Universitat Oberta de Catalunya

uoc.edu

MediSearch – Un cercador de proves mèdiques

Xavier Rius Borja
Grau d'Enginyeria Informàtica
Java EE

Albert Grau Perisé
Santi Caballe Llobet

09/01/2019

FITXA DEL TREBALL FINAL

Títol del treball:	*MediSearch – Un cercador de proves mèdiques*
Nom de l'autor:	*Xavier Rius Borja*
Nom del consultor/a:	*Albert Grau Perisé*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*01/2019*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Java EE*
Idioma del treball:	*Català / Anglès*
Paraules clau	*Cercador / Medicina / Mútua*

Resum del Treball (màxim 250 paraules):

MediSearch intenta donar solució al problema que molts cops qualsevol persona s'ha trobat quan, a l'hora d'haver de realitzar una prova mèdica, no ha trobat de forma fàcil i àgil a quins centre la pot realitzar, i també amb tota probabilitat, segons la mútua mèdica de la qual n'és membre.

D'aquesta manera ens trobem en un context de donar una eina que bàsicament serà un cercador on, depenent de les dades entrades (bàsicament quina prova mèdica volem rebre informació i si som o no d'alguna mútua), aquest ens donarà totes les opcions disponibles en el mateix cercador. Com que és important tenir una forma fàcil per manipular les diferents dades, el cercador incorpora una gestió d'usuaris, on depenent dels permisos, es podrà tenir accés des del propi cercador a la base de dades.

La metodologia emprada ha estat basada en les tècniques de l'enginyeria de programari pel desenvolupament, des de la creació de la idea inicial fins a la implementació final amb tecnologia Java EE.

Com a resultat s'ha aconseguit una documentació amb el desenvolupament del programari amb les diferents decisions presses i models empleats, així com el programari desitjat, el qual realitza la funció esperada.

Les principals conclusions és que s'ha creat una eina que pot donar resposta de forma ràpida i eficaç, í que són també moltes les línies de futur que hi ha en aquesta aplicació, ja que l'abast inicial no s'ha pogut acabar de complir degut principalment al temps necessari.

Abstract (in English, 250 words or less):

The aim of the present final degree project is to give an answer to a person who's willing to do a medical test and doesn't know where to attend. Moreover, and most probably the case for many potential users, this person could be a member of a Health Insurance Company which will be associated with some Health Centers, and thanks to this new tool, it could tell the member where to go.

Therefore, this is mainly a search tool where depending on user's input data, it will give a list of health center where the user could attend his desired medical test. Apart of its main search function, a data base is needed to maintain all the data up to date from the user interface. Then, differents user's roles has been created to give them some rights to have access or not to the data base. Also, a normal user has the option to register in the system and introduce which it's health insurance company for fast future searches.

The metodology applied in this project is based on Software engineering methods studied during the degree, which encompasses from the creation of the initial idea, the development of the diferent usage case till the developement of the software with Java EE technology.

The final result is a expalation document with all the steps carried out and the aplication itself with the main functions. Due to the big scope projected and short time, there's a few future work to be done.

Índex

1. Introducció

1.1 Context i justificació del Treball

Avui en dia es ben segur que després d'una visita al metge, al sortir aquest ens hagi demanat la realització d'una prova mèdica, de tipus i especialitat que sigui, per la propera visita amb aquest. Segurament el primer cop vam pensar que l'acció de trobar el lloc on realitzar-la seria una tasca senzilla, però molts cops degut a la gran quantitat de possibilitats i molts entra també la variable de si som d'una mútua mèdica, aquest pot ser un procés molt més farragós del esperat.

Actualment aquesta cerca la realitzaríem mitjançant internet al navegador web habitual, o en cas de disposar de mútua mèdica, ens posaríem en contacte amb aquesta perquè ens informi de les possibilitats. Gràcies a aquesta solució proposada, tindríem en un sol lloc totes les possibles mútues, proves mèdiques i centres mèdics on realitzar-les, on de forma fàcil i àgil ens donarà resposta en cas de necessitat de realitzar alguna prova mèdica.

A més, també ens podrà informar de les autoritzacions necessàries i el lloc on aconseguir-les, la documentació prèvia a portar, les prescripcions mèdiques per la realització d'aquesta, així com el mètode per realitzar la cita. Aquesta aplicació tracta de facilitar a l'usuari tot aquest procés, i porta per nom: "MediSearch".

A més, serà necessària una funcionalitat de *BackOffice* on els administradors (les mútues, clíniques, hospitals, encarregats del manteniment de la pàgina, etc.) hauran d'anar introduint i/o actualitzant les dades necessàries pel funcionament de l'aplicació.

1.2 Objectius del Treball

- Desenvolupar la idea inicial mitjançant la metodologia de la enginyeria de programari per realització del programari corresponent amb tecnologia Java EE.
- Llistar els possibles centres on l'usuari es pot fer la prova, segons les dades entrades per l'usuari (tipus de prova i mútua a la que pertany).
- Funcionalitats de *backoffice* per la gestió de la informació del sistema.
- Funcionalitat de registre de l'usuari per tenir emmagatzemar la mútua d'aquest funcions més avançades.
- Llistar totes les autoritzacions necessàries per la realització de la prova mèdica, i indicar els mètodes de com aconseguir-les, segons la informació entrada en el objectiu anterior.
- Llistar tots els documents necessaris per la realització de la prova (en alguna necessites petició metge + autorització metge, o per exemple,

per una eco de segon semestre necessites portar la eco del primer semestre, etc...)
- Informar a l'usuari de les prescripcions necessàries per la realització de la prova (estar en dejú, etc.)

1.3 Enfocament i mètode seguit

Per la realització d'aquest treball i partint de la base que es buscava fer un producte que solucionés una necessitat, i amb l'objectiu també de poder aplicar moltes tècniques apresses durant el Grau d'Enginyeria Informàtica, s'ha escollit la realització tot el cicle de vida del desenvolupament de programari. Per tant, s'ha partit de la idea inicial, obtenció dels requisits, el model de casos d'ús, un model de pantalles, el diagrama estàtic d'anàlisi fins el disseny de la bases de dades relacional. Amb tota aquesta documentació generada, s'ha passat a l'elecció dels *frameworks* amb els que desenvolupar l'aplicació, per acabar amb el desenvolupament d'aquest.

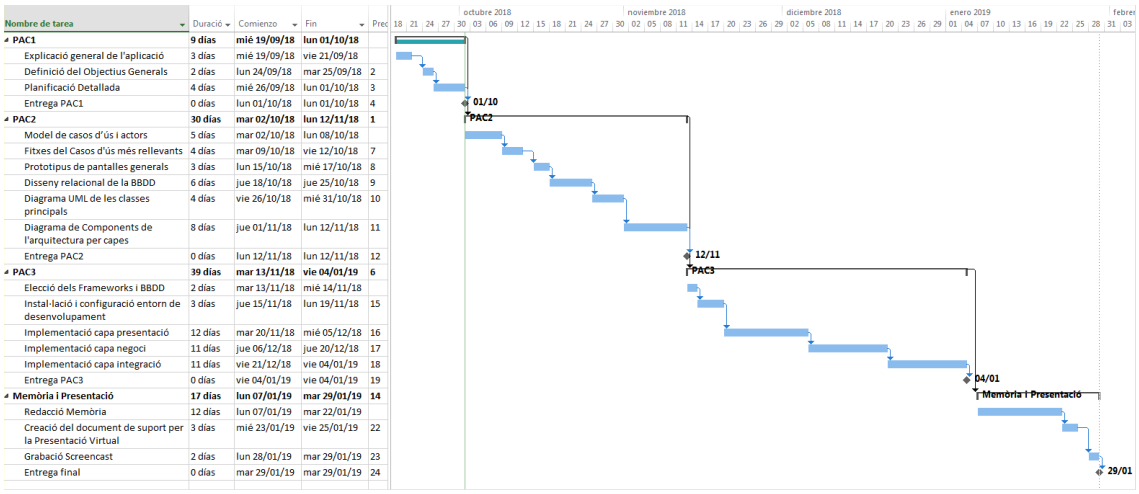
La valoració que es fa d'aquest estratègia és creu que ha estat molt encertada degut a que a permès afrontar totes el passos necessaris per un projecte de desenvolupament de programari requereix, així com posar en pràctica real diferents mètodes assimilats durant el Grau.

1.4 Planificació del Treball

Per la realització d'aquest projecte s'ha utilitzat les següents eines de programari pel desenvolupament del producte final:

- **IDE Eclipse**: és el entorn de desenvolupament escollit per familiaritat de treball, amb el que es desenvoluparà tota la programació de l'aplicació en Java.
- **Maven**: programari per la gestió i construcció (build) de projectes de programari. Es una alternativa a la eina *Apache Ant*, però que ens aportarà més facilitats i funcionalitats que aquesta.
- **Aplicació del Patró MVC**: per la familiaritat amb aquest i bon desacoblament entre capes, es farà servir Java Server Faces, amb la biblioteca de components Primefaces.
- **Lògica de Negoci**: es farà mitjançant l'estàndard Java EE, és a dir amb Enterprise Java Beans.
- **Persistència de dades**: s'ha escollit la base de dades relacional PostgreSQL.

Tot seguit trobem la planificació del treball en format diagrama de Gantt:



Nombre de tarea	Duració	Comienzo	Fin	Prec
PAC1	**9 días**	**mié 19/09/18**	**lun 01/10/18**	
Explicació general de l'aplicació	3 días	mié 19/09/18	vie 21/09/18	
Definició del Objectius Generals	2 días	lun 24/09/18	mar 25/09/18	2
Planificació Detallada	4 días	mié 26/09/18	lun 01/10/18	3
Entrega PAC1	0 días	lun 01/10/18	lun 01/10/18	4
PAC2	**30 días**	**mar 02/10/18**	**lun 12/11/18**	**1**
Model de casos d'ús i actors	5 días	mar 02/10/18	lun 08/10/18	
Fitxes del Casos d'ús més rellevants	4 días	mar 09/10/18	vie 12/10/18	7
Prototipus de pantalles generals	3 días	lun 15/10/18	mié 17/10/18	8
Disseny relacional de la BBDD	6 días	jue 18/10/18	jue 25/10/18	9
Diagrama UML de les classes principals	4 días	vie 26/10/18	mié 31/10/18	10
Diagrama de Components de l'arquitectura per capes	8 días	jue 01/11/18	lun 12/11/18	11
Entrega PAC2	0 días	lun 12/11/18	lun 12/11/18	12
PAC3	**39 días**	**mar 13/11/18**	**vie 04/01/19**	**6**
Elecció dels Frameworks i BBDD	2 días	mar 13/11/18	mié 14/11/18	
Instal·lació i configuració entorn de desenvolupament	3 días	jue 15/11/18	lun 19/11/18	15
Implementació capa presentació	12 días	mar 20/11/18	mié 05/12/18	16
Implementació capa negoci	11 días	jue 06/12/18	jue 20/12/18	17
Implementació capa integració	11 días	vie 21/12/18	vie 04/01/19	18
Entrega PAC3	0 días	vie 04/01/19	vie 04/01/19	19
Memòria i Presentació	**17 días**	**lun 07/01/19**	**mar 29/01/19**	**14**
Redacció Memòria	12 días	lun 07/01/19	mar 22/01/19	
Creació del document de suport per la Presentació Virtual	3 días	mié 23/01/19	vie 25/01/19	22
Grabació Screencast	2 días	lun 28/01/19	mar 29/01/19	23
Entrega final	0 días	mar 29/01/19	mar 29/01/19	24

1.5 Breu sumari de productes obtinguts

En aquest projecte s'ha obtingut un sol producte, el programari MediSearch, una aplicació web seguint l'estàndard JEE, el qual defineix una arquitectura heterogènia amb sistema client-servidor organitzat en n capes, on nosaltres farem servir el mínim que defineix: 3 capes (presentació, negoci i integració). D'aquesta manera el processament de l'aplicació (entra el tipus de prova mèdica o el nom de la mútua), i l'administració de les dades (de tots els centres on es poden realitzar proves mèdiques, prescripcions mèdiques d'aquestes, etc.) seran processos separats lògicament. Gràcies també a la distribució per capes afavorim l'escalabilitat i tal i com hem apuntat, podrem aïllar les funcionalitats que ofereixen en capes amb interfícies ben definides.

Tal i com defineix Java EE, aquesta és una plataforma basada en components distribuïts en cada capa, on la interacció entre aquests es dura a terme mitjançant invocacions de les seves operacions. Es definiran els components amb la màxima cohesió (cada component s'encarrega de fer la funció per la qual ha estat dissenyat) i minimitzar l'acoblament entre ells, per maximitzar el manteniment.

1.6 Breu descripció dels altres capítols de la memòria

En aquest document trobarem una descripció de tots el processos que s'ha dut a terme pel desenvolupament del programari final, començant per la definició dels objectius, l'obtenció de requisits, el model de casos d'ús que s'extreu del pas anterior, una definició de la interfície gràfica d'usuari, el diagrama estàtic d'anàlisi a partir del qual farem el disseny relacional de la base de dades.

Un cop tinguem el punt de la informació establert, passarem a definir l'arquitectura del sistema que s'implementarà. El següent pas serà ja la part pràctica d'aquest treball, amb la implementació del programari basant-nos en tots els passos previs.

Finalment arribarem a quines línies de futur ens deixa aquest treball final de grau i conclusions, per veure si s'han assolit o no tots els objectius inicialment plantejats.

2. Obtenció de requisits

En aquest apartat definirem quins són els requisits d'aquest programari, els quals seran peça clau per l'èxit o el fracàs de tot el projecte. Amb aquests expressarem les característiques observables del nostre sistema que denoten una necessitat o restricció que un actor o *stakeholder* té sobre aquest. Hem de recordar que un *stakeholder* és qualsevol persona o entitat que té un interès en el projecte. També ens delimitaran quines de les possibles solucions del problema plantejat inicialment són adequades perquè compleixen els requisits definits, i quines no (per igual condició).

Dins de la classificació conceptual dels tipus de requisits de l'assignatura Enginyeria de requisits, aquí només farem referència als requisits de producte, degut a que els requisits de procés que creiem es troben en un treball de grau son inexistents, on no tenim en compte el cost del desenvolupament en hores, temps o altres. Només val acabar amb èxit.

2.1 Identificació de stakeholders i entrevistes

Per la obtenció dels requisits farem servir una de les tècniques més utilitzades, la qual consisteix a entrevistar-se amb els stakeholders per obtenir d'ells les necessitats que tenen del sistema a desenvolupar.

- **Persona què ha de realitzar una prova mèdica**: el seu principal interès es poder saber els llocs on pot realitzar la prova mèdica, horaris en que es realitzen les proves, si necessita o no autorització de la seva mútua (en cas de tenir-ne), etc. També té interès en poder guardar la informació que l'aplicació li pot proporcionar, mitjançant l'enviament d'aquesta en un correu electrònic, missatge SMS, etc. Podem trobar usuaris que no siguin de cap mútua però vulguin saber en quins centres privats es poden fer la prova i preu, degut a que en el sistema públic hi ha una cua molt llarga. N'hi haurà que els hi agradaria poder disposar d'un usuari dins l'aplicació per emmagatzemar les seves dades (nom i cognoms, mútua/es a les que pertany, direcció correu electrònic), tenir un registre de les proves realitzades, etc. També podem trobar-ne que vulguin saber les opinions que altres usuaris tenen dels centres on han realitzat les proves, segons les seves experiències, per tenir més informació a l'hora d'escollir un o altres centre.

- **Gerent d'una mútua**: interessat a en disposar d'una aplicació que faciliti la realització d'una prova mèdica als seus clients segons la cobertura que ells donen, i poder incrementar la qualitat del servei i satisfacció d'aquests. Interessat també en agilitzar el procés d'obtenció d'autoritzacions per la realització de proves mèdiques pels seus clients, on tradicionalment aquestes s'aconseguien anat en persona a les pròpies oficines de la mútua, però avui en dia ja trobem un número de telèfon específic o una adreça de correu electrònic per demanar-les. A

més, un bon funcionament i acceptació pels clients d'aquesta podria comportar el dedicar menys recursos a les vies actual de gestió d'autoritzacions, i poder dedicar-los a altres departaments. Pot també tenir interessos en disposar d'estadístiques d'utilització d'aquesta, quants dels seus clients la fan servir, etc.

