Desenvolupament d'un software de gestió d'instal·lacions esportives amb Spring Roo

Jaume Micó Ferri
Grau d'Enginyeria Informàtica
Java EE

Albert Grau Perisé

12 de Gener de 2017

FITXA DEL TREBALL FINAL

Títol del treball:	Desenvolupament d'un software de gestió d'instal·lacions esportives amb Spring Roo
Nom de l'autor:	*Jaume Micó Ferri*
Nom del consultor/a:	*Albert Grau Perisé*
Nom del PRA:	*Albert Grau Perisé*
Data de lliurament (mm/aaaa):	*01/2017*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*TFG - Java EE*
Idioma del treball:	*Català*
Paraules clau	*Spring, Roo, GvNix*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Des d'un punt de vista personal entenc el TFG com la forma d'aplicar aquelles habilitats i coneixements apreses durant tot el grau estudiat. A més des d'un principi vaig intentar enfocar el meu TFG com una forma de seguir aprenent més que de crear un software o producte innovador.

El principal inconvenient al qual m'he enfrontat és el trobar el suficient temps per a poder realitzar un treball complet i de qualitat. Tot açò, finalment el resultat ha sigut satisfactori i amb l'objectiu d'aprendre més que complit.

Com a metodologia he aplicat un desenvolupament clàssic (estudi, anàlisi, disseny, desenvolupament, proves) però aplicant un cicle de vida iteratiu en tot moment. Aquestes iteracions ens permeten un desenvolupament més complet i de major qualitat.

Una volta finalitzat el desenvolupament podem considerar que els resultats obtinguts han sigut satisfactoris. Encara que no tots els requisits han sigut completats en totalitat, el resultat obtingut és un software complet i funcional. També voldria indicar que s'ha aconseguit una aplicació senzilla i molt usable, uns dels objectius que ens posarem al començament del projecte.

Per a concloure indicaré que amb l'experiència que em dóna aquest treball final de grau, puc dir que he fet un salt, ja no de qualitat tècnica, sinó d'experiència en un projecte complet des de zero fins al resultat final. Aquest fet em servirà per al futur en la meva carrera professional.

Abstract (in English, 250 words or less):

In this final project of degree the rapid application development tool for Java, "Spring Roo" was studied. I chose this topic because I have knowledge in Java and I considered the possibility of learning while doing this project. I knew this framework but I had never used it.

I had knowledge that most similar rapid development frameworks did not have a very correct behavior and often was discarded its use for not managing to reduce working time. I can say that after doing the work, I think that with Spring Roo has taken a big step in the automated programming since the development times can be greatly reduced.

I have used the classical methodology but applying iterative development. With this way of working I have managed to improve the results and not take errors of the first phases until the end.

The result has been remarkably satisfactory. Although I have had problems with some requirements, the end result is a simple and usable application. The greatest achievement I have reached with the development of this final work, has been learning to develop an entire project from the beginning and passing phase by phase. This experience will be of great help for my future work development.

In conclusion, I would like to highlight my satisfaction with the learning and experience gained both this project and during the years of study of the degree. Very grateful to the university and each of the teachers I have had.

Índex

1. Introducció

Dins de la meva vida professional, en gran part d'aquesta, em dedico al manteniment d'aplicacions J2EE i el desenvolupament de nous apartats o mòduls d'aquestes. Fins ara encara no m'havia trobat amb la situació de la creació d'un nou software des de zero.

Amb la meva experiència amb desenvolupament Java em vaig propossar la cerca d'una forma innovadora i interessant de producció d'aquestes. A la meva feina havia vist i realitzat petites modificacions o correccions d'errors en aplicaciones que en origen es crearen amb Spring Roo o la seva extensió GvNix. Després d'investigar un poc, em vaig decidir per enfocar aquest TFG en aquesta direcció.

Pel que fa al tema de la nostra aplicació em vaig decidir pel món de les instal·lacions esportives per ser una part important en la meva vida diària.

El gruix de les especificacions inicials han pogut ser complides, encara que com és habitual he hagut d'adaptar-me als problemes habituals en qualsevol desenvolupament. El problema més gran que he trobat i que finalment m'ha fet prendre la decisió de no complir un dels requisits ha sigut el fer que la nostra aplicació utilitze la seva pròpia base de dades com a contenidor dels usuaris que l'utilitzen, més endavant explicarem el perquè d'aquesta situació i com s'ha quedat l'aplicació finalment.

