti hierarchia kezelése:

1. Szervezeti egység keresése a rendszerben
2. Új egység hozzáadása, adatok módosítása vagy törlése

Adatrögzítő

Ez a szerepkör látja el a recepciós feladatokat, fogadja a betegeket érkezéskor, illetve az elbocsátáshoz szükséges műveleteket hajtja végre, például beteg rögzítése a rendszerben, számlák kiállítása stb.



55. ábra Adatrögzítő

Betegfelvétel

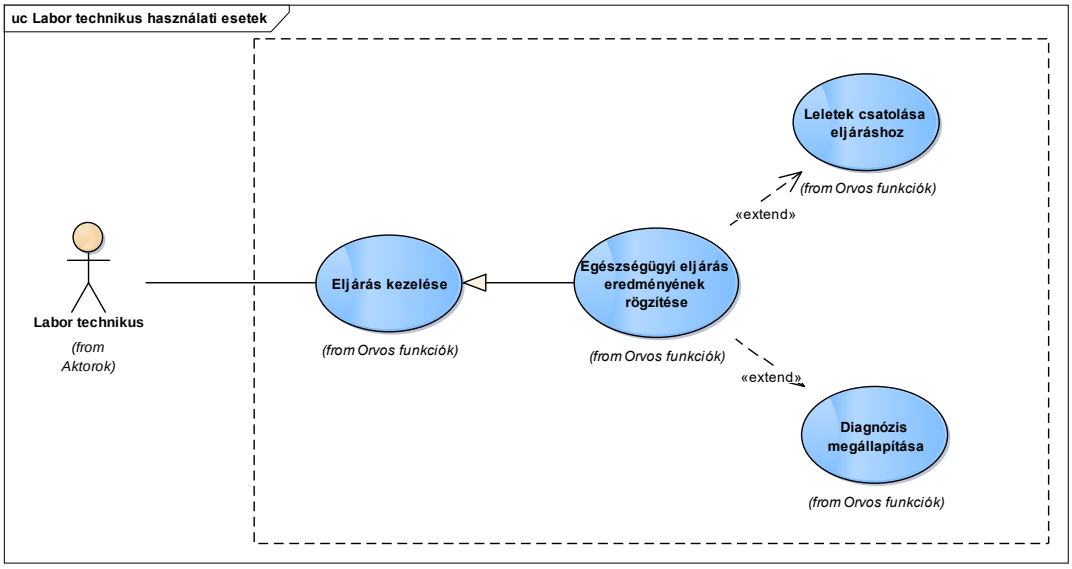
1. Páciens keresése az adatbázisban
2. Páciens kiválasztása/új páciens hozzáadása
3. Új kezelés hozzáadása
4. Betegfelvételi eljárás kiírása a páciens igénye szerinti osztályra

Beteg elbocsátás

1. Számla készítése a fizetetlen tételekből
2. Zárójelentés készítése

Labor technikus

Ez a szerepkör az orvosi funkciók egy részét látja el, azoknak is egy specializált folyamatát. A gyakorlatban ilyen lehet vérképek készítése, szövettani elemzések stb.



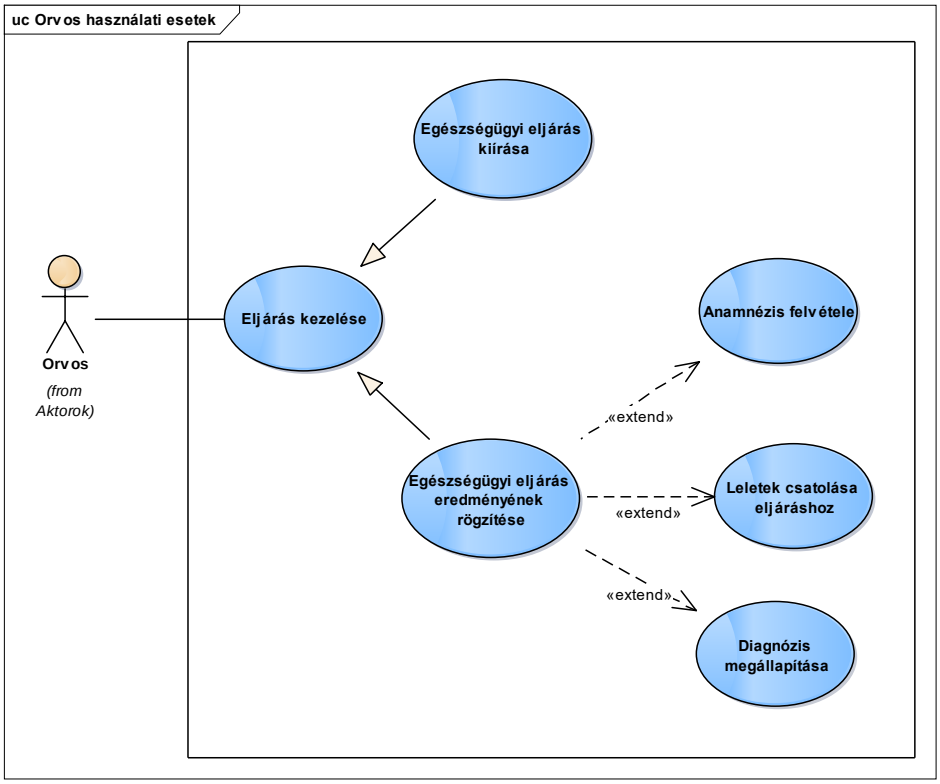
56. ábra Labor technikus

Eljárás kezelése

1. Egészségügyi eljárás keresése
2. Leletek csatolása az eljáráshoz
3. Diagnózis megállapítása és rögzítése a vizsgálati eredmények alapján

Orvos

Az egyik legszélesebb körű funkcionalitással rendelkező szerepkör. Lényegében a teljes kórtörténeti szerkesztés elérhető számára.