- **Personal d'atenció al client / autoritzacions d'una mútua**: encarregats de la gestió de les autoritzacions pels usuaris de la mútua segons les vies que aquesta disposi, proporcionar informació dels centres associats a la mútua que realitzen proves mèdiques. Un dels principals interessos que tenen és que l'aplicació mostri clarament la informació que un usuari ha de proporcionar a la mútua per tal d'aconseguir una autorització (aquesta és generalment, el nom i cognom, el número de la targeta de la mútua i la petició del metge). També hi ha mútues que la petició d'autorització ha d'anar en un volant de la mútua o acompanyada d'aquest (un volant és una espècie de full de talonari amb codi de barres i número específic de cada mútua). Un aspecte important que comenten, es que moltes mútues no donen una autorització d'una prova fins que no saben el centre on es vol realitzar aquesta. Per tant, volen que aquesta aplicació faci veure clarament als usuaris que primer s'ha de demanar hora al centre desitjat i després l'autorització. Comenten el temor que aquesta nova aplicació els incrementi la feina si han de resoldre dubtes als clients que no se'n surtin amb l'aplicació. Per tant, i per mirar de mitigar-ho, tenen interès en que l'aplicació sigui molt intuïtiva i clara, que doni tota la informació, i orientada a persones de qualsevol rang d'edat i pocs coneixement d'aplicacions informàtiques (principalment gent gran que possiblement son les persones amb més necessitats de proves mèdiques).

- **Gerents dels centres que realitzen proves mèdiques**: tenen un interès en poder donar a conèixer el seu centre a més clients potencials, i així fer pujar el volum de negoci, perquè pensen que molts cops les mútues no donen prou informació sobre els centres mèdics on els seus clients es poden fer proves mèdiques. Comenta que es important fer saber que tota prova mèdica que vulgui fer-se un client d'una mútua associada al seu centre mèdic, ha d'haver-hi obligatòriament una petició / prescripció d'un metge (es a dir, que ha estat sol·licitada per aquest). Aquesta s'ha de portar impresa el dia de la prova o trobem centres que permeten enviar-la per email. Per altra banda, hi ha certes proves mèdiques que els centres necessiten tenir una autorització de la mútua del pacient per realitzar-la. Aquesta s'ha de portar impresa el dia de la prova o algunes faciliten una direcció de correu electrònic per ser enviada prèviament (a vegades és la pròpia mútua que la envia després de la sol·licitud d'un pacient). Poder facilitar el procés d'enviament i recepció d'aquests dos documents es veu com un punt a tenir en compte en la nova aplicació. Per últim també comenta que hi ha certes proves en les que és necessari portar proves mèdiques anteriors per la seva realització (com per exemple portar mamografies anteriors per la realització d'una nova), i l'aplicació hauria de mostrar.

- **Personal d'atenció al client d'un centres que realitzen proves mèdiques**: es l'encarregat de concertar cites per la realització de les proves, així com resoldre dubtes relacionat, etc. Comenta que l'aplicació hauria de mostrar els documents necessaris que els clients han de portar el dia de a prova (aquests sempre són la targeta de la mútua i la petició del metge, i de forma opcional, l'autorització de la mútua, proves anteriors com pot ser una mamografia anterior, prova prèvia necessària com un anàlisis de sang). A més a més i quan aplica, hauria de indicar les preparacions que el pacients han de fer per realitzar la prova (com anar en dejú per una ecografia abdominal o beure 1,5 litre d'aigua per una ecografia reno-vesical). Comenta que seria molt positiu que l'aplicació indiques els dies i horaris en que es poden realitzar cada tipus prova, al ser una de les consultes habituals que reben.

- **Personal mèdic que realitza les proves**: els interessa que els pacients que han de realitzar la prova tingui i puguin consultar de forma clara, quines son les preparacions que el seu centre demana al pacients per realitzar les diferents proves mèdiques (com per exemple anar amb dejú, portar els resultats de proves que s'ha de fer prèviament a la desitjada, etc.). Indiquen que seria interessant que aquesta informació fos enviada al pacient només fer la reserva de la cita, i la pugui consultar en tot moment i de forma fàcil.

- **Administrador de continguts**: persona encarregada d'introduir tota la informació persistent que necessita la aplicació, com totes les dades de la mútua, totes les proves que es poden realitzar en aquesta mútua, etc. El seu interès es poder disposar d'una interfície clara i entenedora a l'hora d'entrar la informació, que permeti fer modificacions de forma ràpida i eficaç en cas d'error, i visualitzar de forma clara la informació emmagatzemada per fer possibles modificacions.

- **Cap del projecte i desenvolupador de l'aplicació web**. En aquest cas i al ser el TFG, el seu principal interès es la resolució amb èxit de tot el projecte, amb

2.2 Identificació de requisits

A partir de la identificació dels stakeholders i sobretot, de les entrevistes aquests, obtenim quines necessitats tenen respecte al desenvolupament de l'aplicació. Una de les tècniques que s'han fet servir és la realització d'entrevistes i qüestionaris, on volem obtenir directament els requisits que tenen sobre el futur sistema.

2.2.1 Requisits funcionals

Aquí llistarem els requisits que defineixen la funcionalitat que ha de proporcionar el sistema i les dades que ha de conèixer i emmagatzemar. També trobarem quines operacions fa el sistema, com manipula les dades, etc. A partir de les entrevistes als diferents stakeholders,

elaborem una llista de requisits seguint el format d'història d'usuari: Com a <paper> vull <requisit>.

Requisits de Funcionalitat

Descriuen el comportament del sistema explicant quina resposta (observable des de fora) ha de donar davant estímuls que li arribin de l'exterior (generalment, accions dels usuaris):

1. Com a persona què tinc de realitzar una prova mèdica i que soc d'una mútua, vull saber el centres on puc realitzar-la i en quins dies de la setmana i horaris.
2. Com a persona que ha de realitzar una prova mèdica i que no sóc de cap mútua, vull consultar els centres privats on puc realitzar-la, i el preu d'aquesta.
3. Com a persona què tinc de realitzar una prova mèdica i que soc d'una mútua, vull saber si necessito autorització d'aquesta per realitzar-la, i les vies per aconseguir-la.
4. Com a persona que ha de realitzar una prova mèdica, vull consultar quines formes tinc per reservar una prova mèdica, i realitzar-la en cas de poder-se fer informàticament.
5. Com a persona què tinc de realitzar una prova mèdica, vull poder rebre els resultats de la meva cerca feta a l'aplicació.
6. Com a persona què tinc de realitzar una prova mèdica, vull poder disposar d'un usuari al sistema per emmagatzemar les meves dades per agilitzar les cerques i tenir un registre de proves realitzades.
7. Com a gerent d'una mútua vull que els meus clients pugin fer peticions d'autoritzacions de proves mèdiques de forma automàtica.
8. Com a personal d'atenció al client / autoritzacions d'una mútua, vull que es mostri la informació necessària per demanar una autorització.
9. Com a personal d'atenció al client / autoritzacions d'una mútua, vull que es mostri un missatge de no realitzar una petició d'autorització sense prèvia reserva al centre on es realitzarà la prova mèdica.
10. Com a gerent d'un centre on es realitzen proves mèdiques, vull que es mostri un missatge que no es pot realitzar cap prova mèdica sense una petició d'un metge als clients de mútues associades que desitgen fer-ne una en el meu.
11. Com a gerent d'un centre on es realitzen proves mèdiques, vull que es pugui enviar de forma automàtica les autoritzacions de les diferents mútues a una adreça de correu específica del centre.
12. Com a personal d'atenció al client d'un centres que realitzen proves mèdiques, vull que es mostri els documents necessaris que els clients han de portar el dia en que realitzen la prova.
13. Com a personal mèdic que realitza les proves, vull que es mostri i s'enviïn per correu electrònic les preparacions que el centre demana que els pacients han de fer per realitzar la prova.

14. Com a administrador de continguts vull poder introduir al sistema noves mutes i totes les seves dades associades.
15. Com a administrador del continguts vull poder visualitzar i modificar les dades de les diferents mútues.

Requisits de Dades

Ens descriuen quines dades ha de conèixer el sistema, on normalment, parlem de les que el sistema guardarà de forma persistent:

- El sistema ha de conèixer les dades d'una mútua; en particular, ha de saber el nom, adreça de la seu central, telèfon, correu electrònic d'informació i horari d'atenció al públic, i amb quins centres mèdics que realitzen proves està associada.
- El sistema ha de conèixer els centres mèdics on es realitzen proves mèdiques; en concret, ha de saber el nom, adreça, telèfon, correu electrònic d'informació, quines proves mèdiques si realitzen i amb quines mútues treballen.
- El sistema ha de conèixer el tipus d'autoritzacions amb que treballa cada mútua, si van amb volant o no, i les vies en que es poden demanar.
- El sistema ha de conèixer les prescripcions de qualsevol prova mèdica.

2.2.2 Requisits no funcionals

Aquí definirem els requisits que defineixen qualitats esperades del sistema, que es tradueixen a restriccions sobre els conjunt de solucions tals que si aquestes no acaben de satisfer-la, aquesta es donarà com a qualitat no vàlida:

- El sistema s'ha de poder fer servir per persones amb poca a cap coneixement d'aquest tipus d'aplicacions, i amb possibles discapacitats visuals.
- El sistema ha de ser fàcil de entrar i modificar les diferents dades permanents, amb peticions de confirmació de cada canvi i permetre accions de desfer.
- El sistema es podrà fer servir en català i castellà.

3. Model de Casos d'ús

En aquest treball farem servir els casos d'ús com a tècnica de documentació de requisits, la qual destaca entre moltes altres per tenir el suport de UML i que ens permet diversos graus de detall i formalisme, la qual cosa els fa adequats en escenaris molt diversos.

3.1 Actors principals i de suport

Com bé sabem, un *stakeholder* és algú que participa en el contracte de l'aplicació, metres que un actor és algú que hi té comportament. Es a dir, els primers participen en els casos d'ús mitjançant els seus interessos, i els actors són els qui hi participen. Em de tenir clar que un actor no ha de ser només una persona, també ho pot ser una organització o un sistema informàtic: qualsevol cosa que tingui capacitat per interactuar amb el sistema amb un comportament propi. En el nostre cas definim els següents actors principals:

- **Usuari no registrat (UNR)**: podrà fer gestions entorn la realització de proves mèdiques, entrant cada cop la informació personal.
- **Usuari registrat (UR)**: podrà fer gestions entorn la realització de proves mèdiques fent servir la informació del seu perfil, i disposar d'un registre de les proves realitzades.
- **Empleat mútua (EM)**: podrà donar d'alta la seva mútua introduir o editar tota la informació d'aquesta.
- **Empleat centre mèdic (ECM):** podrà donar d'alta el seu centre mèdic i introduir o editar tota la informació d'aquesta.
- **Administrador (A)**: podrà donar d'alta EM i ECM, així com un control general de l'aplicació.

3.2 Diagrama de casos d'ús

A partir de l'estudi de totes els requisits funcionals recopilats a través de les entrevistes al diferents *stakeholders* implicats, ja tenim clares les funcionalitats que, en una primera fase, ha d'oferir la nostra aplicació. Cada funcionalitat és tradueixen a un cas d'ús concret que tot seguit es llisten, amb l'identificador que tindran a partir d'ara, els requisits funcional que han portat a la seva definició, així com l'actor principal que en fa ús.

CU_01: Identificar-se (login) (Requisit 6) (UR, EM, ECM, A)

CU_02: Registrar-se (Requisit 6) (UNR)

CU_03: Consultar reserves (Requisit 6) (UR)

CU_04: Modificar dades personals (Requisit 6) (UR)

CU_05: Sortir (logout) (Requisit 6) (UR, EM, ECM, A)

CU_06: Cercar centre mèdic per prova (Requisit 1, 2 i 3) (UNR, UR)

CU_07: Demanar cita (Requisit 4, 10, 12 i 13) (UNR, UR)

CU_08: Enviar resultats cerca (Requisit 5) (UNR, UR)

CU_09: Demanar autorització (Requisit 7, 8 i 9) (UNR, UR)

CU_10: Enviar autorització: (Requisit 11) (UNR, UR)

CU_11: Donar d'alta una mútua (Requisit 14) (A)

CU_12: Cercar mútua per nom (Requisit 14) (UNR, UR)

CU_13: Modificar dades mútua (Requisit 14) (EM)

CU_14: Esborrar una mútua (Requisit 14) (EM, A)

CU_15: Donar d'alta un centre mèdic (Requisit 15) (A)

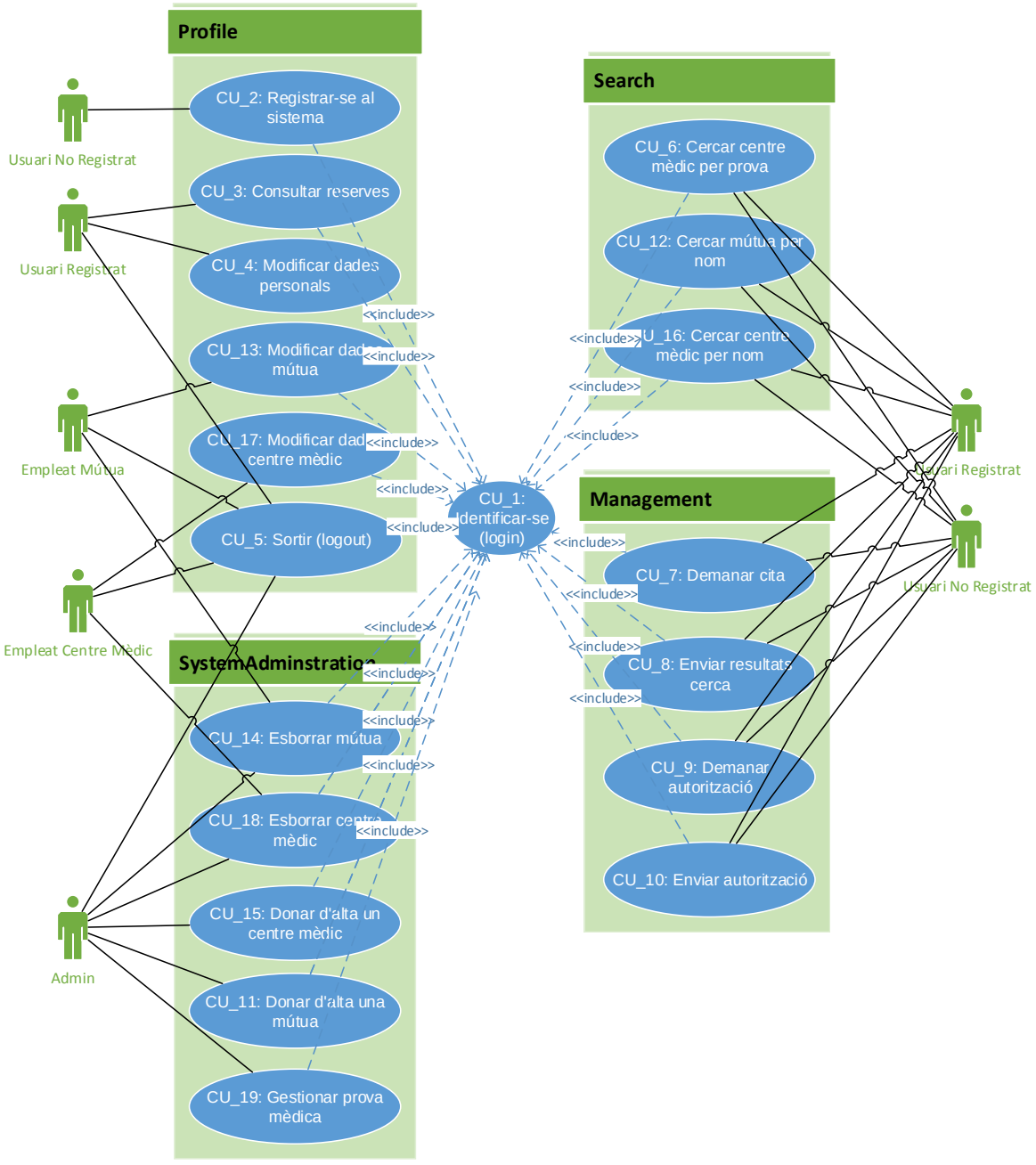
CU_16: Cercar centre mèdic per nom (Requisit 15) (UNR, UR)

CU_17: Modificar dades centre mèdic (Requisit 15) (ECM)

CU_18: Esborrar un centre mèdic (Requisit 15) (EMC, A)

CU_19: Gestionar prova mèdica (A)

Amb aquestes funcionalitats definides, fem el diagrama de casos d'ús segons aquestes. Tots els que estan compartits per l'Usuari registrat i no registrat, tenen la opció en algun moment de fer la funcionalitat de identificar-se al sistema (login), per això està inclòs dins d'aquests, tot i que es pot realitzar sense. S'han posat els actors diverses vegades al diagrama per llegibilitat.