Per a finalitzar aquesta introducció, voldria comentar que, encara que no trobar solució per algun problema m'ha suposat algun mal de cap i gran pèrdua de temps, hem trobe totalment satisfet per la feina feta i pels coneixements adquirits a mesura que s'avançava el desenvolupament.

1.1 Objectius del TFG

Els objectius principals del TFG han estat l'anàlisi, disseny i implementació d'una aplicació per aprofundir en el desenvolupament d'aplicacions utilitzant tecnologia J2EE, també aprendre l'ús d'una eina de desenvolupament ràpid.

Altres objectius importants i que em poden servir per al meu futur al món laboral, han sigut la possibilitat de planificar tot el projecte des de zero i veure com adaptar aquesta planificació als problemes que van sorgint.

Finalment, altre objectiu era la construcció d'una aplicació senzilla però pràctica per a la finalitat d'aquesta.

1.2 Enfocament i mètode seguit

Com qualsevol projecte de desenvolupament d'aplicacions, el procés de construcció s'ha dividit en fases diferenciades i amb objectius diferenciats. Les fases utilitzades han sigut les següents: Estudi i anàlisi de requisits, disseny, desenvolupament i proves.

Òbviament aquestes fases de construcció no tenen un desenvolupament lineal sino que la realitat i un progrés més efectiu ens porten a realitzar diferents iteracions de totes les fases. L'aplicació d'aquestes iteracions ens ajuden a detectar mancances i defectes en els requisits inicials i que podem anar resolent a mida que s'avança el projecte.

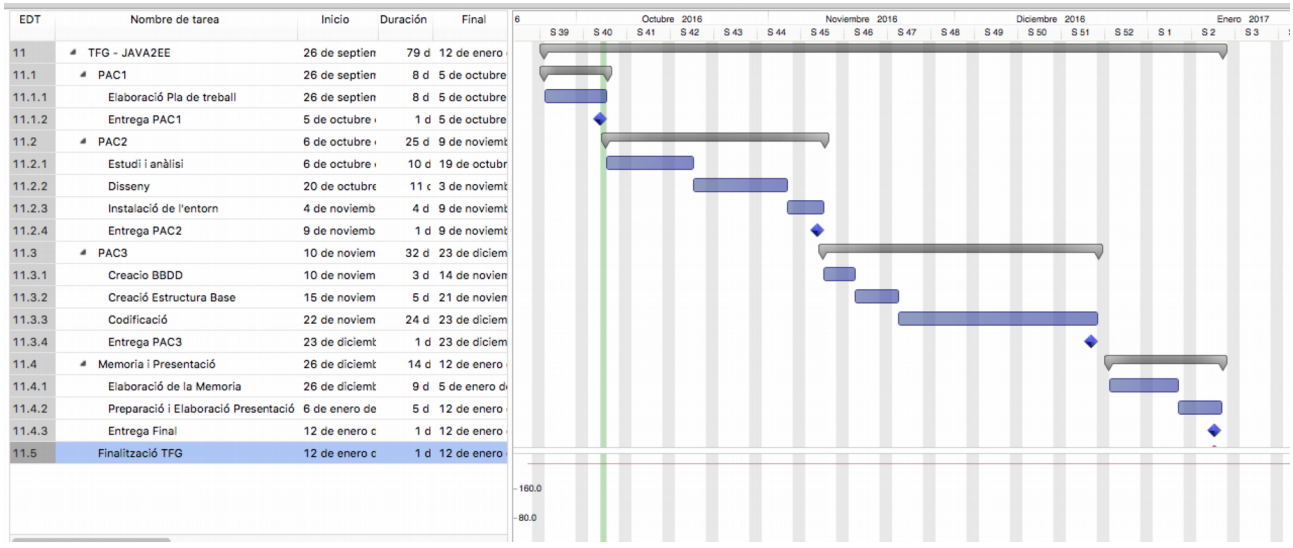
Una de les dificultats del projecte ha sigut la corba d'aprenentatge de la tecnologia utilitzada, encara que finalment ha sigut possible dur a terme aquest projecte i el gruix de les funcionalitats descrites al pla de treball inicial.