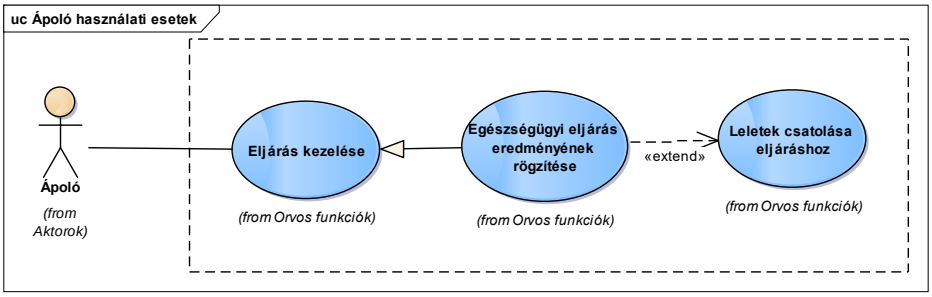
57. ábra Orvos

Eljárás kezelése

1. Új eljárás kiírása/eljárás eredményeinek rögzítése
2. Eljárás adatainak módosítása: anamnézis felvétele, leletek feltöltése, diagnózis rögzítése

Ápoló

Ehhez a szerepkörhöz tartozik a legkevesebb funkcionalitás, lényegében csak vizsgálati eredmények rögzítéséhez van jogosultsága (például röntgenfelvétel feltöltése stb.).