3.3 Especificació de casos d'ús

En aquesta apartat farem una especificació textual dels casos d'ús més rellevants, on ens servirà per veure quina és la informació que proporciona l'usuari al sistema i quina en retorna aquests. Això ens serà de gran ajuda per definir després la interfície gràfica de forma més fàcil. Per facilitar encara més aquesta tasca, en alguns casos d'ús s'ha inclòs una petita descripció de la pantalla que es mostrarà.

CU_01: Identificar-se (login)	
Resum de la funcionalitat:	permet a l'usuari identificar-se al sistema.
Actor principal:	Usuari registrat, empleat mútua, empleat centre mèdic, Administrador.
Precondició:	-
Escenari principal d'èxit	1. El sistema mostra la pantalla "Login". 2. L'usuari introdueix les dades d'identificació. Totes són obligatòries. 3. El sistema comprova que les dades d'identificació i enregistra la identificació de l'usuari a la seva sessió. 4. El cas d'ús finalitza
Extensions	3a. Les dades proporcionades per l'usuari no són correctes. 3a1. El sistema mostra el missatge d'error amb la descripció d'aquests 3a2. Anar al punt 2.
Pantalles	• Login: Dades mostrades: - correu de l'usuari i mot de pas.

CU_02: Registrar-se	
Resum de la funcionalitat:	permet a l'usuari registrar-se al sistema.
Actor principal:	Usuari no registrat.
Precondició:	-
Escenari principal d'èxit	1. L'usuari indica que es vol registra al sistema. 2. El sistema mostra la pantalla "Dades registre nou usuari" 3. L'usuari introdueix les dades sol·licitades. Només són obligatòries el correu i la paraula de pas. 4. El sistema comprova que les dades són correctes i enregistra l'usuari.
Extensions	4a. Les dades introduïdes per l'usuari no són correctes perquè el mot de pas i la seva confirmació no coincideixen, o el correu de l'usuari ja existeix dins el sistema. 4a1. El sistema mostra el formulari "Dades registre nou usuari" amb un missatge d'error indicatiu. 4a2. Anar al pas 3.
Pantalles	• Dades registre nou usuari: Dades sol·licitades: - nom, cognoms, adreça correu electrònic, mútua a les que pertany, número targeta de la mútua, mot de pas, confirmació de mot de pas.

CU_03: Consultar reserves	
Resum de la funcionalitat:	permet a l'usuari veure l'historial de les reserves realitzades mitjançat l'aplicació.
Actor principal:	Usuari Registrat
Precondició:	l'usuari s'ha d'haver identificat al sistema.
Escenari principal d'èxit	1. L'usuari selecciona la opció de configuració personal 2. El sistema mostra la pantalla "Configuració personal" 3. L'usuari selecciona la opció de veure l'historial de reserves 4. El sistema mostra la pantalla "Historial de reserves" 5. L'usuari selecciona una reserva dins la llista mostrada 6. El sistema mostra la pantalla "Detall reserva" 7. El cas d'ús finalitza.
Extensions	6a. El usuari selecciona la opció tornar a veure tot l'historial de reserves 6a1. El sistema torna al punt 4
Pantalles	• Configuració personal: Enllaços mostrats: - Es pot anar a: modificar dades personals, historial de reserves i sortir. • Historial de reserves: Dades mostrades: - Es mostra una llista amb totes les reserves realitzades per l'usuari • Detall reserva: Dades mostrades: - Es mostra una fitxa amb la data de la reserva, el centre.

CU_04: Modificar dades personals	
Resum de la funcionalitat:	permet a l'usuari modificar les seves dades personals
Actor principal:	Usuari Registrat
Precondició:	L'usuari s'ha d'haver identificat al sistema.
Escenari principal d'èxit	1. L'usuari selecciona la opció de configuració personal 2. El sistema mostra la pantalla "Configuració personal" 3. L'usuari selecciona la opció modificar dades personals 4. El sistema mostra la pantalla "Editar usuari" 5. L'usuari edita tots els camps disponibles i els confirma mitjançant un botó 6. El sistema valida els canvis i mostra un missatge que els canvis s'han efectuat de forma correcte. 7. El cas d'ús finalitza.
Extensions	5a. L'usuari cancel·la l'edició 5a1. El sistema torna a la pantalla "Inici"
Pantalles	• Editar usuari: Dades mostrades:

	- Nom, cognoms, mútua a la que pertany, número targeta de la mútua, correu electrònic i mot de pas.

CU_06: Cercar centre mèdic per prova	
Resum de la funcionalitat:	Permet a l'usuari fer una cerca d'un centre a partir de les dades que se li van demanant. En cas d'estar registrat, aquestes seran menors.
Actor principal:	Usuari registrat i usuari no registrat
Precondició:	-
Escenari principal d'èxit	1. L'usuari selecciona una prova mèdica 2. El sistema pregunta si l'usuari pertany a alguna mútua mèdica. 3. L'usuari selecciona afirmativament. 4. El sistema mostra les mútues disponibles. 5. L'usuari selecciona una mútua. 6. El sistema mostra els centres on es pot realitzar la prova adherits a la mútua. 7. L'usuari selecciona un centre concret. 8. El sistema mostra la pantalla "Fitxa centre segons prova", i afegeix el missatge que s'ha de portar sempre la petició del metge per realitzar-la. 9. El cas d'ús finalitza.
Extensions	2a. L'usuari està identificat al sistema i té guardada la mútua a la que pertany. 2a1. Anar al punt 6. 3a. L'usuari selecciona negativament. 3a1. El sistema mostra els centres on es pot realitzar la prova. 3a2. L'usuari selecciona un centre concret. 3a3. El sistema mostra la pantalla "Fitxa segons preu". 3a4. El cas d'ús finalitza. 8a. L'usuari selecciona tornar a la llista de centres trobats. 9a. Anar al punt 6. 8b. L'usuari selecciona tornar a fer una cerca. 8b1. Anar al punt 1.
Pantalles	• Fitxa centre segons prova: Dades mostrades: - Nom, adreça completa, els horàries en que es realitza la prova, si fa falta o no autorització, i les preparacions per realitzar-la. Enllaços: - Dona accés al CU_07, CU_08, CU_09, CU_10 • Fitxa centre segons preu: Dades mostrades: - Nom, adreça completa, els horàries en que es realitza la prova, les preparacions per realitzar-la i el preu. Enllaços: - Dona accés al CU_07, CU_08.

CU_07: Demanar cita	
Resum de la funcionalitat:	Permet a l'usuari demanar una cita al centre mèdic desitjat, i en cas que aquest no disposi d'un sistema de reserves amb una API definida, es mostraran les vies alternatives per realitzar la gestió.
Actor principal:	Usuari registrat i usuari no registrat
Precondició:	L'usuari s'ha de trobar dins la pantalla "Fitxa centre segons prova"
Escenari principal d'èxit	1. L'usuari selecciona la opció de demanar cita. 2. El sistema recupera el calendari d'hores disponibles per aquella prova mitjançant la API del sistema del centre mèdic. 3. El sistema mostra un calendari amb les hores disponibles en verd. 4. L'usuari selecciona la hora desitjada. 5. El sistema mostra la pantalla "Dades reserva cita" 6. L'usuari introdueix les dades sol·licitades. Totes les dades són obligatòries. 7. El sistema mostra la pantalla "Confirmació de cita", i les dades d'aquesta. 8. L'usuari confirma la seva cita. 9. El sistema enregistra la cita al sistema remot del centre mitjançant la seva API, enregistra la cita al propi sistema, i envia per correu electrònic a l'usuari amb les dades d'aquesta. 10. El sistema mostra la pantalla "Cita reservada". 11. El cas d'ús finalitza.
Extensions	2a. El centre mèdic no disposa d'un sistema de reserves amb API definida per connectar-se a través de la aplicació. 2a1. El sistema mostra totes les vies disponibles per fer la reserva d'aquell centre: telefònicament, link a la pàgina web del centre on es realitzen les cites o l'adreça del centre. 2a2. El cas d'ús finalitza 5a. L'usuari està identificat al sistema i té guardades totes les dades necessàries per demanar cita. 5a1. Anar al punt 6. 8a. L'usuari cancel·la la petició de cita. 8a1. Anar a punt 2.
Pantalles	• Dades reserva cita: Dades sol·licitades: - Nom i cognoms, mútua, número targeta i adreça de correu electrònic.

CU_08: Enviar resultats cerca	
Resum de la funcionalitat:	Permet a l'usuari enviar el centre mèdic, els horaris en que es realitza la prova, i la preparació d'aquesta (en cas de tenir-ne) a una direcció de correu especificada.
Actor principal:	Usuari registrat i usuari no registrat
Precondició:	L'usuari s'ha de trobar dins la pantalla "Fitxa centre segons prova"

Escenari principal d'èxit	1. L'usuari selecciona la opció enviar resultats cerca. 2. El sistema mostra la pantalla demanant l'adreça de correu electrònic. 3. L'usuari introdueix l'adreça de correu electrònic. 4. El sistema envia el correu electrònic i mostra la pantalla "Informació enviada" 5. El cas d'ús finalitza.
Extensions	2a. L'usuari es troba identificat al sistema i té guardat una adreça de correu. 2a1. Anar al punt 4.

CU_09: Demanar autorització	
Resum de la funcionalitat:	Permet a l'usuari demanar una autorització a la seva mútua per realitzar una prova mèdica concreta.
Actor principal:	Usuari registrat i usuari no registrat
Precondició:	L'usuari s'ha de trobar dins la pantalla "Fitxa centre segons prova"
Escenari principal d'èxit	1. L'usuari selecciona la opció de demanar una autorització. 2. El sistema mostra el missatge "Es recomana tenir feta la reserva al centre mèdic desitjat". 3. El sistema mostra la pantalla "Autorització segons mútua". 4. L'usuari selecciona la opció correu electrònic. 5. El sistema mostra la pantalla "Dades petició autorització segons mútua" 6. L'usuari introdueix les dades sol·licitades. Totes són obligatòries. 7. El sistema obre l'aplicació de correu predefinida en la màquina de l'usuari amb les dades del formulari dins el cos d'aquest, i la direcció de correu de la mútua com a destinatari. 8. El cas d'ús finalitza.
Extensions	4a. L'usuari desitja fer la petició telefònicament i agafa la informació que necessita i prem tornar. 5a. L'usuari es troba identificat al sistema i té guardades totes del formulari. 5a1. El sistema només demana el número de petició mèdica. 5a2. L'usuari introdueix el número de la petició mèdica. 5a3. Anar al punt 7.
Pantalles	• Autorització segons mútua: Dades mostrades: - Telèfon de petició d'autoritzacions de la mútua. - Documents requerits per aconseguir l'autorització en aquesta mútua Enllaços: - Dona accés a fer la gestió mitjançant correu electrònic. • Dades petició autorització segons mútua: Dades sol·licitades: - Nom i cognoms, número targeta de la mútua, número petició mèdica, número volant mèdic (depenent de la mútua).

CU_10: Enviar autorització	
Resum de la funcionalitat:	Permet a l'usuari que ha rebut un document d'autorització de la seva mútua enviar-lo al centre mèdic on vol realitzar la prova.
Actor principal:	Usuari registrat i usuari no registrat
Precondició:	L'usuari s'ha de trobar dins la pantalla "Fitxa centre segons prova"
Escenari principal d'èxit	1. L'usuari selecciona la opció d'enviar autorització. 2. El sistema mostra la pantalla "Dades enviament autorització a centre mèdic" 3. L'usuari introdueix les dades sol·licitades. Totes són obligatòries. 4. El sistema demana el fitxer on es troba l'autorització dins el sistema de fitxers de la màquina de l'usuari. 5. El sistema obre l'aplicació de correu predefinida en la màquina de l'usuari amb les dades del formulari, el fitxer d'autorització adjunt, i la direcció de correu del centre mèdic com a destinatari. 6. El cas d'ús finalitza.
Extensions	2a. L'usuari es troba identificat al sistema i té guardat una adreça de correu. 2a1. Anar al punt 4.
Pantalles	• Dades enviament autorització a centre mèdic: Dades sol·licitades: - Nom i cognom, mútua, número targeta mútua i prova a realitzar.

CU_11: Donar d'alta una mútua	
Resum de la funcionalitat:	Permet a l'administrador crear una nova mútua dins el sistema i un usuari per gestionar-la, i enviar les dades a aquest mitjançant un correu electrònic.
Actor principal:	Administrador
Precondició:	L'Administrador s'ha d'haver identificat al sistema.
Escenari principal d'èxit	1. L'administrador indica que vol donar d'alta una nova mútua. 2. El sistema mostra el formulari per donar d'alta una nova mútua. 3. L'administrador introdueix les dades sol·licitades. Totes les dades són obligatòries. 4. El sistema registra un nou empleat d'una mútua al sistema associat a la nova mútua, i la direcció de correu proporcionada i una clau d'accés provisional. 5. El sistema obre l'aplicació de correu predefinida en la màquina de l'usuari amb les dades del formulari, i la direcció de correu proporcionada (del empleat de la mútua) com a destinatari. 6. El cas d'ús finalitza.
Extensions	-

CU_12: Cercar mútua per nom	
Resum de la funcionalitat:	permet a un usuari fer una cerca de la fitxa de la mútua segons el nom introduït.
Actor	Usuari registrat i usuari no registrat

principal:	
Precondició:	-
Escenari principal d'èxit	1. L'usuari indica que vol realitzar una cerca de mútua. 2. El sistema mostra la pantalla "Buscador mútua" 3. L'usuari introdueix el nom de la mútua. 4. El sistema mostra la pantalla "Fitxa mútua" 5. El cas d'ús finalitza.
Extensions	4a. El sistema no troba cap mútua amb el nom introduït. 4a1. El sistema mostra un missatge "Mútua no trobada" i torna al punt 2. 4b. El sistema troba més d'una coincidència amb el nom introduït. 4b1. El sistema mostra una llista de les coincidències. 4b2. L'usuari escull el centre desitjat. 4b3. Anem al punt 4.
Pantalles	• Fitxa mútua: Dades mostrades: - Nom, direcció, horaris d'obertura i tancament del centre, telèfon informació, correu electrònic d'informació general.