1.3 Planificació

Resum de les dades clau i lliurament de tasques del nostre projecte

Document	Inici	Lliurament	Tasca a realitzar	Nº Dies
PAC 1	26/09/2016	05/10/2016	Elaboració del pla de treball.	10
PAC 2	06/10/2016	09/11/2016	Arquitectura i anàlisi general del projecte. Configuració de l'entorn de programació.	35
PAC 3	10/11/2016	23/12/2016	Implementació del projecte.	45
Memòria i Presentació	24/12/2016	12/01/2017	Redacció de la memòria i la presentació final.	20

Per a la planificació correcta del projecte, la definició de les fites i el seguiment del desenvolupament, utilitzarem el programari gratuit StakePoint, a continuació presentem el diagrama de gant:



1.4 Producte obtingut

El producte obtingut l'he anomenat UOCFitness, unint el tema de l'aplicació amb el nom de la nostra aplicació. La nostra aplicació consisteix en un sistema de gestió de centres esportius amb un catàleg d'activitats, instal·lacions esportives o horaris d'activitats.

El producte obtingut s'ha desenvolupat amb tecnologies com: J2EE, Spring Roo, Tomcat i Maven.

Els detalls en l'arquitectura i la configuració de l'entorn els veurem més endavant en la documentació del programari.

1.5 Contingut específic de la memoria

En aquesta memòria hi trobarem la documentació habitual en el desenvolupament d'un projecte de construcció de programari. Aquests documents van creant-se a mida que avancem en el projecte i son el següents:

- Requisits del projecte.
- Definició d'actors i casos d'ús.
- Anàlisi de Requisits de l'aplicació.
- Disseny dels requisits.
- Arquitectura

2. Anàlisi de Requisits

En aquest apartat realitzarem l'anàlisi en profunditat dels requisits del nostre TFG. Cal remarcar que una bona pressa de requisits és la base fonamental per a un bon desenvolupament del projecte i té un gran impacte a les futures fases del projecte.

Com aquest no és un projecte de la vida real, no existeixen les entrevistes amb els stakeholders, però tractarem aquesta anàlisi com si fos un aplicatiu que es tingués que utilitzar al món real.

2.1 Característiques generals

Com ja em indicat abans, la nostra aplicació consistirà en un sistema de gestió de centres esportius amb un catàleg d'activitats, pistes i clients. Aquests usuaris també podran gestionar l'ús de les pistes o distintes instal·lacions esportives o, també, els horaris de les distintes activitats.

Tot i açò, també serà necessari una gestió de l'accés dels usuaris que els permetran l'accés segons el role que tinguin asignat.

2.2 Requisits funcionals

Anem ara a detallar les funcionalitats que ha de tindre l'aplicatiu. Ens basarem en la descripció feta al pla de treball però aprofundint més i agrupant per entitats o tipologies:

- Gestió d'usuaris (Sols el rol d'administrador tindrà accés a aquesta funcionalitat)
  - Alta d'usuaris
    - Alta amb un formulari
  - Baixa d'usuaris
    - Donar de baixa els usuaris del sistema
  - Cerca d'usuaris
    - Es podrà realitzar la cerca d'usuaris a través d'una pantalla amb filtres, on es mostrarà un llistat segons la cerca introduïda.
  - Modificació d'usuaris

        - Es permetrà modificar les característiques d'un usuari creat prèviament.
        - També podrem assignar la quota corresponent.

- Gestió d'activitats dirigides
    - Alta d'activitats (Rol administrador)
        - Alta amb un formulari.
    - Baixa d'activitats (Rol administrador)
        - Es donaran de baixa les activitats.
    - Cerca d'activitats (Rol administrador)
        - Es podrà realitzar la cerca d'activitats a través d'una pantalla amb filtres, on es mostrarà un llistat segons la cerca introduïda.
    - Modificació d'activitats (Rol administrador)
        - Es permetrà modificar les característiques d'una activitat creada prèviament.
    - Assignació d'activitats a usuaris (Rol abonat)
        - Secció perquè els abonats es puguin inscriure a activitats amb places lliures.