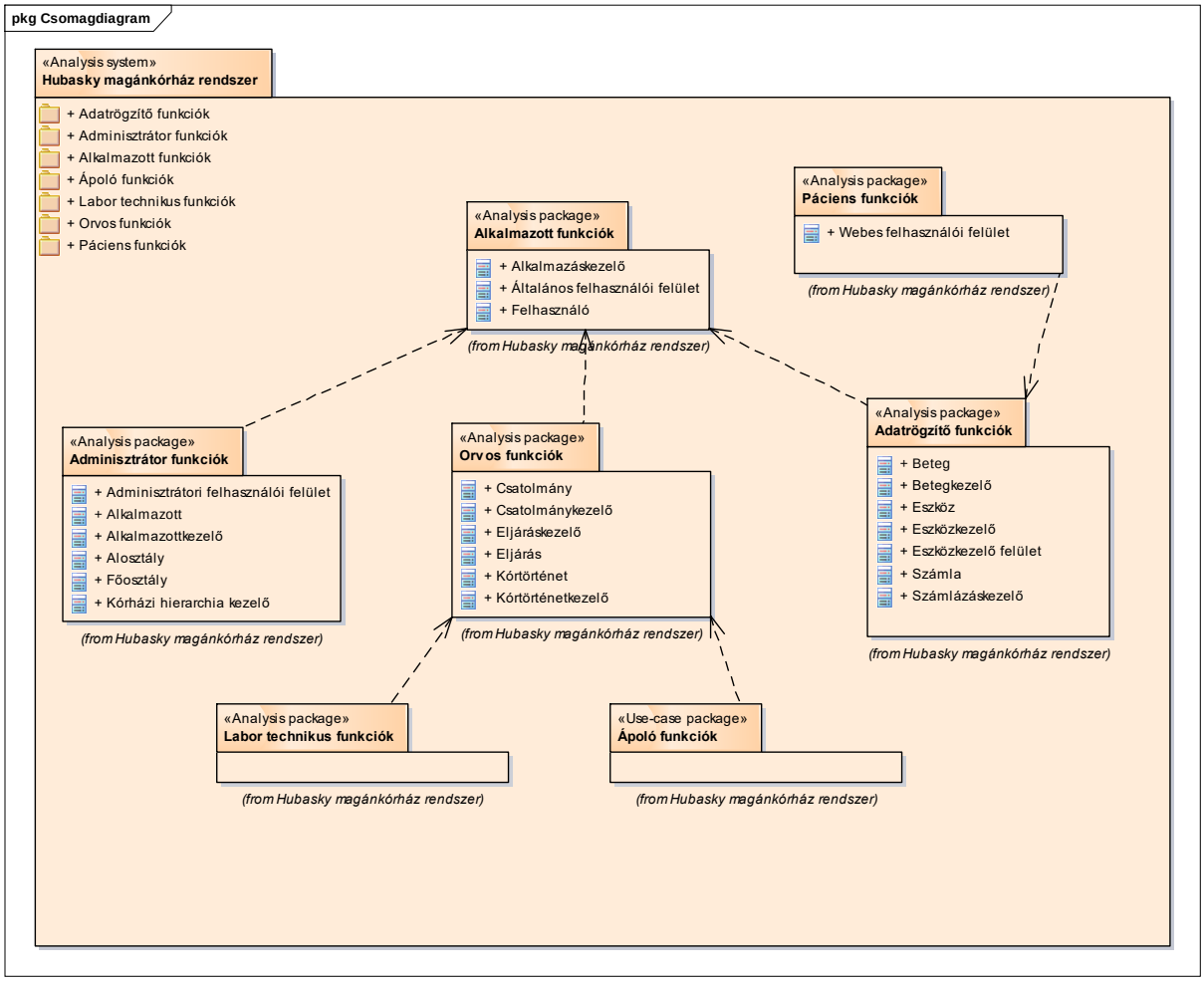
58. ábra Ápoló

Eljárás kezelése

1. Eljárás kiválasztása
2. Lelet csatolása, feltöltése a rendszerbe

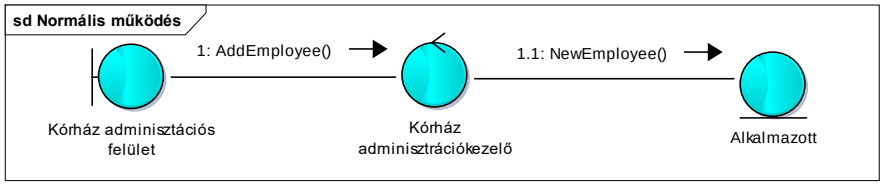
2. melléklet – Analízis modell

Csomagdiagram

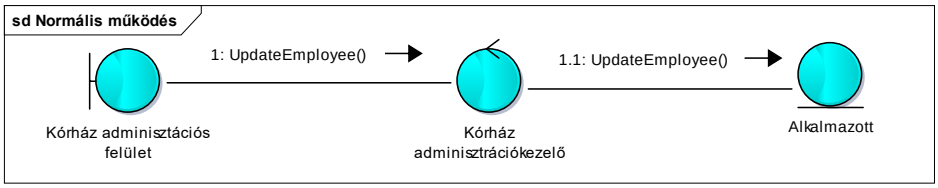


59. ábra Csomagdiagram

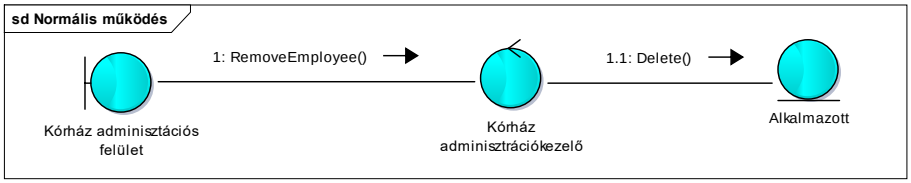
Kommunikációs diagramok



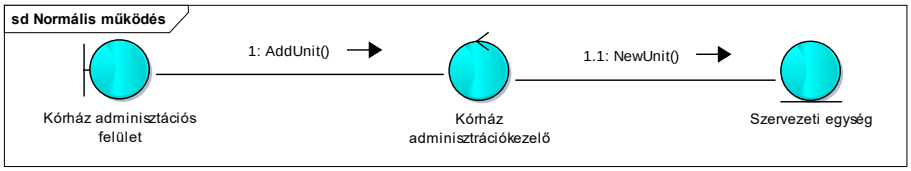
60. ábra Új alkalmazott felvétele



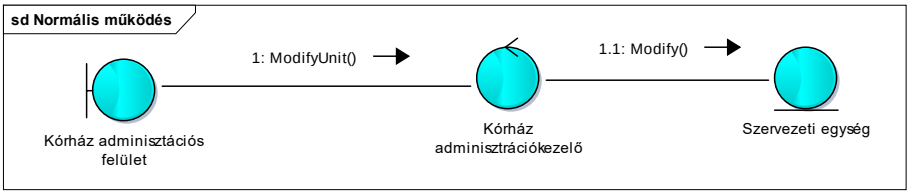
61. ábra Alkalmazott adatainak változtatása



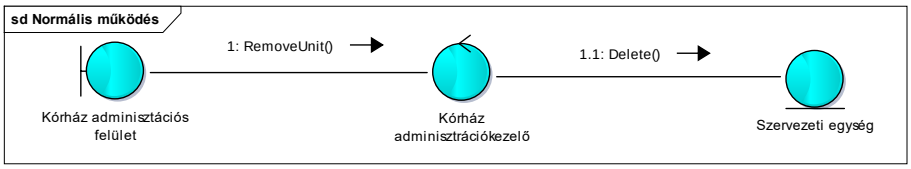
62. ábra Alkalmazott törlése



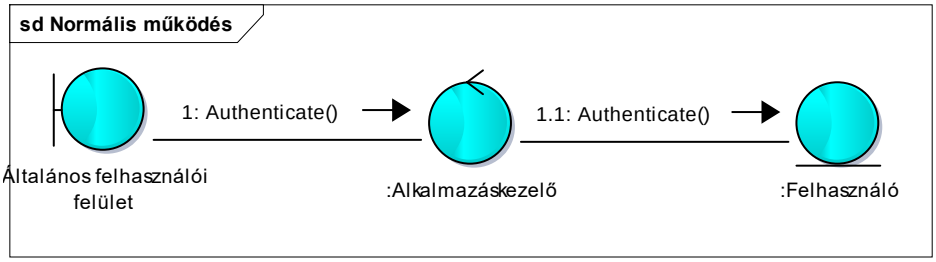
63. ábra Új szervezeti egység felvétele



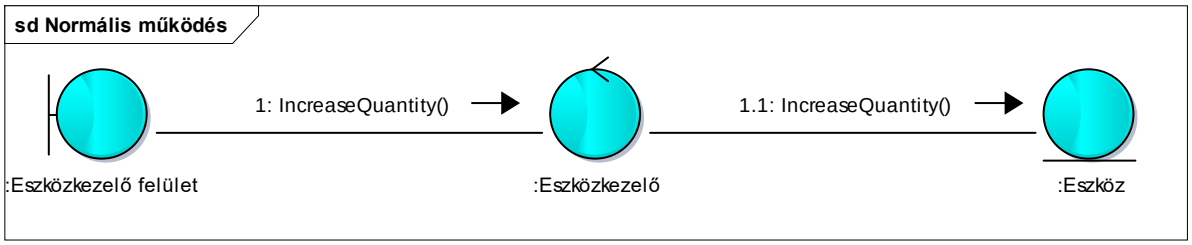
64. ábra Szervezeti egység adatainak változtatása



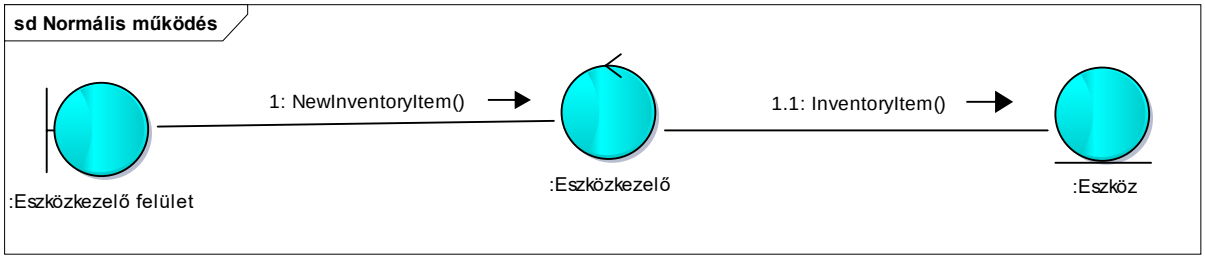
65. ábra Szervezeti egység megváltoztatása