CU_13: Modificar dades mútua	
Resum de la funcionalitat:	permet a un empleat de la mútua modificar les dades d'aquesta. Aquest actor és únic administrador d'aquesta mútua.
Actor principal:	Empleat Mútua (EM)
Precondició:	l'empleat de la mútua s'ha d'haver identificat al sistema.
Escenari principal d'èxit	1. El EM indica que vol modificar les dades de la mútua. 2. El sistema mostra la pantalla "Mútua detall" amb tots els camps en mode edició. 3. El EM realitza les modificacions desitjades i valida l'operació. 4. El sistema mostra el missatge "Confirmació modificació?" 5. El EM confirma les seves modificacions. 6. El sistema enregistra els canvis i mostra el missatge “Modificacions realitzades" 7. El cas d'ús finalitza.
Extensions	6a. La base de dades no està disponibles (error de servidor) 6a1. El sistema indica a l'usuari que no ha estat possible realitzar les modificacions. 6a2. Anar al punt 2.
Pantalles	• Mútua detall: Dades mostrades: - Nom, direcció, horaris d'obertura i tancament del centre, telèfon informació i de petició autoritzacions, correu electrònic d'informació general i per peticions d'autoritzacions, les proves mèdiques que estan cobertes i en quins centres mèdics, si fa falta autorització per aquestes, i si la mútua treballa amb volants mèdics per les autoritzacions.

CU_14: Esborrar una mútua	
Resum de la funcionalitat:	permet a un Empleat Mútua o Administrador donar de baixa la mútua del sistema i totes les seves dades emmagatzemades.
Actor principal:	Empleat mútua (EM) i Administrador (A).
Precondició:	Els actors s'han d'haver identificat al sistema.
Escenari principal d'èxit	1. El EM indica que vol eliminar la seva mútua. 2. El sistema mostra el missatge "Confirmació eliminar la mútua?" 3. El EM confirma les seves modificacions. 4. El sistema esborra la mútua i el seu EM associat. 5. El sistema mostra el missatge "Eliminació mútua i usuari confirmat" 6. El cas d'ús finalitza.
Extensions	1a. El Administrador indica que vol eliminar una mútua. 1a1. El sistema mostra un llistat amb totes les mútues del sistema. 1a2. El Administrador escull la mútua desitjada. 1a3. Anem al pas 2.

El CU_15, CU_16, CU_17 i CU_18 es consideren equivalents als CU_11, CU_12, CU_13 i CU_14 respectivament, on només canviaran les pantalles, i per això no els hem especificat.

Finalment, el CU_05, no s'ha especificat al considerar que es farà servir la funcionalitat clàssica d'aquesta.

El CU_19 és la gestió (alta, baixa, modificació, llistar) que només un Administrador pot fer d'una prova mèdica. Aquest és l'encarregat de donar-la d'alta al sistema. Després l'empleat d'un centre mèdic podrà indicar quines realitza el seu centre, i el de la mútua, quines cobreixen, i a on.

4. Interfície gràfica d'usuari

Per fer un anàlisi més general de les diferents, agafarem els casos d'ús que representen les funcionalitats principals o més característiques de l'aplicació i farem un mapa navegacional per veure les diferents pantalles i transicions que es poden donar durant l'execució d'aquest. En l'apartat anterior ja hem definit algunes pantalles, que ens ajudaran a definir millor les diferents pantalles.

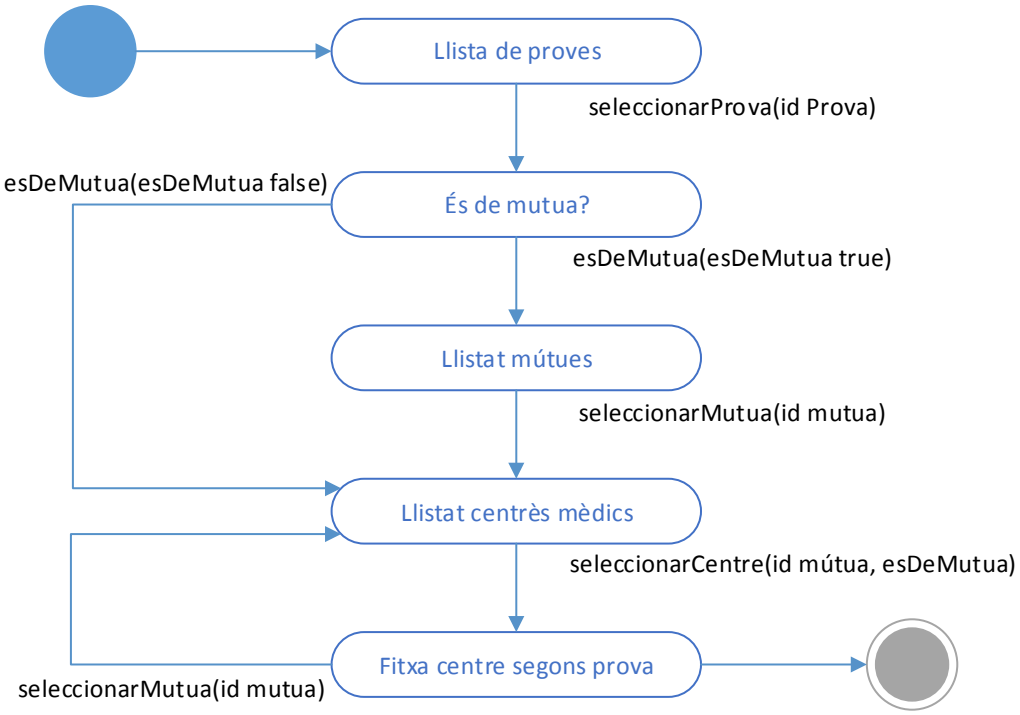
4.1 Casos d'ús essencials i model de pantalles

Anomenem casos d'ús essencials els que ens descriuen la interacció entre actors i sistema de manera independent de la tecnologia i la implementació. Aquests vindrien a ser tots els que hem especificat, però en aquest apartat en triarem els més representatius per fer-ne el mapa navegacional i les pantalles corresponents.

Amb el mapa navegacional tenim un model que ens dóna informació sobre quin és el flux entre pantalles les diferents pantalles que pot anar seguint l'usuari. Així doncs, la finalitat d'aquest mapa és donar una visió general de quines accions es poden fer a cada pantalla, i en quins casos es passa d'una pantalla a una altra.

CU_06: Cercar centre mèdic per prova

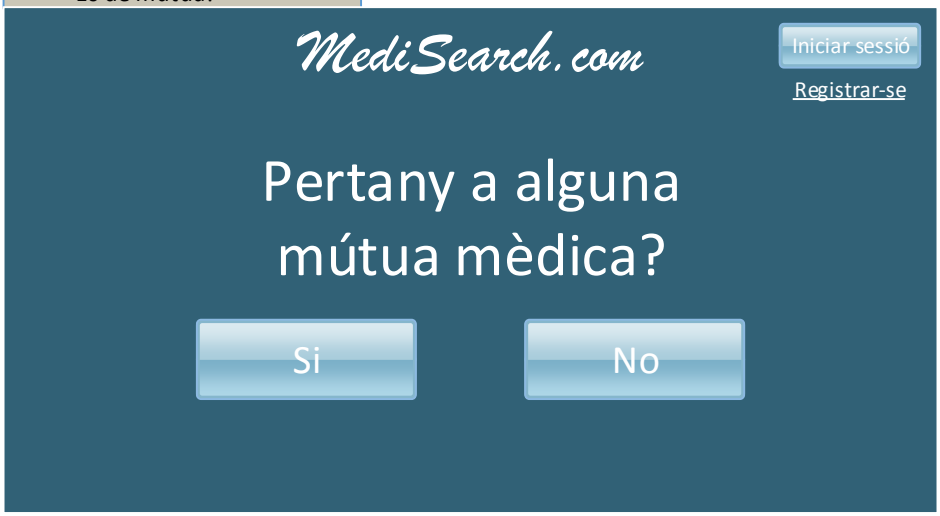
El mapa navegacional d'aquest cas d'ús, amb les seves pantalles mostrades és tal com:



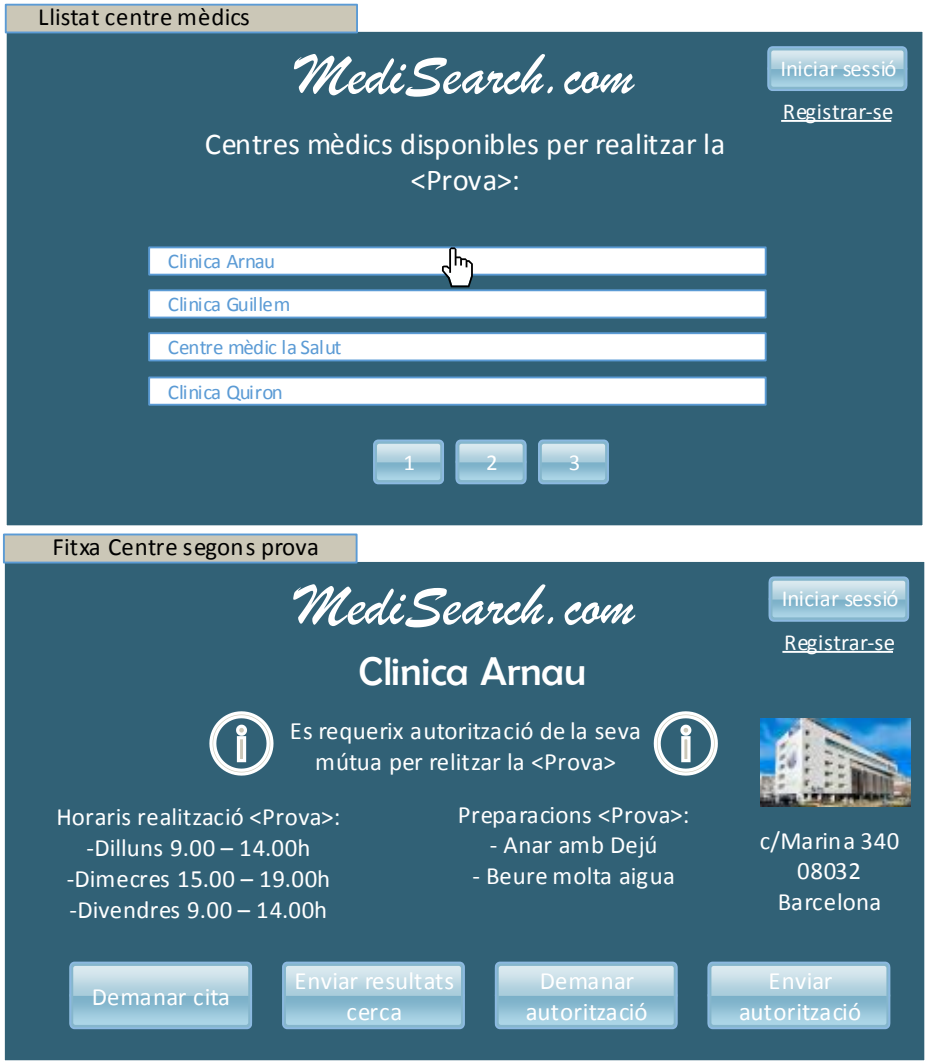
A partir d'aquest mapa podem anar presentant les diferents pantalles:

Llistat de proves

És de mútua?

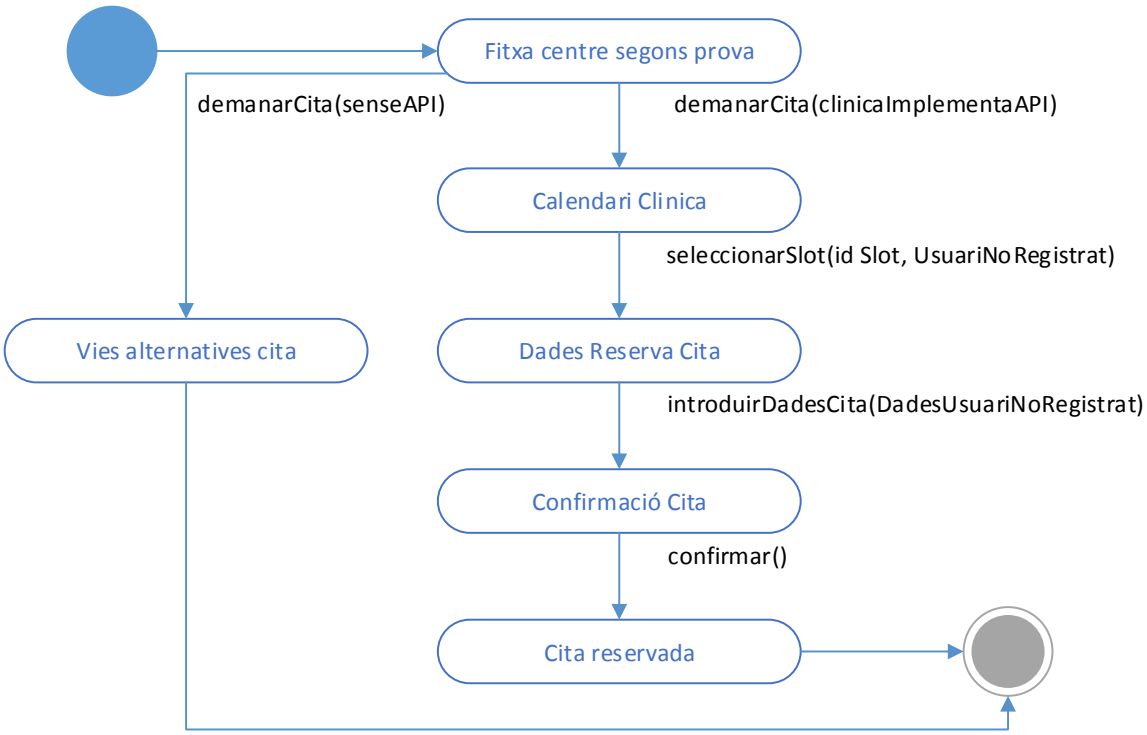


Llistat de mútues



CU_07: Demanar cita

El mapa navegacional d'aquest cas d'ús, amb les seves pantalles mostrades és tal com:



A partir d'aquest mapa podem anar presentant les diferents pantalles:

Dades reserva cita

Confirmació Cita

Cita reservada

4.2 Altres pantalles

Veiem aquí pantalles destacades d'altres casos d'ús:

5. Diagrama estàtic d'anàlisi

En aquest apartat farem una representació dels conceptes del que farem servir en l'aplicació, mitjançant el diagrama estàtic d'anàlisi, també conegut com model conceptual. L'objectiu d'aquest és reflectir allò que el nostre sistema coneix del món real des del punt de vista d'anàlisi i, per tant, no tindrem en compte la tecnologia que farem servir d'implementació.

Es farà servir una diagrama de classes UML en que apareixerà una classe per cada concepte definit del món real que el nostre sistema hagi d'interactuar, i les relacions que apareixen entre aquests.

5.1 Mútues mèdiques i centre mèdics

De les mútues mediques volem guardar la informació bàsica (nom, adreça, telèfons atenció client i peticions autoritzacions, adreces correu electròniques). A part, del centre mèdic en voldrem també la informació bàsica, i les proves mèdiques que realitzen. Aquestes les representem també amb una classe i mitjançant una relació amb els centre mèdics podem saber quins les ofereixen, i fem servir una classe associativa per poder saber-ne quin preu carrega cada centre mèdic quan es oferta sense cobertura d'una mútua.

Les mútues mèdiques tenen acords comercials perquè els seus clients puguin realitzar proves mèdiques als diferents centres mèdics. Aquests acords són per prova, és a dir, una mútua contracta certes proves d'un centre mèdic, no té perquè contractar totes les que realitza el centre mèdic. A més a més, cada mútua demanarà autorització a certes proves que tinguin contractades. D'aquesta manera, s'ha fet servir un tipus de relació ternària per representar les proves ofertes per les mútues realitzades a centres mèdics associats, i una classe associativa per representar si fa falta o no autorització.

5.2 Reserves de proves mèdiques i autoritzacions a la mútua

L'aplicació permet que els usuaris registrats facin reserves per realitzar una prova mèdica a un centre mèdic, i ho representem amb una relació ternària entre aquests tres, i una classe associativa per representar la data i hora que s'ha reservat.