- Gestió de pistes o instal·lacions
    - Alta d'instal·lació (Rol administrador)
        - Alta amb un formulari.
    - Baixa d'instal·lació (Rol administrador)
        - Es donaran de baixa les instal·lacions o pistes.
    - Cerca d'instal·lació (Rol administrador)
        - Es podrà realitzar la cerca d'instal·lacions a través d'una pantalla amb filtres, on es mostrarà un llistat segons la cerca introduïda.
    - Modificació d'instal·lació (Rol administrador)
        - Es permetrà modificar les característiques d'una instal·lació creada prèviament.
    - Assignació d'instal·lació a usuaris (Rol abonat)
        - Secció perquè els abonats puguin reservar pistes o instal·lacions segons horari i dia indicats

- Sistema d'accés d'usuaris
    - Pantalla de login
        - El sistema disposarà d'un control de seguretat perquè els usuaris només puguin accedir a les àrees segons els seu rol.

2.3 Requisits no funcionals

Aquests tipus de requisits no estan generalment relacionats amb la funcionalitat del sistema. Ens defineix qualitats o atributs globals del sistema, o estableixen restriccions sobre el producte a desenvolupar, el procés de desenvolupament o externs.

Són igualment necessaris, ja que, d'aquests depèn que el sistema sigui usable, segur o molt eficient. Detallem, a continuació, els principals:

- **Multi idioma**: La nostra aplicació deu tindre el nombre més gran d'idiomes possibles, ja que d'aquesta manera pot arribar al major nombre d'usuaris possibles.

- **Interfície d'usuari**: Els formularis i qualsevol ferramenta del software han de ser intuitius per a l'usuari, el seu deplegament front a aquest deu ser ràpida i ha de permetre la navegació a través dels exploradors més comuns com Mozilla Firefox, Internet Explorer o Chrome. A més, del fet que la corbaa d'aprenentatge per part dels usuaris sigui el més curta possible.

- **Seguretat**: Es requereix la implementació de polítiques de seguretat comunament acceptades i s'han de considerar els següents aspectes:
  - **Identificació i autentificació**: L'autentificació ha de ser en l'àmbit d'aplicatiu i les dades relacionades com l'usuari i password es deuen emmagatzemar amb encriptació.
  - **Rols**: L'accés a la informació s'ha de controlar a través de la funció de rols d'usuari que permetran visualitzar la informació segons el tipus d'aquests.

- **Eficiència**: El sistema ha de ser ràpid i fiable, per maximitzar la satisfacció de l'usuari final de l'aplicatiu.

3. Especificació

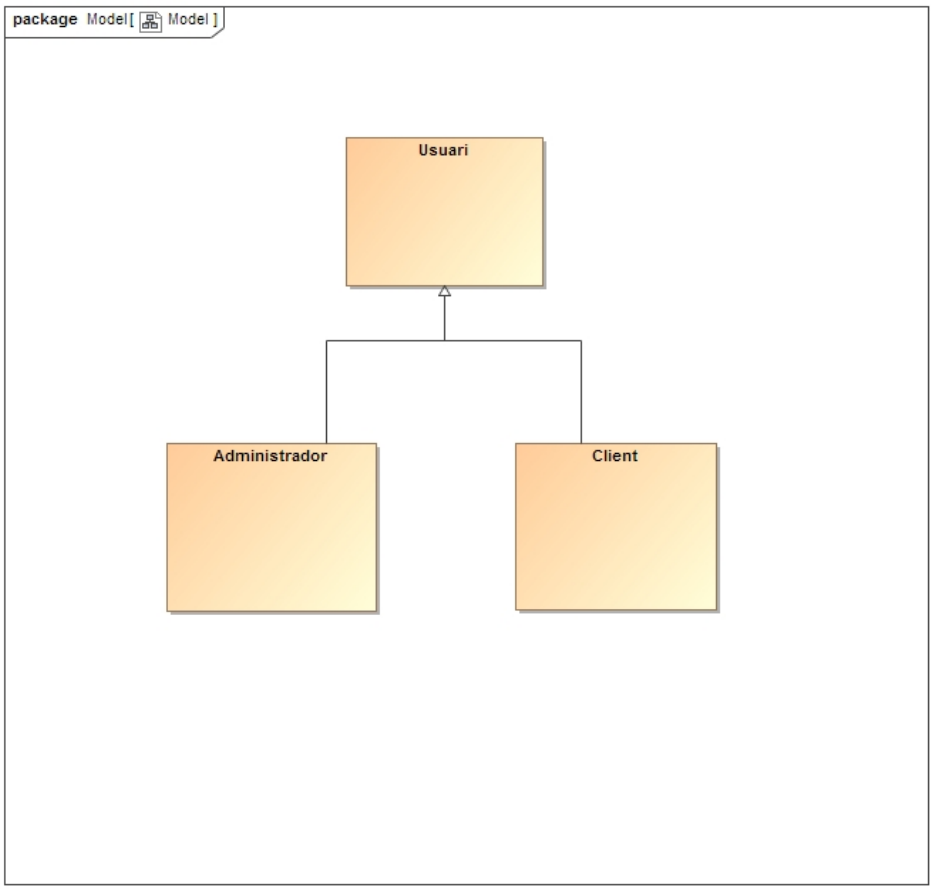
Basant-nos en els requeriments definits en l'apartat anterior, amb l'especificació del sistema tractarem de definir el comportament de l'aplicatiu.