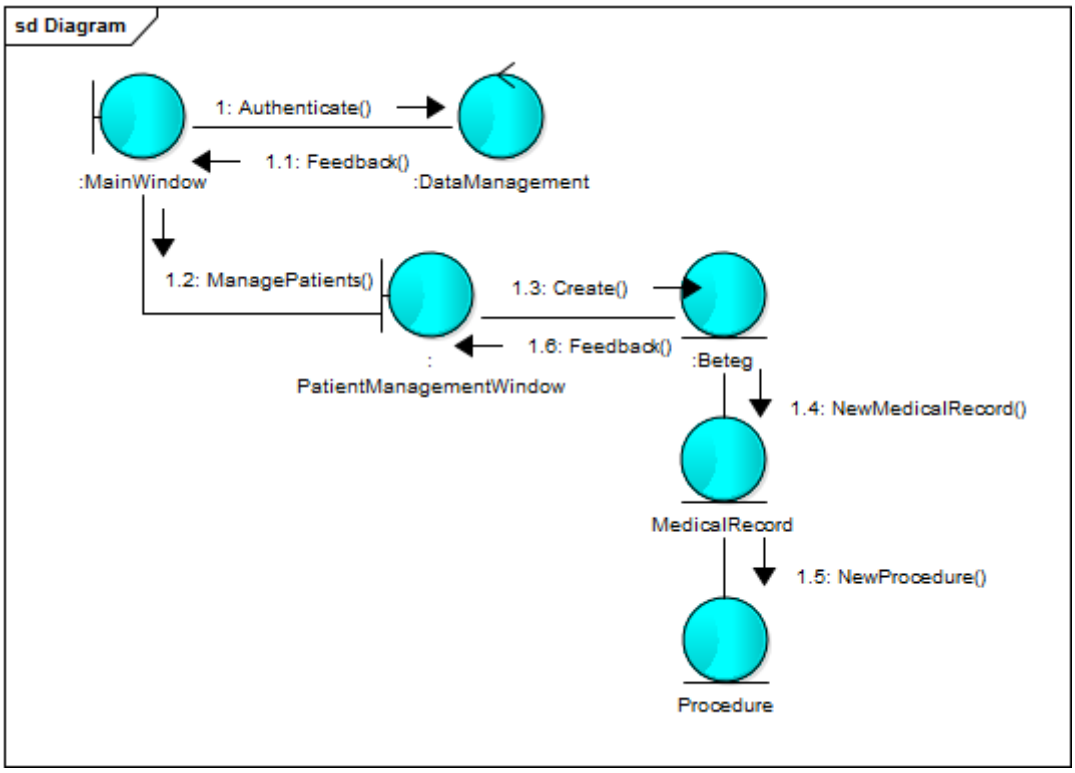
66. ábra Autentikáció



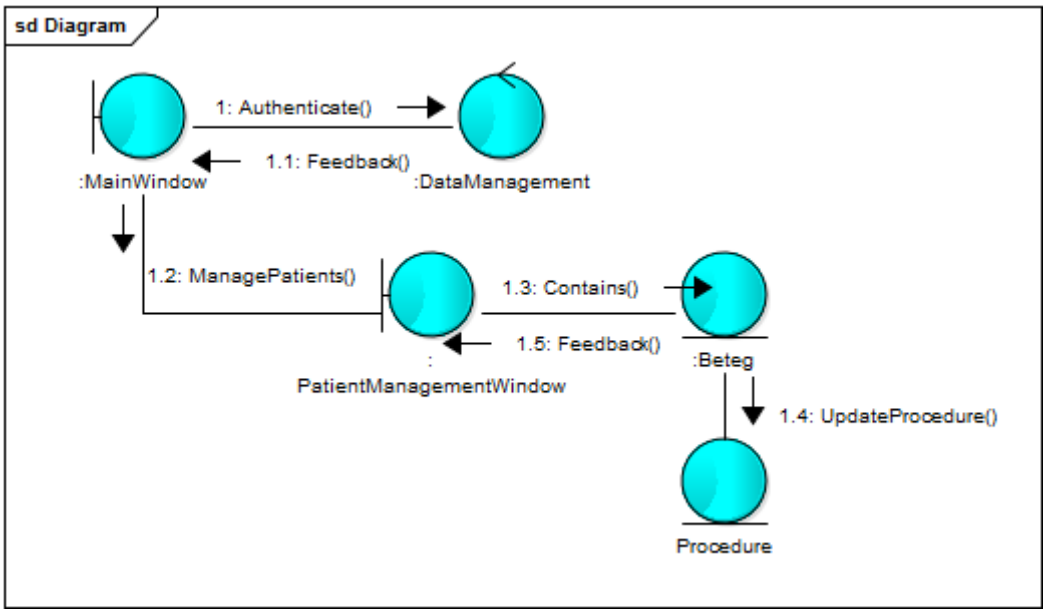
67. ábra Eszközadatok megváltoztatása



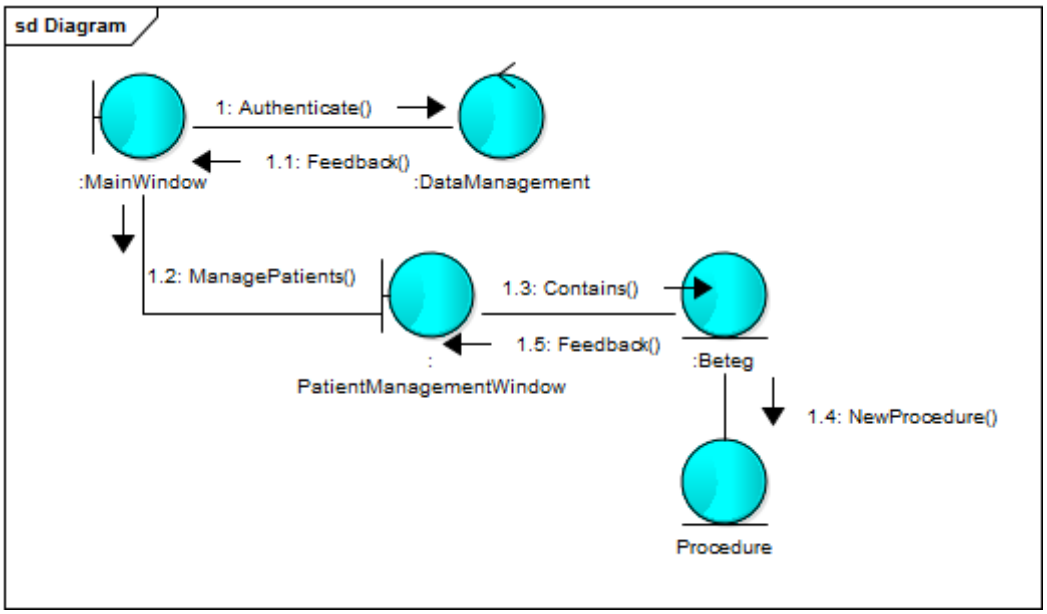