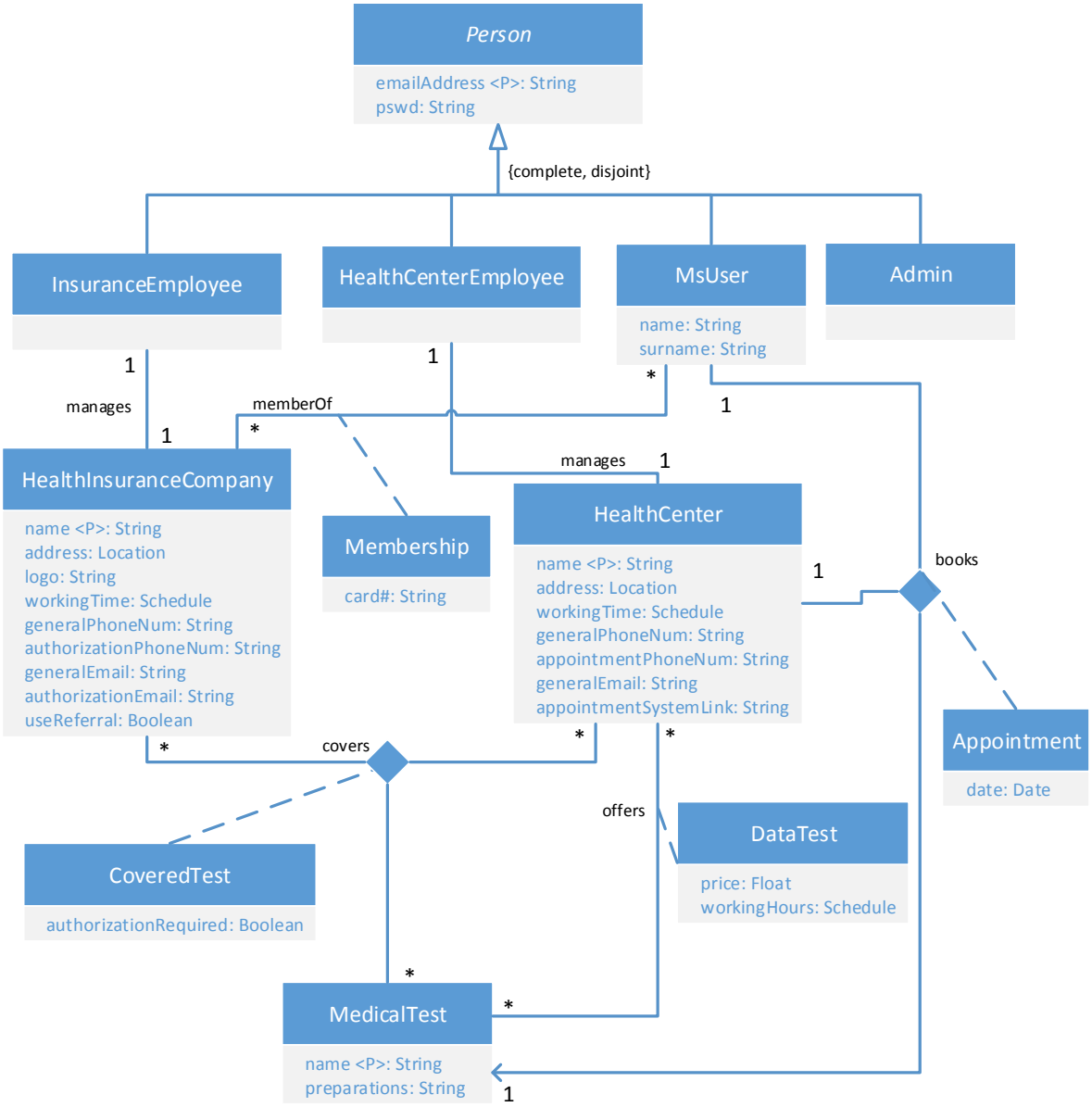
Per altra banda, l'aplicació en permet demanar mitjançant un correu electrònic la autorització a una mútua per realitzar la prova mèdica a un centre mèdica. La informació que s'ha de proporcionar sempre és el nom i cognom, mútua a la que es pertany, número de la targeta d'aquesta, prova a realitzar i centre. Hi ha mútues que aquesta petició ha d'anar en un volant mèdic (un document amb codi de barres i número específic de cada mútua, en forma de talonari). No necessitem crear cap classe específica per l'autorització ja que podem recopilar la informació de les altres, i s'afegirà un atribut a la classe que ens representa les mútues

(HealthInsuranceCompany) per saber si treballen amb volant o no (useReferral), i així demanar a l'usuari aquesta informació, quan apliqui.

Les proves mèdiques seran gestionades només per l'administrador. Després un empleat d'un centre mèdic podrà especificar quines d'aquestes realitza el seu centra, mentre que per el cas de l'empleat d'una mútua, aquest podrà indicar quines proves cobreix la seva mútua, i en quins centres associats es realitzen.

5.3 Model conceptual

Amb el punts referents a com es tracta la informació comentats, el model conceptual resultant és el següent:



6. Disseny relacional de la base de dades

En aquest apartat en centrarem en convertir el model conceptual expressat en llenguatge UML en un esquema lògic per un tipus de base de dades relacional. Podríem utilitzar una eina CASE (*computer-aided software engineering*) per fer la traducció de forma automàtica, però per la dimensió del problema i que els resultats d'aquestes poden no ajustar-se a les condicions de cada cas concret, ho farem de forma manual.

Tot seguit representem el model lògic relacional obtingut, amb el nom de la taula primerament, entre els diferents atributs d'aquesta (columnes), on si el trobem subratllat indica que és clau primària i en negreta, NOT_NULL. Les claus foranes s'indiquen apart:

HealthInsuranceCompany (name, address, logo, workingTime, generalPhoneNum, authorizationPhoneNum, generalEmail, authorizationEmail, useReferral)

HealthCenter (name, address, workingTime, generalPhoneNum, appointmentPhoneNum, generalEmail, appointmentSystemLink)

MedicalTest (name, preparations)

CoveredTest (healthInsuranceCompany, healthCenter, medicalTest, **authorizationRequired**)

{healthInsuranceCompany} is foreign key to HealthInsuranceCompany
{healthCenter} is foreign key to HealthCenter
{medicalTest} is foreign key to MedicalTest

OfferedTest (medicatTestName, healthCenterName, price, workingHours)

{ medicatTestName } is foreign key to MedicalTest
{healthCenterName} is foreign key to HealthCenter

Appointment (userEmail, date, **healthCenter**, **medicalTest**)

{userEmai} is foreign key to MsUser
{ healthCenter} is foreign key to HealthCenter
{ medicalTest } is foreign key to MedicalTest

InsuranceEmployee (email, **pswd**, **healthInsuranceCompany**)

{healthInsuranceCompany} is foreign key to HealthInsuranceCompany

HealthCenterEmployee (email, **pswd**, **healthCenter**)

{healthCenter} is foreign key to HelthCenter

MsUser (email, **pswd**, name, surname)

HealthInsuranceMembers (msuser, healthInsuranceCompany, **cardNumber**)

{msuser} is foreign key to MsUser
{ healthInsuranceCompany } is foreign key to HealthInsuranceCompany
{cardNumber} is unique

Admin (email, **pswd**)

7. Arquitectura del sistema

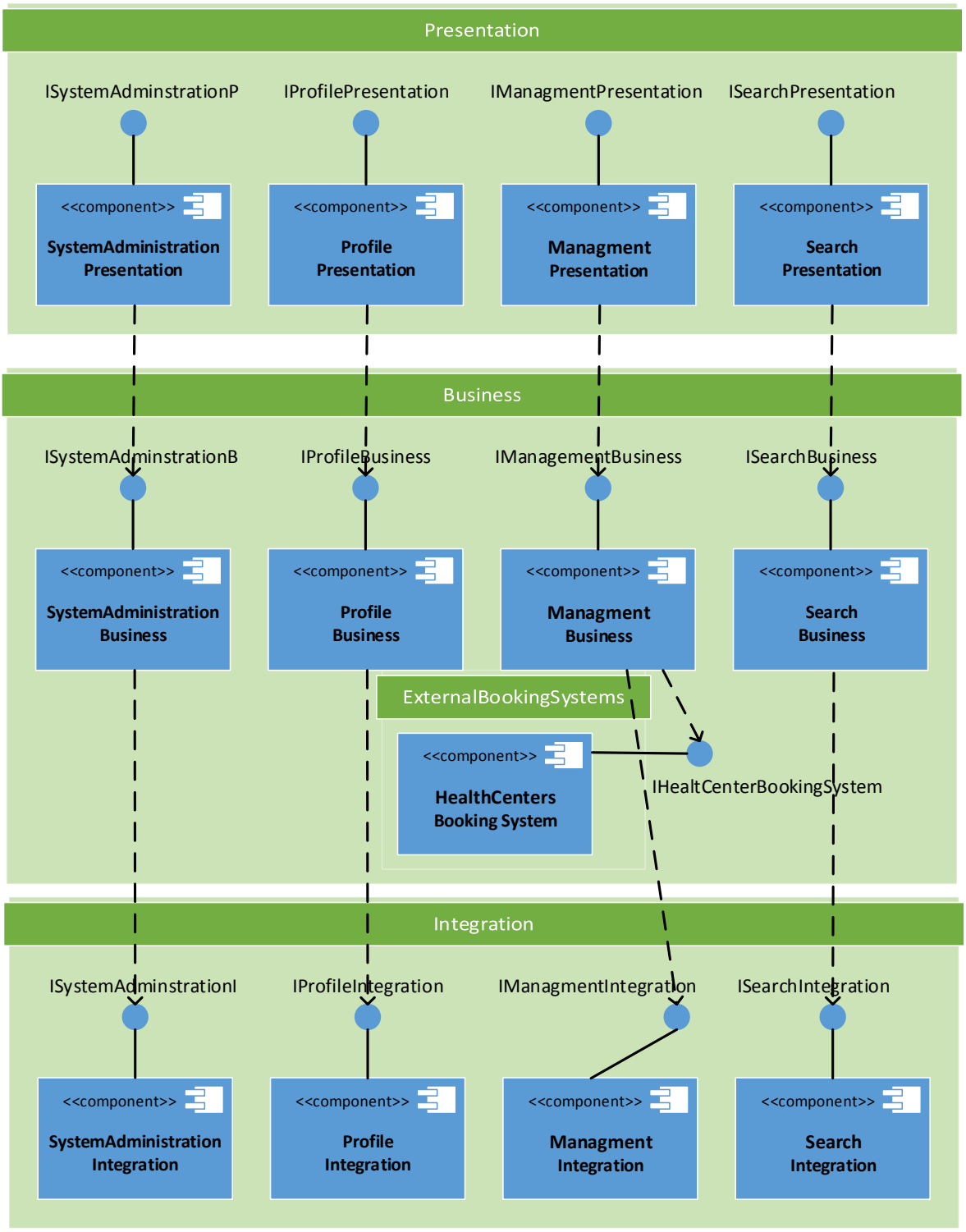
7.1 Arquitectura en tres capes

Per a l'arquitectura general del nostre sistema farem servir el patró d'arquitectura en capes, més concretament ho separarem en tres capes: presentació, lògica de negoci i integració. Mitjançant aquest patró aconseguim treballar en disseny i implementació de cada capa situant-nos en cada cas, en un nivell d'abstracció diferent.

Si apliquem el principi d'inversió de dependències, que ens apunta que el mòduls o classes d'alt nivell no haurien de dependre dels de baix nivell, sinó d'una abstracció, a mida que anem fent el disseny d'una capa, haurem d'especificar l'abstracció que aquesta espera de la capa inferior. Si seguim aquesta filosofia, farem un disseny de les capes on ordre descendent, començant òbviament per la capa de presentació.

Pel diagrama de components hem partit de l'agrupació de casos d'ús que hem fet segons la funcionalitat aportada a l'apartat 2.3 d'aquest document, on es creen 4 blocs amb funcionalitats relacionades, els quals els traduïm a components amb les seves interfícies ben definides.

7.2 Diagrama de components


Presentation
ISystemAdminstrationP
IProfilePresentation
IManagmentPresentation
ISearchPresentation
<<component>>
SystemAdministration Presentation
<<component>>
Profile Presentation
<<component>>
Managment Presentation
<<component>>
Search Presentation
Business
ISystemAdminstrationB
IProfileBusiness
IManagementBusiness
ISearchBusiness
<<component>>
SystemAdministration Business
<<component>>
Profile Business
<<component>>
Managment Business
<<component>>
Search Business
ExternalBookingSystems
<<component>>
HealthCenters Booking System
IHealtCenterBookingSystem
Integration
ISystemAdminstrationI
IProfileIntegration
IManagmentIntegration
ISearchIntegration
<<component>>
SystemAdministration Integration
<<component>>
Profile Integration
<<component>>
Managment Integration
<<component>>
Search Integration

8. Implementació

8.1 Selecció de les tecnologies

Capa de presentació

Per aquesta capa s'ha fet servir Java Server Faces (JSF), una tecnologia per desenvolupar interfícies d'usuari a la banda del servidor, a la vegada que també és un framework web basat en el patró Model – Vista – Controlador, el qual fa servir components reutilitzables.

JSF fa servir els *Facelets* com a sistema de plantilles de presentació, i així com altres possibilitats, tot i que aquest primer ha estat l'escollit per aquest treball.

Tal i com passa molts cops, JSF és l'estàndard definit, però moltes altres tecnologies el poden implementar i crear les seves pròpies llibreries de components. En aquest treball s'ha fet servir una d'aquestes llibreries de components que han aportat molta potència a l'hora del desenvolupament, així com un gran riquesa visual:

Primerfaces és una *suite* de components JSF de codi obert, amb varies extensions. Dins les seves principals característiques, podem destacar:

- Elevat nombre de components com ara *Dialog, AutoComplete,* etc.
- Incorpora funcions de AJAX (Asynchronous JavaScript And XML) basades en l'estàndard de JSF APIs.
- Fàcil configuració basat en un sol fitxer.jar.
- Incorpora una col·lecció de *Themes* per presentacions potents sense necessitat d'implementar CSS.

A més, en aquesta capa s'ha fet servir també s'ha fet servir CDI (Context and Dependeny Injection), un mecanisme per resoldre les dependències entre serveis i Beans dins de l'estàndard JEE, a partir de la versió 6.

Gràcies a aquest podem injectar components en una aplicació en mode *typesafe*, que inclou la possibilitat de seleccionar en temps de desplegament quina implementació volem fer servir. Amb el suport de CDI podem "declarar" qualsevol bean i un punt d'injecció sobre el qual es realitzarà aquesta, quan això sigui necessari.

CDI no és més que la especificació, però no proporciona una implementació pròpia, per tant la s'ha fet servir la implementació que fa **Weld,** un projecte de Red Hat que apareix com la evolució del *core* de JBoss Seam 2.

Capa de negoci

En aquesta capa s'ha fet servir els Enterprise Java Beans, una tecnologia de components de servidor que permet el desenvolupament i el desplegament d'aplicacions empresarials distribuïdes basades en components. Els EJB són components desenvolupats amb Java que compleixen unes especificacions i, per tant, es poden desplegar en qualsevol contenidor d'EJB d'un servidor d'aplicacions compatible amb la plataforma Java EE.