Aleshores, haurem de decidir quins són els actors del sistema, quins casos d'ús podem realitzar i en quins escenaris.

3.1 Actors del sistema

Com ja hem parlat als requisits, la nostra aplicació disposarà d'un sistema de rols d'usuari per definir a quins llocs pot accedir cada rol, protegint així les dades privades de l'establiment o dels usuaris. Definirem dos tipus d'usuaris de la nostra aplicació, l'usuari administrador i l'usuari client del centre esportiu, tots dos hi hauran de logar-se al sistema.

Definirem el següent diagrama dels actors del sistema:



3.2 Diagrama de casos d'ús

A continuació es veu el diagrama de casos d'ús.

3.3 Especificació de casos d'ús

Ara mostrarem el anàlisis dels casos d'ús de l'aplicació. Em separat aquests casos segons les entitats a les que estem accedint i segons la funcionalitat analitzada.

3.3.1 Casos d'ús: Gestió d'usuaris i cuotes

3.3.1.1 Alta d'usuaris

Cas d'ús: Alta d'usuaris al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per crear un nou usuari. S'hauran d'insertar les dades d'aquest.

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari administrador accedeix a l'aplicació	
	2. El sistema mostra la página inicial
3. L'usuari clica sobre l'opció "Usuari>>Nou usuari"	
	4. Els sistema mostra el formulari per insertar les dades de l'usuari
5. L'usuari omple les dades del formulari i perm el botó "Desar"	
	6. El sistema valida les dades i si, totes són correctes, l'usuari es registra a la base de dades i es mostra un missatge de resultat correcte

Cas alternatiu:

6. a. El sistema detecta que alguna o algunes de les dades són incorrectes i mostra un missatge d'error a l'usuari.

3.3.1.2 Baixa d'usuari

Cas d'ús: Eliminar un abonat al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per eliminar un usuari.

Condició prèvia: Login previ a l'aplicació amb rol administrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix al usuari	
	2. El sistema mostra les dades d'aquest
3. L'usuari clica sobre l'opció "Eliminar usuari"	
	4. Els sistema elimina l'usuari del sistema

Cas alternatiu:

4.a El sistema no pot eliminar l'usuari i ens mostra un missatge d'error.

3.3.1.3 Cerca d'usuari

Cas d'ús: Cerca d'un usuari al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per realitzar

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a l'opció cerca d'usuaris	
	2. El sistema mostra un formulari de filtres
3. L'usuari introdueix les dades a cercar	
	4. Els sistema mostra un llistat amb els usuaris que compleixen els requisits introduïts

Cas alternatiu:

4.a. El sistema no troba cap usuari amb les opcions introduïdes i ens mostra un missatge.

3.3.1.4 Modificació d'abonat

Cas d'ús: Visualització i modificació d'un usuari al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per visualitzar les dades d'un usuari i modificar alguna d'aquestes dades

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a la secció d'usuaris i realitza una cerca	
	2. El sistema mostra el llistat d'usuaris
3. L'usuari clica sobre algun dels usuaris	
	4. Els sistema mostra les dades de l'usuari
5. L'usuari modifica alguna d'aquestes dades	
	6. El sistema validad les dades i si, totes són correctes, l'usuari es modifica a la base de dades i es mostra un missatge de resultat correcte

Cas alternatiu:

5. a. L'usuari decideix no modificar ninguna dada i prem l'opció Cancel·lar.