68. ábra Új eszköz felvétele



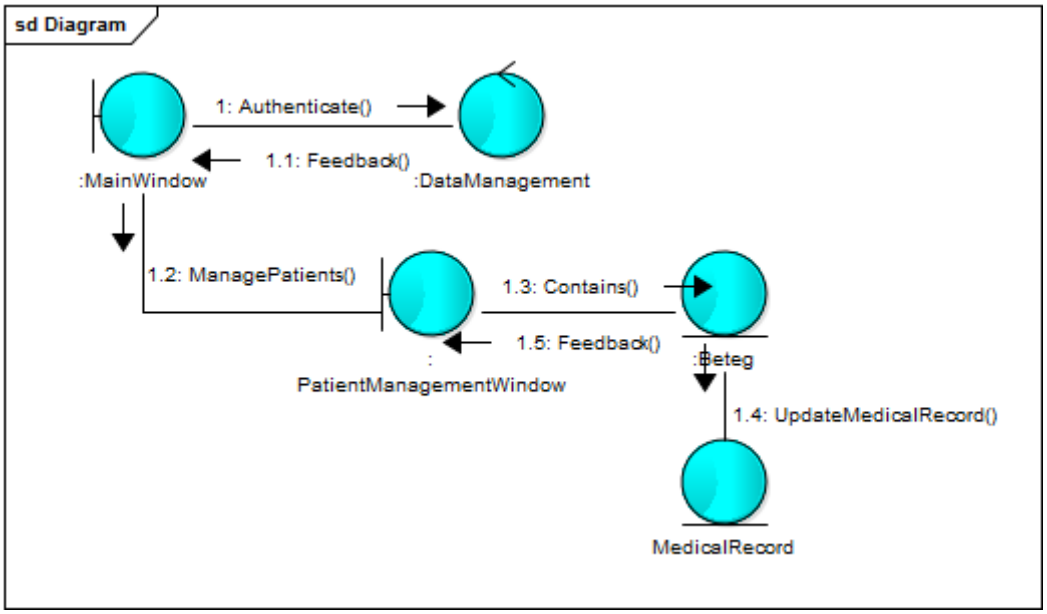
69. ábra - Betegfelvétel



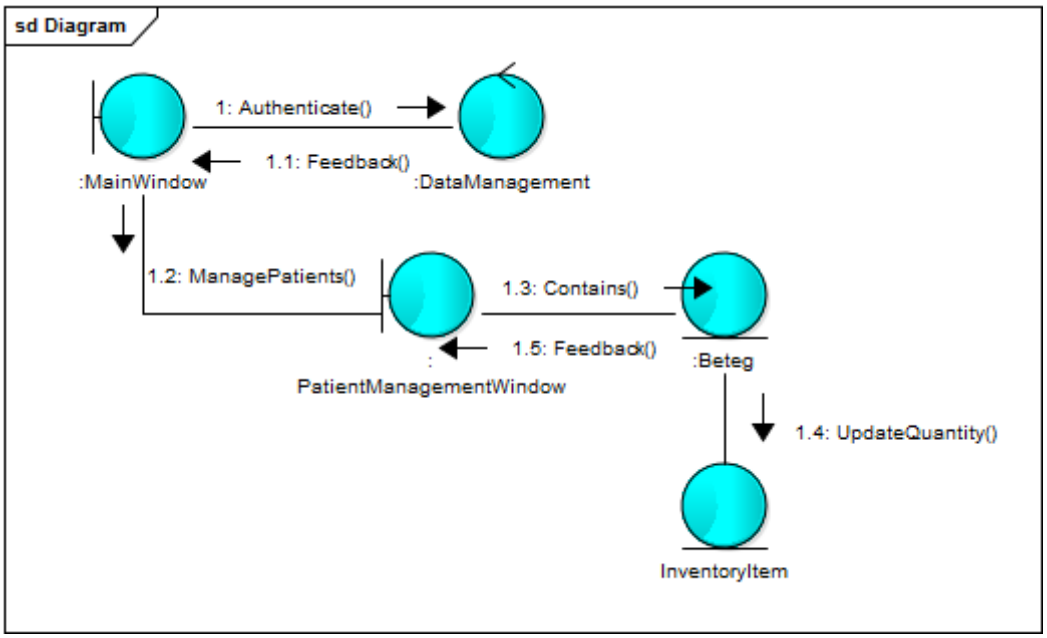
70. ábra – Az egészségügyi eljárás eredménye rögzítése