Capa de persistència

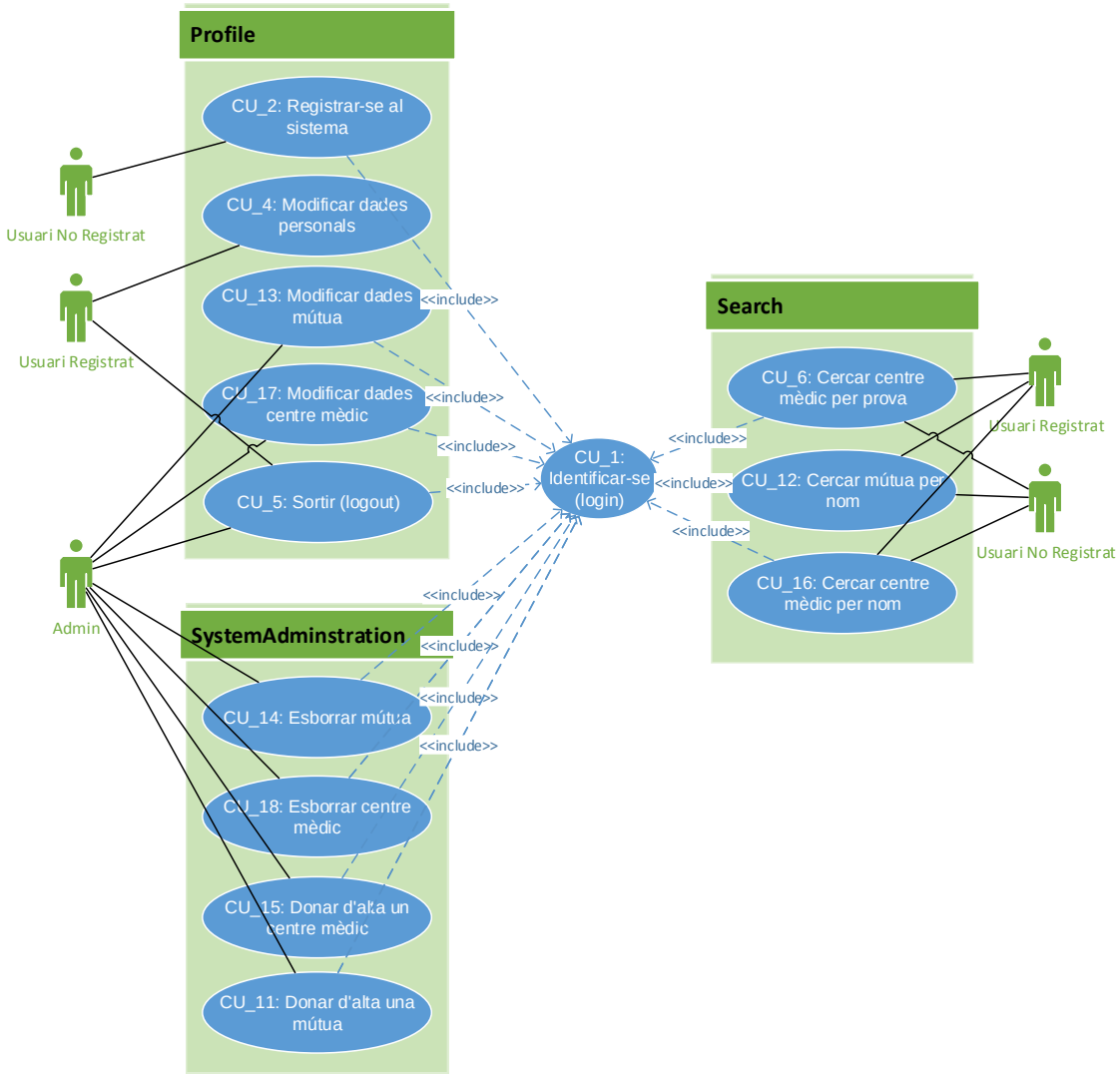
Per la última cap s'ha fet servir la tecnologia i els components que defineix Java EE, el qual promou l'ús de JPA (Java Persistence API). En aquest cas hem fet servir la implementació que Hibernate fa de l'estàndard, amb el gestor de base de dades PostgreSQL.

Servidor d'Aplicacions

En aquest apartat s'ha fet servir WildFly, anteriorment anomenat Jboss, el qual és un servidor d'aplicacions Java EE de codi obert implementat en Java, totalment enfocat a aplicacions Java EE.

8.2 Part implementada de l'abast projectat

Degut al temps disponible per la realització de la part pràctica, la corba d'aprenentatge que tenen els diferents frameworks i d'un abast massa elevat pels recursos i temps disponibles, només s'ha desenvolupat els següents casos d'ús, respecte els plantejats inicialment:



D'aquesta manera s'han reduït a tres el nombre de components i eliminat dos actors.

8.3 Capa d'integració i creació de la base de dades

Els passos seguits per fer la creació i configuració de la base de dades són:

En primer lloc s'ha definit un nou esquema dins els sistema gestor de bases de dades *PgAdmin4* amb el nom de *uoctfg*, amb els usuaris definits prèviament amb *owner* USER i privilegis "ALL".

De cara a la creació de les taules a la BBDD, s'ha deixat aquesta tasca al servidor d'aplicacions Jboss. Així, s'han creat les diferents entitats JPA i configurat el fitxer *persistence.xml* per indicar la tasca de creació, al la següent propietat:

```
 Use only to create the database tables
<property name="hibernate.hbm2ddl.auto" value="create"/>
```

Aquesta només s'ha fet servir el primer cop i després s'ha comentat, ja que sinó cada nova arrencada de l'aplicació ens tornaria a crear les taules, eliminant les dades existents.

Al treballar d'aquesta manera ens assegurem que les entitats JPA definides són correctes, si des de l'administrador de la base de dades *pgAdmin* comprovem que les taules creades coincideixen amb la definició teòrica que s'havia fet d'aquestes.

Per poder treballar amb la capa de persistència, necessitem les llibreries de JPA, les quals en el nostre cas les agafarem de les que porta incorporat el JBoss. Al estar treballant amb Maven, i tenir un servidor d'aplicacions Jboss ja descarregat i configurat, ens hem baixat la dependència del repositori Maven per treballar amb aquest, enlloc del plugin d'aquest.

WildFly: Full Feature Pack » 13.0.0.Final

WildFly: Full Feature Pack

8.3.1 Claus primàries compostes

En el nostre disseny físic de la base de dades s'han creat algunes relacions amb claus primàries compostes. Per tractar aquesta casuística en JPA és fa servir una classe auxiliar on definirem els atributs que formen la clau primària. Veiem amb un exemple com es defineixen aquesta casuística en JPA, amb la següent taula:

OfferedTest (medicatTestName, healthCenterName, price, workingHours)

{medicatTestName} is foreign key to MedicalTest
{healthCenterName} is foreign key to HealthCenter

Primerament, es crea una classe que contindrà els dos atributs que formen la clau primària:

```
@Embeddable
public class OfferedTestId implements Serializable {

    private static final long serialVersionUID = 1L;

    private String medicalTestName;
    private String healthCenterName;
```

Com podem veure, ho tractem amb l'anotació @Embeddable i definim els tipus dels atributs com a cadenes, tot i que en aquest cas, també son clau forana a una relació. Fet això, ja podem definir la entitat JPA que ens definirà la taula que volem mapejar amb la nova classe com a paràmetre i definida com clau primària (mitjançant l'anotació @Id) de l'entitat i els dos atributs restants:

```
@Entity
@Table(name="uoctfg.offeredTest")
public class OfferedTest implements Serializable {


    private static final long serialVersionUID = 1L;

    private OfferedTestId offeredTestId;
    private Float price;
    private Schedule workingHours;

    public OfferedTest() {
        super();
    }
    @Id
    public OfferedTestId getOfferedTestId() {
        return offeredTestId;
    }
```

Finalment, ens queda definir les claus foranes, les quals formen part de la clau primària. Això ho hem definit des de les entitats on es fa la referència, marcant amb anotacions @OneToMany i fent coincidir el nom definit dins l'anotació @JoinColumn amb el del atribut definit dins la classe que forma la clau primària. Per exemple, a la entitat HealthCenter em definit la següent relació:

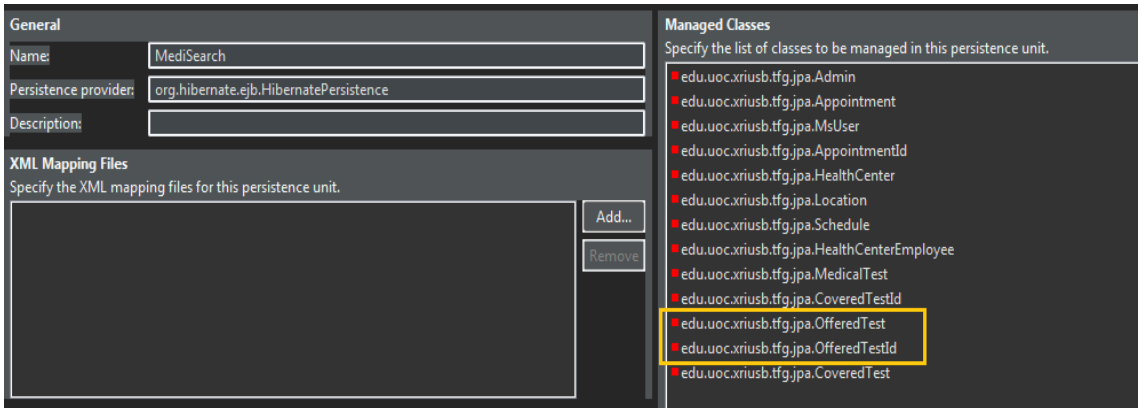
```
@OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY )
@JoinColumn(name = "healthCenterName", foreignKey = @ForeignKey(name = "healthCenter_FK"))
public Collection<OfferedTest> getOfferedTests() {
    return offeredTests;
}

public void setOfferedTests(Collection<OfferedTest> offeredTests) {
    this.offeredTests = offeredTests;
}
```

D'aquesta manera, dins la taula OfferedTest es crearà una clau forana a HealthCenter amb el nom especificat. Si indiquem que sigui el servidor d'aplicacions l'encarregat de la creació de les taules dins el sistema gestor de la base de dades, veiem que el resultat és l'esperat:

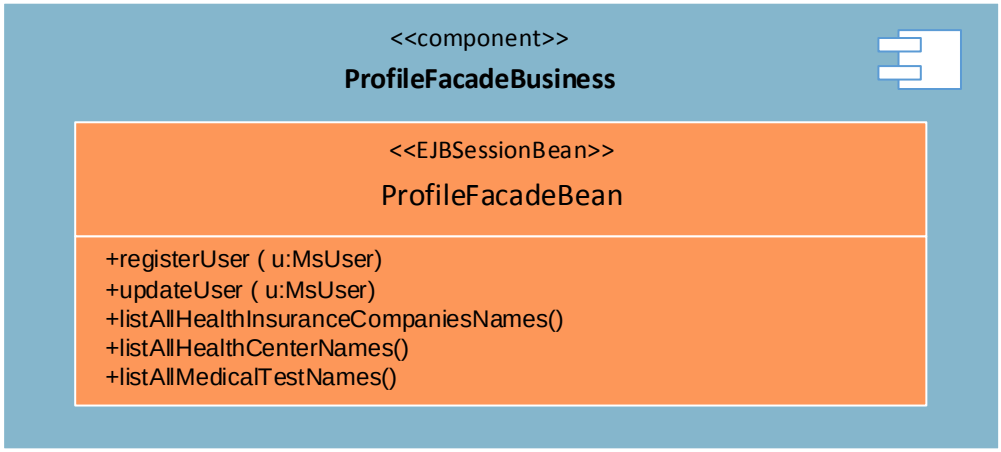
```
CREATE TABLE uoctfg.offeredtest
(
    healthcentername character varying(255) COLLATE pg_catalog."default" NOT NULL,
    medicaltestname character varying(255) COLLATE pg_catalog."default" NOT NULL,
    price real,
    workinghours bytea,
    CONSTRAINT offeredtest_pkey PRIMARY KEY (healthcentername, medicaltestname),
    CONSTRAINT healthcenter_fk FOREIGN KEY (healthcentername)
        REFERENCES uoctfg.healthcenter (name) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT medicaltest_fk FOREIGN KEY (medicaltestname)
        REFERENCES uoctfg.medicaltest (name) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

A més a més, hem de tenir en compte que per no tenir problemes al servidor d'aplicacions amb les noves classes generades per definir la clau primària composta, aquestes també s'han de mapejar dins els fitxer persistence.xml:

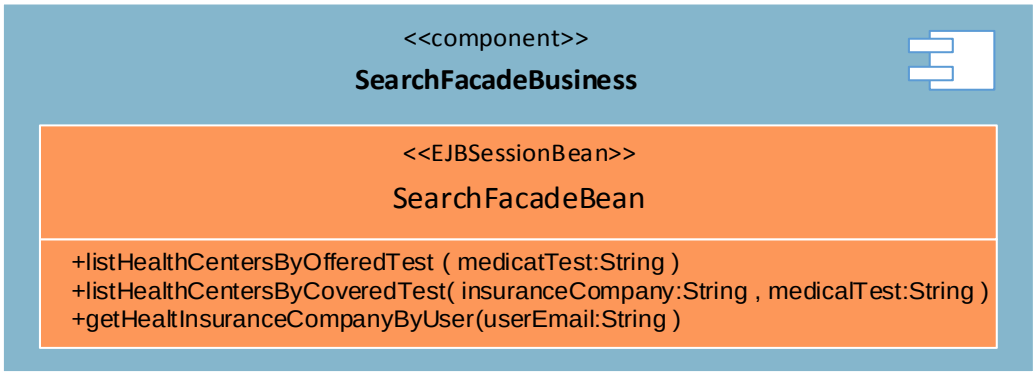


8.4 Capa de negoci

Per la implantació d'aqueta capa s'ha fet uns distribució en els tres components implementats:



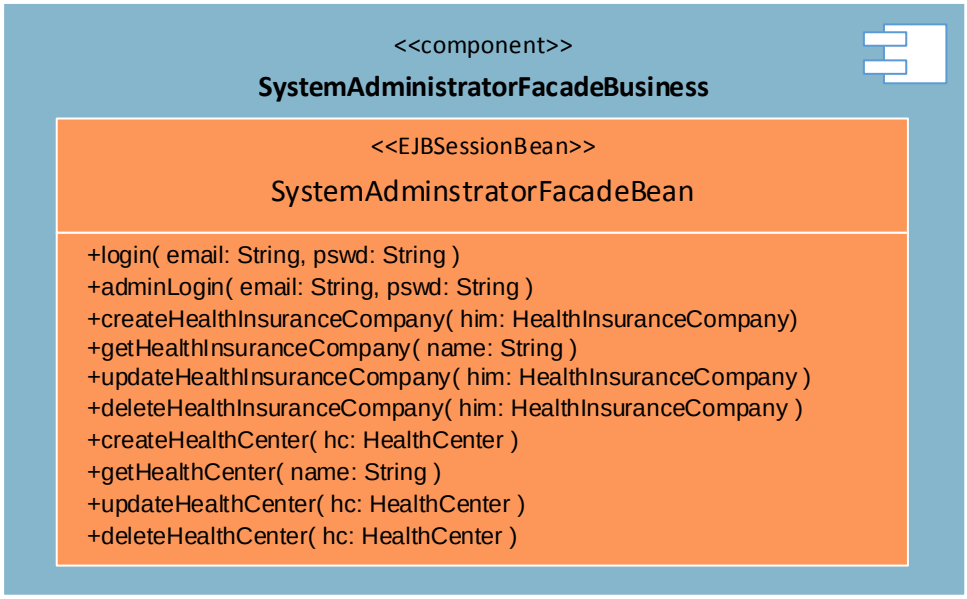
Aquest EJB de sessió i sense estat és l'encarregat de d'implementar el registre i l'actualització de les dades dels usuaris que vulguin tenir les seves dades al sistema. A més, incorporà també els mètodes necessaris perquè l'usuari (registrat o no) tingui accés de lectura a totes els centre mèdics, proves i asseguradores.



Aquest EJB incorpora tota la lògica de negoci del cercador de proves mèdiques de l'aplicació. D'aquesta manera podem cercar quins centres ofereixen una certa prova (*listHealthCentersByOfferedTest*), quins centres mèdics ofereixen proves segons si aquestes estan cobertes per una asseguradora (*listHealthCentersByCoveredTest*). Com podem veure tot seguit, l'obtenció d'aquesta llista és basa amb una consulta a la BBDD on es creuen la taula dels centres mèdics, la qual conté les proves que ofereixen, i la taula de proves mèdiques cobertes (*CoveredTest*), la qual conté les relacions entre les proves mèdiques, l'asseguradora que la cobreix, i el centre mèdic on es pot realitzar.