6. a. El sistema detecta que alguna o algunes de les dades són incorrectes i mostra un missatge d'error a l'usuari.

3.3.2 Casos d'ús: Gestió d'activitats dirigides

3.3.2.1 Alta d'activitat

Cas d'ús: Alta d'activitat al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per crear una nova activiat. S'hauran d'insertar les dades d'aquesta.

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a l'aplicació	
	2. El sistema mostra la página inicial
3. L'usuari clica sobre l'opció "Activitats>>Nova Activitat"	
	4. Els sistema mostra el formulari per insertar les dades de l'activitat
5. L'usuari omple les dades del formulari i perm el botó "Desar"	
	6. El sistema validad les dades i si, totes són correctes, l'activitat es registra a la base de dades i es mostra un missatge de resultat correcte

Cas alternatiu:

6. a. El sistema detecta que alguna o algunes de les dades són incorrectes i mostra un missatge d'error a l'usuari.

3.3.2.2 Baixa d'activitat

Cas d'ús: Eliminar una activitat del sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per eliminar una activitat

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a l' activitat	
	2. El sistema mostra les dades d'aquesta
3. L'usuari clica sobre l'opció "Eliminar activitat"	
	4. Els sistema elimina l'activitat del sistema

Cas alternatiu:

4.a El sistema no pot eliminar l'activitat i ens mostra un missatge d'error.

3.3.2.3 Cerca d'activitat

Cas d'ús: Cerca d'un activitat al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per cercar una activitat

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a l'opció cerca d' activitats	
	2. El sistema mostra un formulari de filtres
3. L'usuari introdueix les dades a cercar	
	4. Els sistema mostra un llistat amb les activitats que compleixen els requisits introduïts

Cas alternatiu:

4.a. El sistema no troba cap activitat amb les opcions introduïdes i ens mostra un missatge.

3.3.2.4 Modificació d'activitat

Cas d'ús: Visualització i modificació d'una activitat al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per visualitzar les dades d'una activitat i modificar alguna d'aquestes

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a la secció d'activitats i realitza una cerca	
	2. El sistema mostra el llistat d'activitats
3. L'usuari clica sobre alguna de les activitats	
	4. Els sistema mostra les dades de l'activitat
5. L'usuari modifica alguna d'aquestes dades	
	6. El sistema validad les dades i si, totes són correctes, l'activitat es modifica a la base de dades i es mostra un missatge de resultat correcte

Cas alternatiu:

5. a. L'usuar decideix no modificar ninguna dada i prem l'opciò Cancel·lar.

6. a. El sistema detecta que alguna o algunes de les dades són incorrectes i mostra un missatge d'error a l'usuari.

3.3.2.5 Assignar activitat

Cas d'ús: Assignar activitat a un abonat

Actors: Usuari administrador i Usuari client

Descripció: Accedim al sistema per assignar una activitat a un abonat

Condició prèvia: Login previ a l'aplicació amb rol adminstrador o client

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari accedeix a la secció "Activitats>>Assignació"	
	2. El sistema mostra el llistat d'activitats
3. L'usuari selecciona activitat, abonat i horari	
	4. El sistema validad les dades i si hi ha disponibilitat mostra un missatge de resultat correcte

Cas alternatiu:

3. a. Si es del tipus client no pot seleccionar un altre abonat

6. a. El sistema detecta que no hi ha disponibilitat i mostra un missatge d'error a l'usuari.

3.3.3 Casos d'ús: Gestió de reserves de pistes o instal·lacions

3.3.3.1 Alta de pista o instal·lació

Cas d'ús: Alta de pista o instal·lació al sistema
Actors: Usuari administrador

Descripció: Accedim al sistema per crear una nova pista o instal·lació. S'hauran d'insertar les dades d'aquesta.