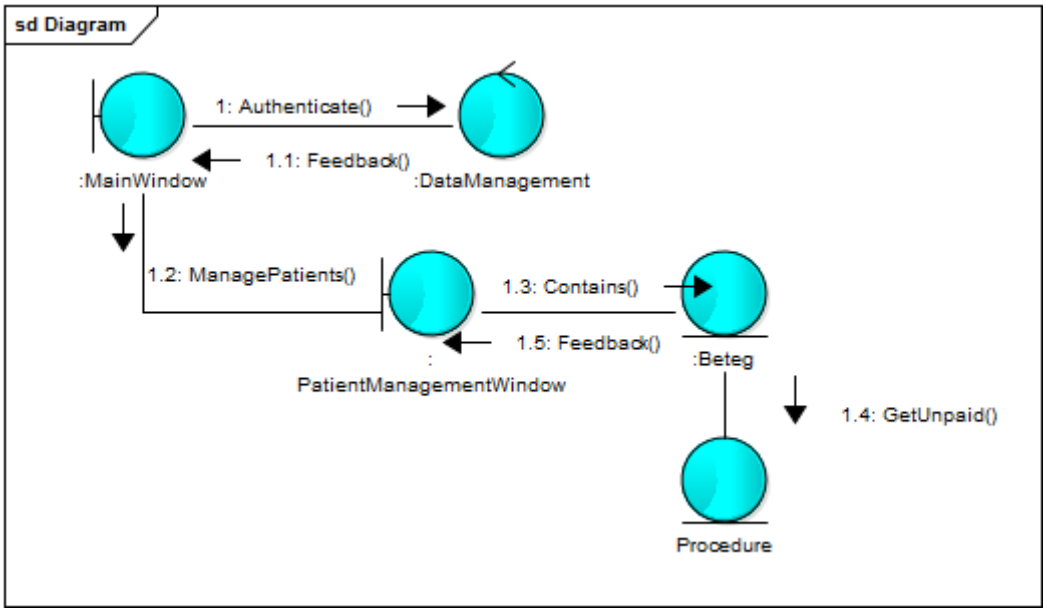
71. ábra – Új egészségügyi eljárás rögzítése