```
public List<HealthCenter> listHealthCentersByCoveredTest(String insuranceCompany, String medicalTest) {
    @SuppressWarnings("unchecked")
    List<HealthCenter> healthCentersByCovered = em.createQuery(
        "select h from HealthCenter h, CoveredTest c where h.name = c.coveredTestId.healthCenterName and "
        + "c.coveredTestId.healthInsuranceCompanyName=:icomp and "
        + "c.coveredTestId.medicalTestName=:mtest")
    .setParameter("icomp", insuranceCompany)
    .setParameter("mtest", medicalTest)
    .getResultList();
    return healthCentersByCovered;
}
```

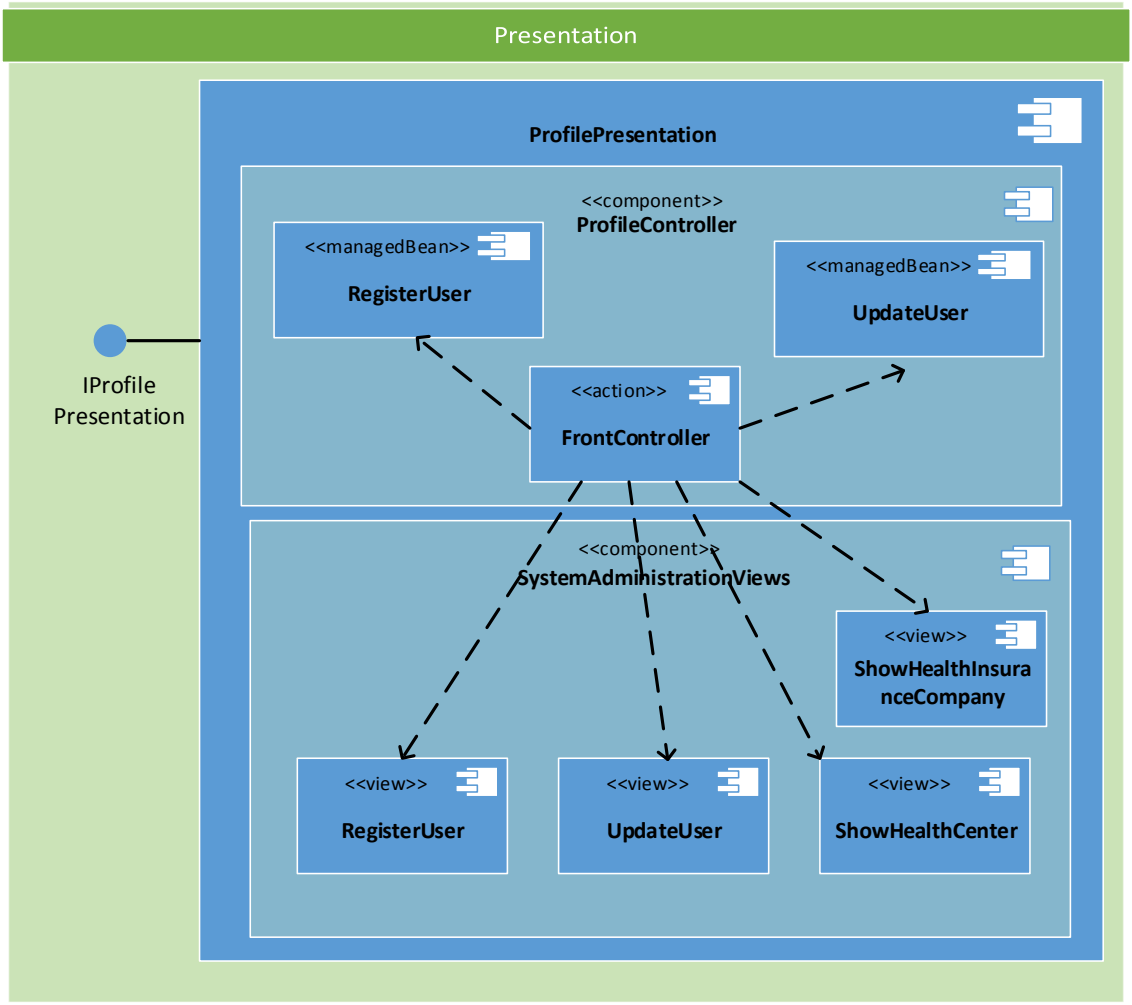
A més incorpora una funció auxiliar que ens retorna la asseguradora del usuari, la qual ens servirà per quan l'usuari hagi fet el *login* corresponent i faci una cerca, ens retornarà l'asseguradora d'aquest i així agilitzar la cerca. La implementació



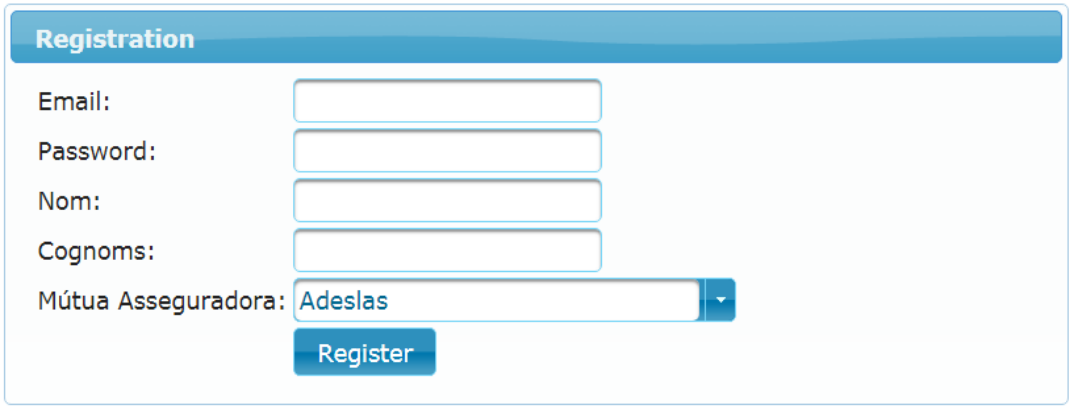
Aquest últim EJB és l'encarregat de les tasques de *Login*, tant d'un usuari com d'un administrador, i tota la gestió que l'administrador pot realitzar sobre els centres mèdics i les asseguradores, que va des de la creació, modificació i eliminació. A més incorpora una funció auxiliar per cada entitat, que ens la retorna segons el nom que li passem, per realitzar les diferents tasques citades.

8.5 Capa de presentació

Aquesta capa també la trobem dividia en els tres components que la formen, que tot seguit els veiem de forma separada:

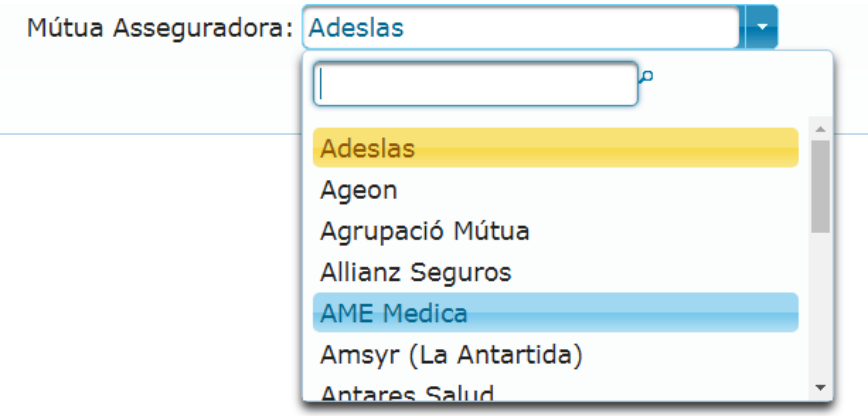


En aquest component bàsicament trobem les funcionalitats de registre d'un usuari, la modificació de les dades d'aquest i el mostrar informació dels centres mèdics o de les asseguradores. Veiem amb una mica més de detall com s'ha tractat el CU_02, registre d'un usuari, on s'ha aprofitat les avantatges que la llibreria de components Primefaces ens dona. En primer lloc veiem la vista:



Aquest està composta per un formulari amb 5 camps, on l'últim és un desplegable on podem seleccionar una opció (en aquest cas

l'asseguradora que pertany l'usuari), el qual també incorpora un cercador, que pertany a un component *selectOneMenu* de Primefaces.

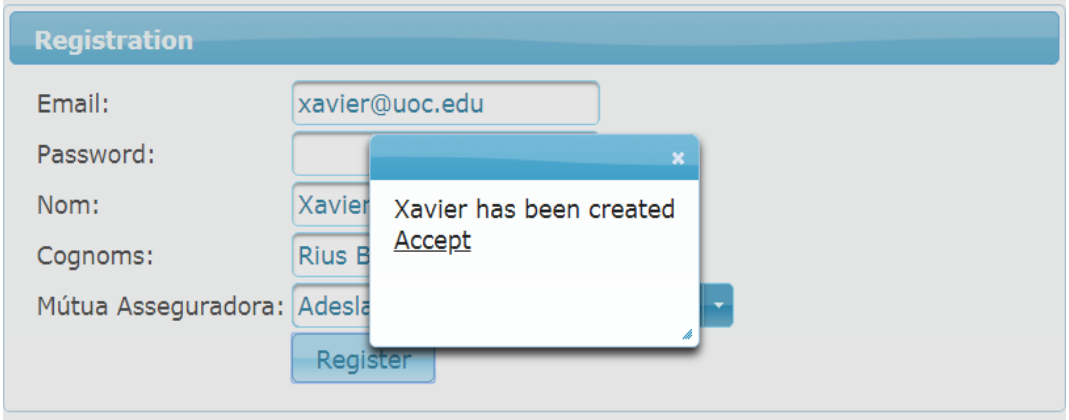


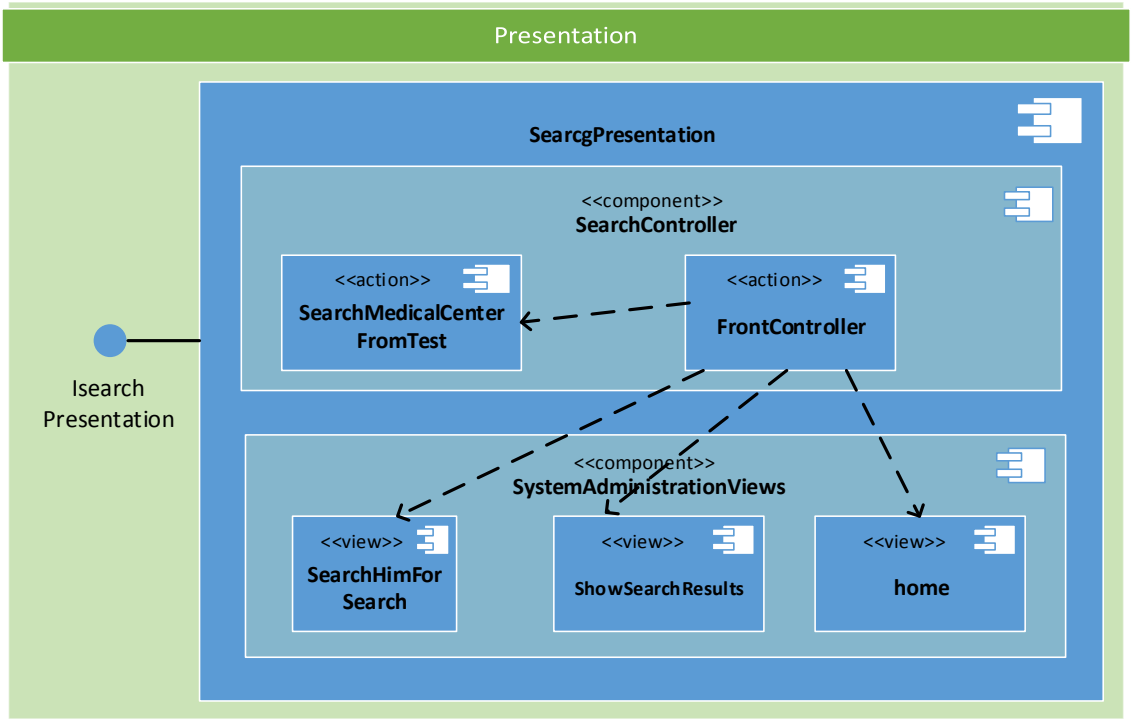
Un cop tenim les dades llestes del formulari, fem la crida al mètode del *managedBean* corresponent, el qual farà les crides a la capa d'integració, i si tot es correcte, fa la crida a un diàleg de la vista, per mostrar per pantalla que tot ha anat correcte. Així, mitjançant Primefaces que es capaç de recuperar el context des del Bean, si executem el següent codi dins aquest:

```
current.executeScript("PF('userCreated').show();");
```

Llavors, si tenim definit el *dialog* corresponent a la vista (fitxer .xhtml), aquest serà executat i en el nostre cas mostrarem el missatge desitjat.

```
<p:dialog widgetVar="userCreated" modal="true" height="100">
    <h:outputText value="#{registerUser.user.name} has been created"/><br/>
    <p:link outcome="home" value="Accept" shape="rect"/>
</p:dialog>
```

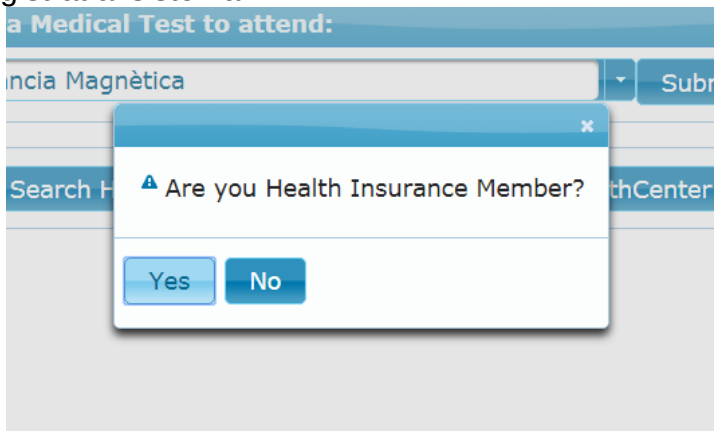




En aquest component trobem la presentació de les funcionalitats de cerca dels centres mèdics segons la prova mèdica que volem. Només es treballa amb un sol *managedBean* que incorpora totes les funcionalitats necessàries. També podem observar que trobem la vista *Home* de l'aplicació, la qual està pensada per ser porta d'entrada directe al cercador:

En aquesta veiem que tenim accés al cercador, així com si volem consultar la informació de tant dels centres mèdics com de les asseguradores. Si entrem en detall en el cas d'ús CU_06, podem començar per escollir la prova mèdica que volem realitzar. Un cop seleccionada un altre cop amb un component Primefaces

selectOneMenu, aquest executarà un diàleg per pantalla per preguntar-nos si som o no d'una asseguradora, sempre i quan no siguem un usuari que està registrat al sistema.



Aquest diàleg, segons la interacció de l'usuari ens redirigirà cap a mostrar els resultats o a l'elecció d'una asseguradora, cosa que aconseguim a través de l'acció AJAX definida a cada botó.