72. ábra – Kórtörténet szerkesztése

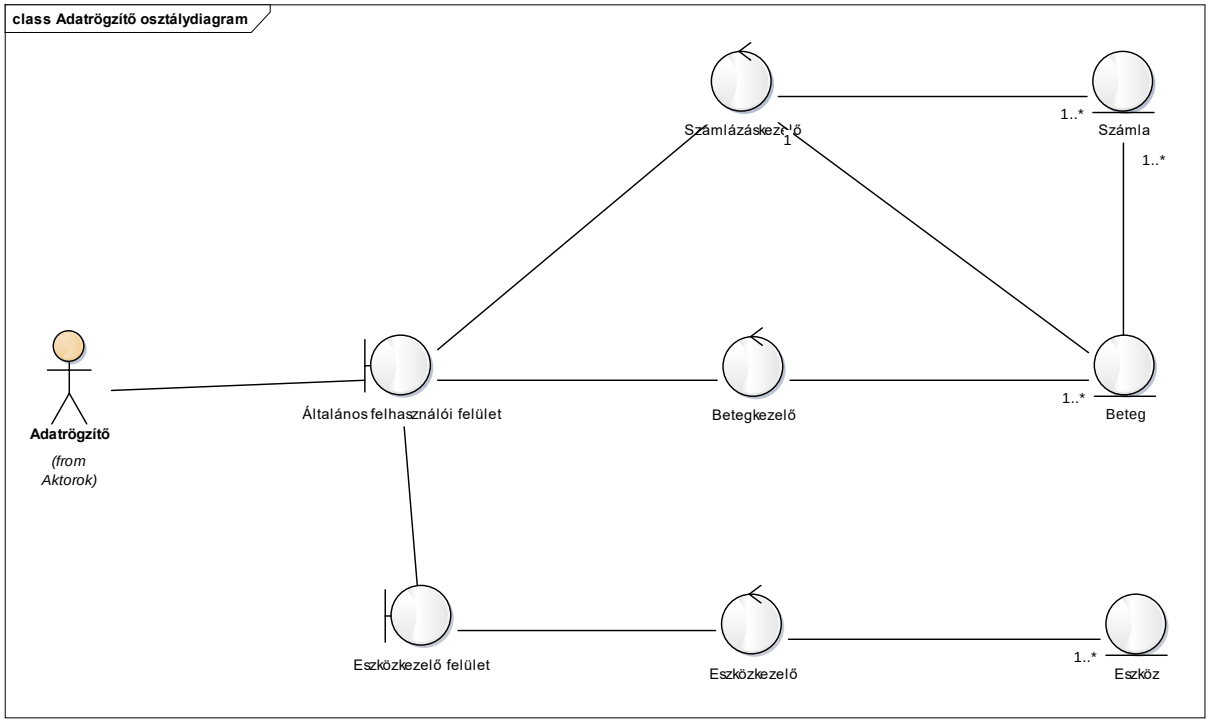


73. ábra – Eszköz törlése

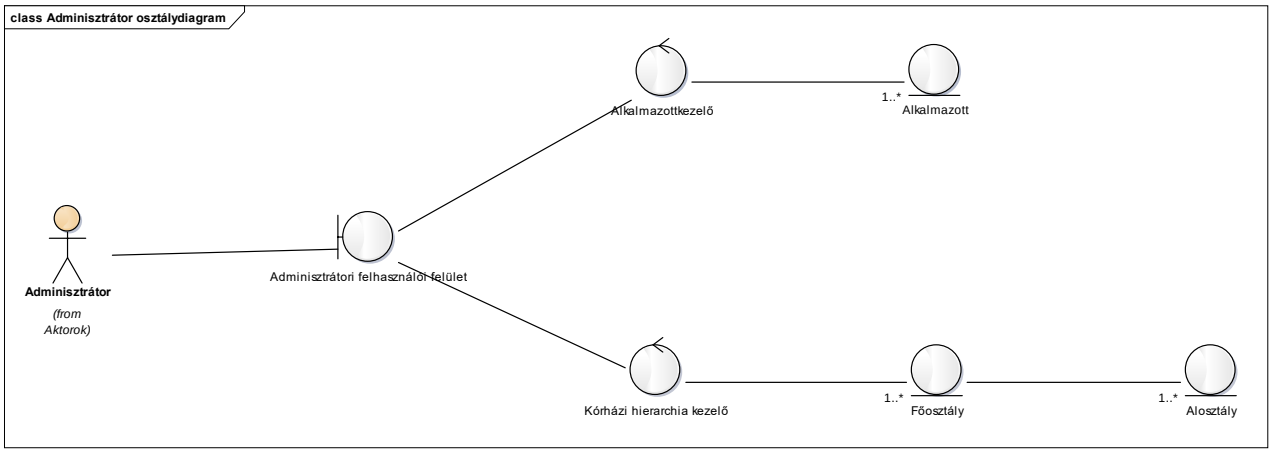


74. ábra – Számla kiállítása

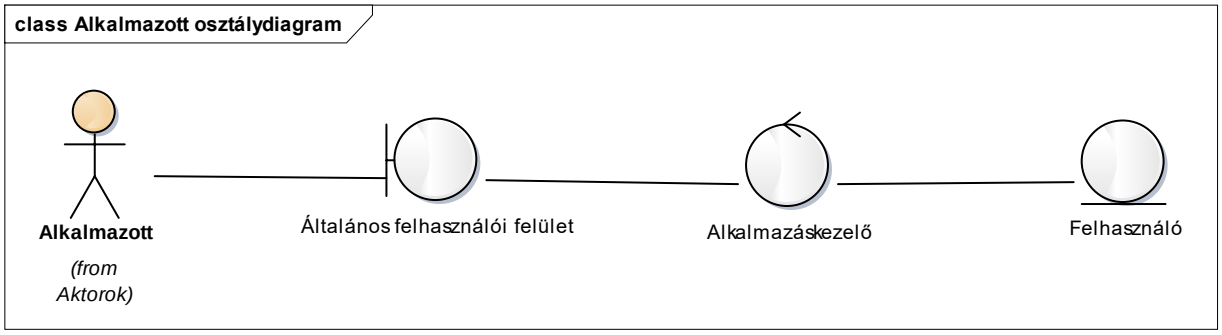
Osztálydiagramok



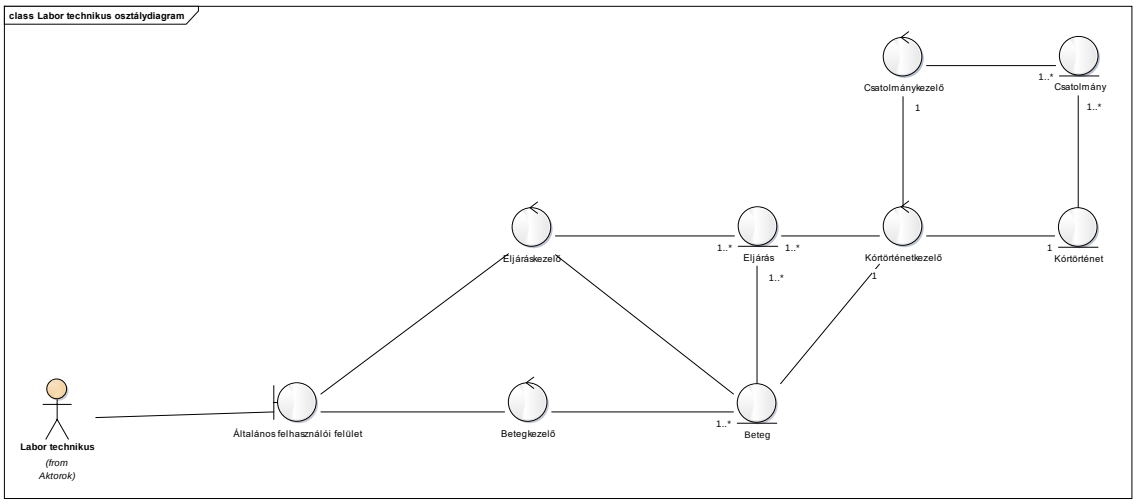
75. ábra Adatrögzítő osztálydiagram



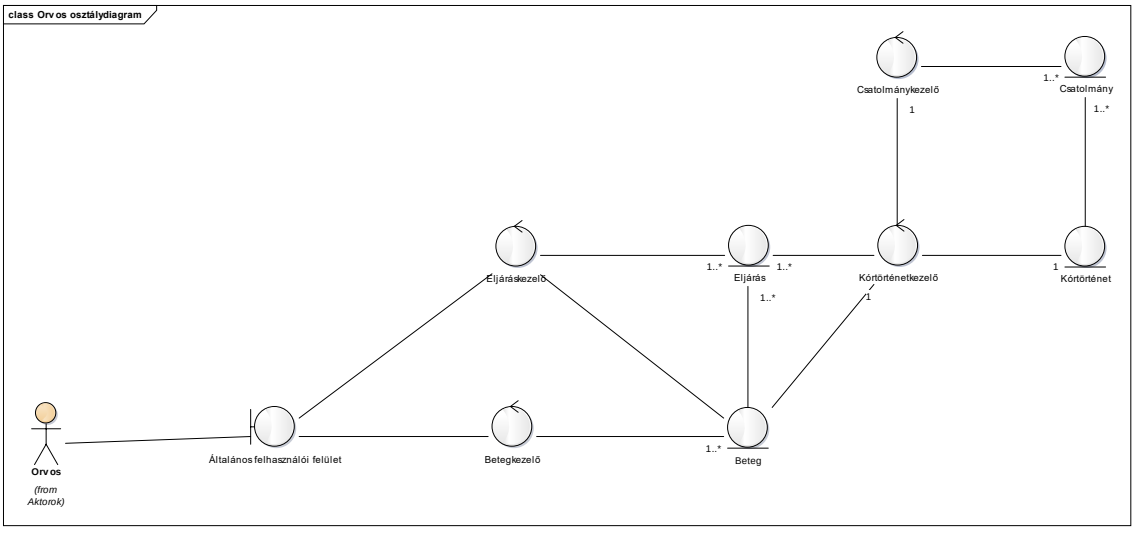
76. ábra Adminisztrátor osztálydiagram



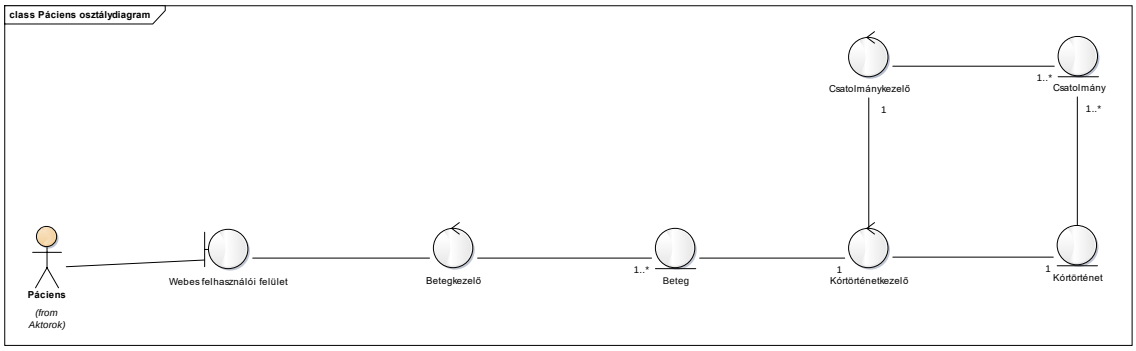
77. ábra Alkalmazott osztálydiagram



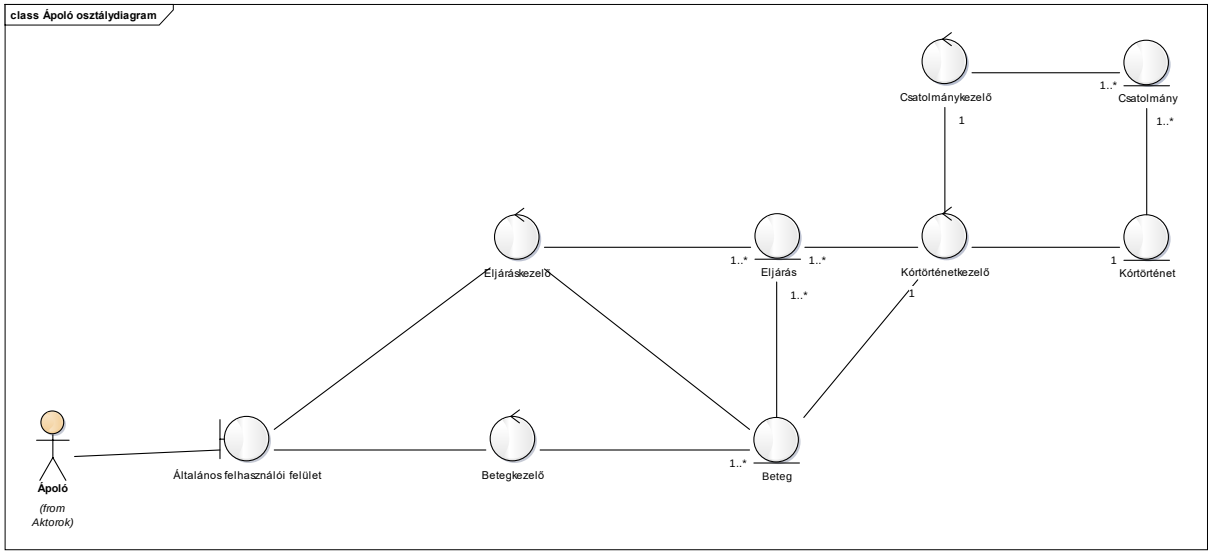
78. ábra Labortechnikus osztálydiagram



79. ábra Orvos osztálydiagram



80. ábra Páciens osztálydiagram



81. ábra Ápoló osztálydiagram

3. melléklet – Jegyzőkönyvek

Jegyzőkönyv

Időpont: 2016.03.27.

Jelen vannak:

Breier Balázs – Kapcsolattartó

Owczarek Artúr – Adminisztrátor

Hazai Péter – Projekt vezető

Stricker Balázs – Demonstrátor

Események:

1. Tervezési modell alkalmazás rétegbeli szerkezet diagramjainak átbeszélése
2. Korábban kialakított osztály struktúrák, kapcsolatok felülvizsgálata

Feladatok kiosztása:

Hazai Péter – HospitalManagement modul

Owczarek Artúr – InventoryManagement, ApplicationManagement modulok

Stricker Balázs – PatientManagement modul

Breier Balázs – WebDataManagement modul

A járulékos és egyéb feladatok közös megbeszélés és munka eredménye.

....

Stricker Balázs
jegyzőkönyvvezető

Hazai Péter
projektvezető

Jegyzőkönyv

Időpont: 2016.03.28.

Jelen vannak:

Breier Balázs – Kapcsolattartó

Owczarek Artúr – Adminisztrátor