```
<p:confirmDialog message="Are you Health Insurance Member?" widgetVar="cd">
    <p:commandButton value="Yes" action="#{searchMedicalCenterFromTest.hasInsurance()}"
    update="form" oncomplete="PF('cd').hide();"/>
    <p:commandButton value="No" action="#{searchMedicalCenterFromTest.dontHasInsurance()}"
    update="form" oncomplete="PF('cd').hide();" />
</p:confirmDialog>
```

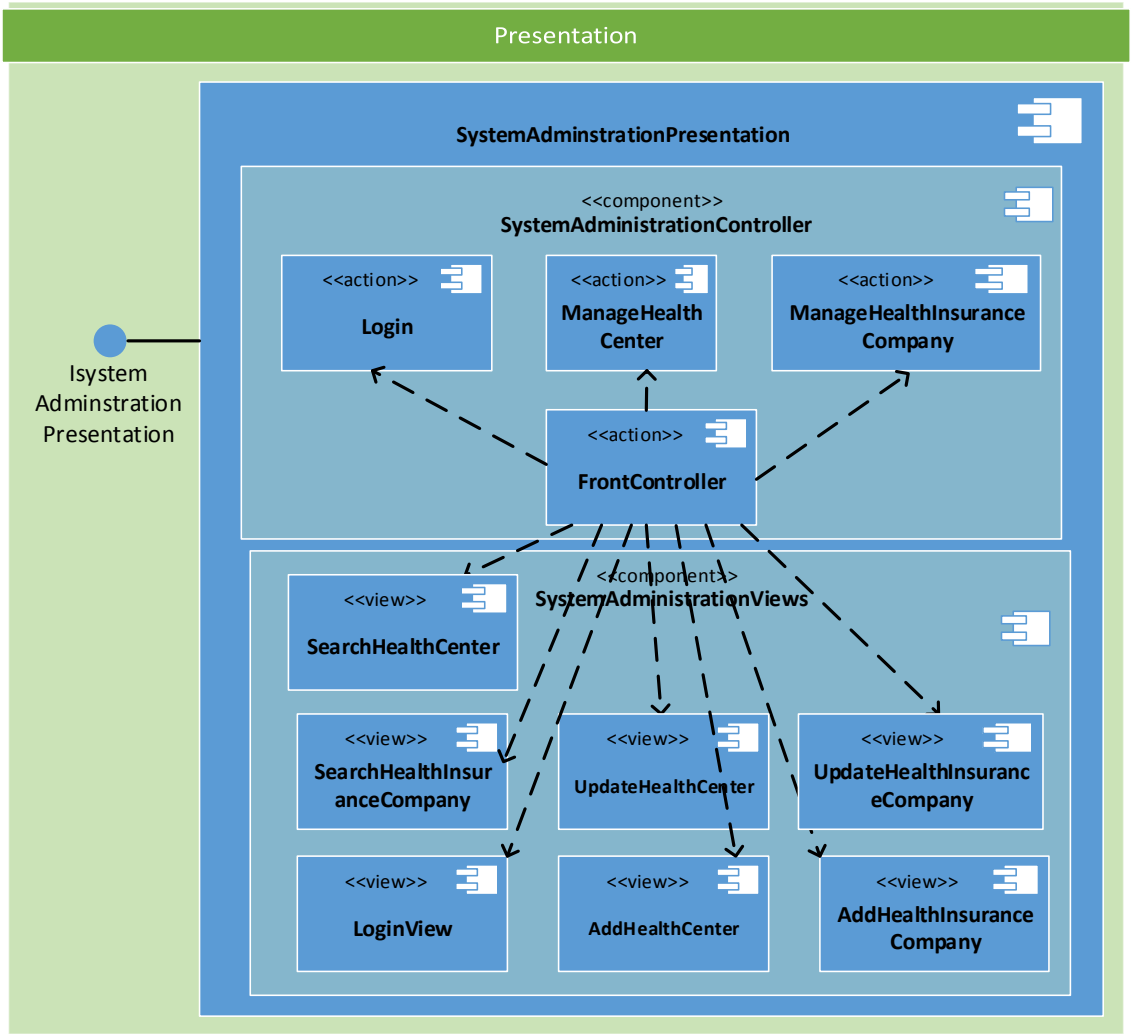
Un cop arribem als resultats, obtenim una llista amb tots els centres mèdics on podem realitzar la prova, on s'ha posat un accés per poder veure els detalls d'aquest només pitjat sobre aquest:

El cas a destacar es quan estem fent la cerca del centre mèdic desitjat mitjançant un usuari que està registrat al sistema. En aquest cas necessitem recuperar la instància del Bean on tenim emmagatzemat les

dades d'aquest, cosa que ho aconseguim gràcies a fer servir la implementació de la injecció de dependències CDI:

```
public List<HealthCenter> getHealthCentersByCoveredTestInsuranced(){
    BeanManager bm = CDI.current().getBeanManager();
    Bean<Login> bean = (Bean<Login>) bm.getBeans(Login.class).iterator().next();
    CreationalContext<Login> ctx = bm.createCreationalContext(bean);
    Login currentUser = (Login) bm.getReference(bean, Login.class, ctx);
    himFromUser = searchFacadeBean.getHealtInsuranceCompanyByUser(currentUser.getEmailAddress());
    return searchFacadeBean.listHealthCentersByCoveredTest(himFromUser.get(0), mtToAttend);
}
```

En aquest mètode obtenim la referència a aquest Bean i d'aquesta manera poder llegir-ne el email, el qual serà el que passarem a la capa de negoci perquè ens retorni els centres mèdics on aquest usuari registrat el qual és membre d'una certa asseguradora, pot realitza la prova mèdica seleccionada.



En aquest darrer component trobem totes les funcions de les que els administradors tenen accés. Primerament, quan ens identifiquem com a administrador del sistema, a la pàgina principal ens desapareix les opcions de cerca i ens apareixen funcionalitats per la gestió dels centres mèdics i asseguradores. Aquesta funcionalitat la aconseguim gràcies

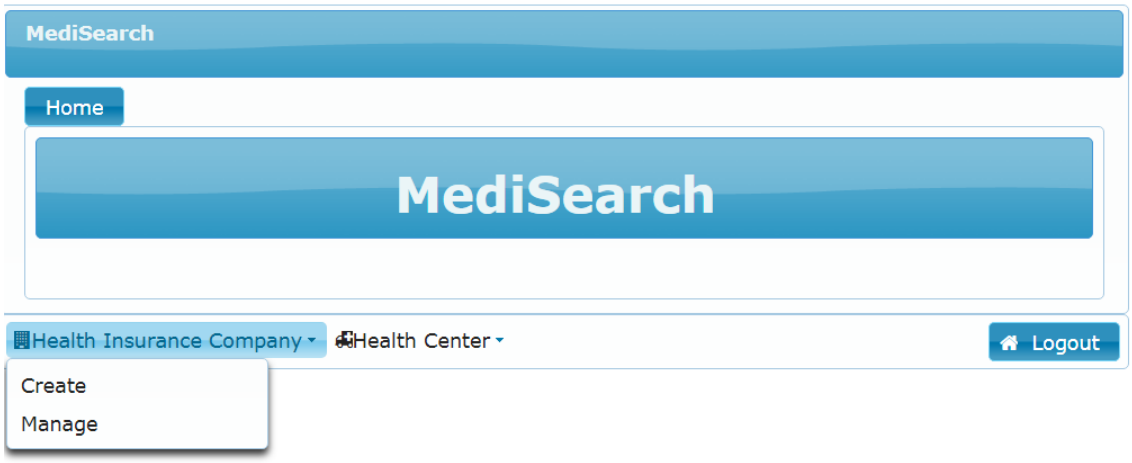
CDI, definit dins el Bean on tindrem la funció que fa la identificació del administrador, es crea un mètode que ens retornarà un booleà per saber si tenim o no aquest actor identificat.

```
public boolean isAdminLoggedIn() {
    return currentAdmin != null;
}
```

Llavors, des de qualsevol vista i amb el paràmetre *rendered* que incorporen els components, el podrem fer servir per mostrar aquestes funcionalitats avançades.

```
<p:menubar rendered="#{login.adminLoggedIn}">
    <p:submenu label="Health Insurance Company" icon="fa fa-building">
        <p:menuitem value="Create" outcome="AddHealthInsuranceCompany"/>
        <p:menuitem value="Manage" outcome="SearchHealthInsuranceCompany"/>
    </p:submenu>
    <p:submenu label="Health Center" icon="fa fa-ambulance">
        <p:menuitem value="Create" outcome="AddHealthCenter"/>
        <p:menuitem value="Manage" outcome="SearchHealthCenter"/>
    </p:submenu>
    <f:facet name="options">
        <p:commandButton value="Logout" action="#{login.logout}"
        icon="fa fa-home" onclick="PF('dlg3').show();" />
    </f:facet>
</p:menubar>
```

I aconseguirem el següent menú:



Dins aquest destaquem els components de Primefaces que incorporen funcionalitats AJAX (Asynchronous JavaScript And XML), on per exemple, per editar un fila d'una taula, esperant l'execució d'un event, del qual podrem recuperar l'obejcte (en el nostre cas pot ser un centre

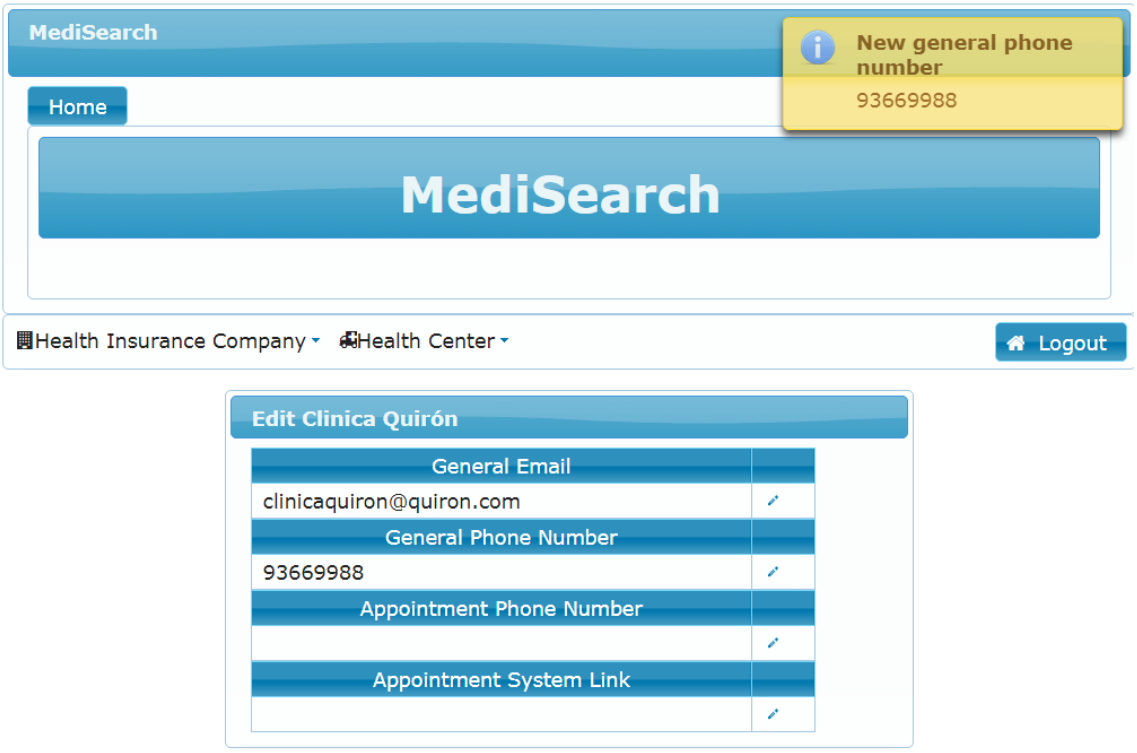
mèdic) i en farem la crida a la capa de negoci corresponent. Llavors en una vista trobem:

```
<p:ajax event="rowEdit" listener="#{manageHealthCenter.onRowEditGeneralPhoneNum}"
update=":form:msgs" />
```

Amb això veiem que amb aquesta etiqueta, quan es dona un esdeveniment AJAX, cridem la lògica del Bean corresponent:

```
public void onRowEditGeneralPhoneNum(RowEditEvent event) {
    healthCenter.setGeneralPhoneNum(
            ((HealthCenter) event.getObject()).getGeneralPhoneNum());
    systemAdministratorFacadeBean.updateHealthCenter(healthCenter);
    FacesMessage msg = new FacesMessage("New general phone number ",
            ((HealthCenter) event.getObject()).getGeneralPhoneNum());
    FacesContext.getCurrentInstance().addMessage(null, msg);
}
```

Dins aquest es recuperen les dades de l'objecte en qüestió i es fa la crida a la capa de negoci, en aquest cas per actualitzar el número de telèfon general d'un centre mèdic.



9. Conclusions

Aquest treball ha estat una gran oportunitat per realitzar el desenvolupament d'una aplicació web, des de la seva idea inicial fins a la seva implementació amb tecnologia Java EE. Durant tot aquest procés s'ha posat en pràctica molts dels conceptes adquirits durant la realització del Grau d'Enginyeria Informàtica, des de l'adquisició del màxim d'informació dels *stakeholders* per definir de forma clara i precisa el objectius del treball, fins a la implementació mitjançant *frameworks* Java. Segurament d'aquest últim punt és on s'ha aprés una de les lliçons més important de tot el treball (la qual també ha afectat al resultat final d'aquest), i es en la dificultat de planificar la fase de desenvolupament del programari.

Durant aquesta última fase han sorgit una gran quantitat d'imprevistos i estancaments, molts cops deguts a la poca experiència en els *frameworks* o tecnologies amb les que es treballaven, que han fet que l'abast assolit final s'hagi reduït del projectat inicialment. Exemples d'aquests podem citar com la configuració inicial del projecte, planificada en dos dies, s'ha acabat traduït en casi un setmana, o molts cops durant la implementació de la capa de presentació, en les vistes, no s'aconseguia el funcionament esperat, i han fet perdre dies sencers per configurar de forma correcte el comportament d'un diàleg amb l'usuari.

Per tant, podem afirmar que no s'han assolit els objectius plantejats inicialment, al no haver pogut realitzar totes les funcionalitats projectades de l'aplicació al programari. La causa d'això la podem trobar en el problema comentat al paràgraf anterior, o potser en un abast del projecte inicial massa gran, comparat amb els recursos disponibles i el temps.

El seguiment que s'ha fet de la planificació inicial ha estat prou encertat amb excepció de la part de la implementació, on els terminis marcat han ajudat a portar el treball al dia i no acumular la feina tota al final. Tot i això, la metodologia no podem dir que hagi estat del tot correcta en la part de la implantació, ja que afrontar una *framework* des de zero, s'ha comprovat com la corba d'aprenentatge és molt més elevada del que s'havia previst inicialment. D'aquesta manera i per entregar un programari que fes la funcionalitat principal, es va haver de retallar el nombre de casso d'ús implementats.

Finalment, aquest treball ens deixa un gran nombre de línies de futur, les quals són principalment la implantació de totes aquestes funcionalitats projectades pendents, així com treure el màxim de partit als *frameworks* amb els que s'ha treballat, ja que no s'ha pogut acabar de fer servir totes les possibilitats que ens donen, per raons de temps i falta d'experiència.

10. Glossari

Framework: conjunt de classes predefinides que hem d'especialitzar i/o instanciar per a implementar una aplicació o subsistema estalviant temps i errors. El bastiment ens ofereix una funcionalitat genèrica ja implementada i provada que ens permet centrar-nos en allò específic de la nostra aplicació.

BBDD (Base de dades): conjunt de dades que pertanyen a un mateix context i s'emmagatzemen sistemàticament per el seu posterior ús.

Cas d'ús: entitat que recull el contracte entre el sistema i els *stakeholders* mitjançant la descripció del comportament observable del sistema.

Stakeholder: qualsevol persona o organització interessada, afectada o implicada en el funcionament del programari que es desenvolupa.

11. Bibliografia

PRADEL MIQUEL, J. RAYA MARTOS, J. (2016) Enginyeria del Programari. Material Docent de la UOC. Barcelona Oberta UOC Publishing, SL.

PRADEL MIQUEL, J. RAYA MARTOS, J. (2016) Enginyeria de Requisits. Material Docent de la UOC. Barcelona Oberta UOC Publishing, SL.

BURGUÉS ILLA, X. Disseny lògic de bases de dades. Material Docent de la UOC. Barcelona Oberta UOC Publishing, SL.

CAMPS RIBA, JM. Java EE. Una plataforma de components distribuïda. Material Docent de la UOC. Barcelona Oberta UOC Publishing, SL.

Çağatay Çivici. Primefaces User Guide 6.0. [en línia] https://www.primefaces.org/documentation/

KING, G. MUIR, P. HARTINGER, J. KOUBA, M. ALLEN, D. ALLEN D. Weld 3.1.0.Beta1 – CDI Reference Implementation**.** CDI: Contexts and Dependency Injection for the Java EE platform. [en línia] http://weld.cdi-spec.org/documentation/