Hazai Péter – Projekt vezető

Stricker Balázs – Demonstrátor

Események:

1. Tervezési modell alkalmazás rétegbeli szerkezet diagramjainak véglegesítése
2. Web alkalmazás modul alkalmazás rétegbeli működésének megbeszélése
3. Tervezési modell megjelenítés rétegbeli szerkezet diagramjainak meghatározása

Feladatok kiosztása:

Hazai Péter – HospitalManagement modul

Owczarek Artúr – InventoryManagement, ApplicationManagement modulok

Stricker Balázs – PatientManagement modul

Breier Balázs – WebDataManagement modul

A járulékos és egyéb feladatok közös megbeszélés és munka eredménye.

....

Stricker Balázs
jegyzőkönyvvezető

Hazai Péter
projektvezető

Jegyzőkönyv

Időpont: 2016.03.29.

Jelen vannak:

Breier Balázs – Kapcsolattartó

Owczarek Artúr – Adminisztrátor

Hazai Péter – Projekt vezető

Stricker Balázs – Demonstrátor

Események:

1. Tervezési modell működéssel kapcsolatos használati eset realizációk és interface műveletek átbeszélése
2. Szekvencia diagramok meghatározása
3. Tervezési modell megjelenítés rétegbeli szerkezet diagramjainak véglegesítése

Feladatok kiosztása:

Hazai Péter – HospitalManagement modul

Owczarek Artúr – InventoryManagement, ApplicationManagement modulok

Stricker Balázs – PatientManagement modul

Breier Balázs – WebDataManagement modul

A járulékos és egyéb feladatok közös megbeszélés és munka eredménye.

A fenti modulok az adott személy fő szkópját jelképezik.

....

Stricker Balázs
jegyzőkönyvvezető

Hazai Péter
projektvezető