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a l'aplicació	
	2. El sistema mostra la página inicial
3. L'usuari clica sobre l'opció "Pistes e Instal·lacions>>Nova Pista/Instal·lació"	
	4. Els sistema mostra el formulari per insertar les dades de la pista o instal·lació
5. L'usuari omple les dades del formulari i prem el botó "Desar"	
	6. El sistema valida les dades i si, totes són correctes, la pista o instal·lació es registra a la base de dades i es mostra un missatge de resultat correcte

Cas alternatiu:

6. a. El sistema detecta que alguna o algunes de les dades són incorrectes i mostra un missatge d'error a l'usuari.

3.3.3.2 Baixa de pista o instal·lació

Cas d'ús: Eliminar una pista o instal·lació del sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per eliminar una pista o instal·lació

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a la pista o instal·lació	
	2. El sistema mostra les dades d'aquesta
3. L'usuari clica sobre l'opció "Eliminar pista o instal·lació"	
	4. Els sistema elimina la pista o instal·lació del sistema

Cas alternatiu:

4.a El sistema no pot eliminar la pista o instal·lació i ens mostra un missatge d'error.

3.3.3.3 Cerca de pista o instal·lació

Cas d'ús: Cerca d'una pista o instal·lació al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per cercar una pista o instal·lació

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a l'opció cerca de pista o instal·lació	
	2. El sistema mostra un formulari de filtres
3. L'usuari introdueix les dades a cercar	
	4. Els sistema mostra un llistat amb les pistes o instal·lacions que compleixen els requisits introduïts

Cas alternatiu:

4.a. El sistema no troba cap pista o instal·lació amb les opcions introduïdes i ens mostra un missatge.

3.3.3.4 Modificació de pista o instal·lació

Cas d'ús: Visualització i modificació d'una pista o instal·lació al sistema

Actors: Usuari administrador

Descripció: Accedim al sistema per visualitzar les dades d'una pista o instal·lació i modificar alguna d'aquestes

Condició prèvia: Login previ a l'aplicació amb rol adminstrador

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari adminstrador accedeix a la secció de pistes o instal·lacions i realitza una cerca	
	2. El sistema mostra el llistat de pistes o instal·lacions
3. L'usuari clica sobre alguna de les pistes o instal·lacions	
	4. Els sistema mostra les dades de la pista o instal·lació
5. L'usuari modifica alguna d'aquestes dades	
	6. El sistema validad les dades i si, totes són correctes, la pista o instal·lació es modifica a la base de dades i es mostra un missatge de resultat correcte

Cas alternatiu:

5. a. L'usuar decideix no modificar ninguna dada i prem l'opció Cancel·lar.

6. a. El sistema detecta que alguna o algunes de les dades són incorrectes i mostra un missatge d'error a l'usuari.

3.3.3.5 Assignar pista o instal·lació

Cas d'ús: Assignar pista a un abonat

Actors: Usuari administrador i Usuari client

Descripció: Accedim al sistema per assignar una pista a un abonat

Condició prèvia: Login previ a l'aplicació amb rol adminstrador o client

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari accedeix a la secció "Pistes e Instal·lacions>>Assignació"	
	2. El sistema mostra el llistat de pistes/instal·lacions
3. L'usuari selecciona pista, abonat, dia i horari	
	4. El sistema validad les dades i si hi ha disponibilitat mostra un missatge de resultat correcte

Cas alternatiu:

3. a. Si es del tipus client no pot seleccionar un altre abonat

6. a. El sistema detecta que no hi ha disponibilitat i mostra un missatge d'error a l'usuari.

3.3.4 Casos d'ús: Gestió d'accés d'usuaris

3.3.4.1 Login

Cas d'ús: Login a l'aplicació

Actors: Usuari anònim

Descripció: L'usuari accedeix a la pàgina de login per tal d'accedir a la seva àrea personal de l'aplicació.

Condició prèvia: Cal haver fet l'alta d'abonat (Cas d'ús 4.3.1.1) sols per al rol client

Curs de l'esdeveniment:

Actor	Sistema
1. Usuari accedeix a la página de login	
	2. El sistema mostra formulari per fer login
3. L'usuari posa el nom d'usuari i el password i fa clic al botó de lgin	
	4. El sistema valida les dades i si l'usuari existeix accedeix a la página d'inici

Cas alternatiu:

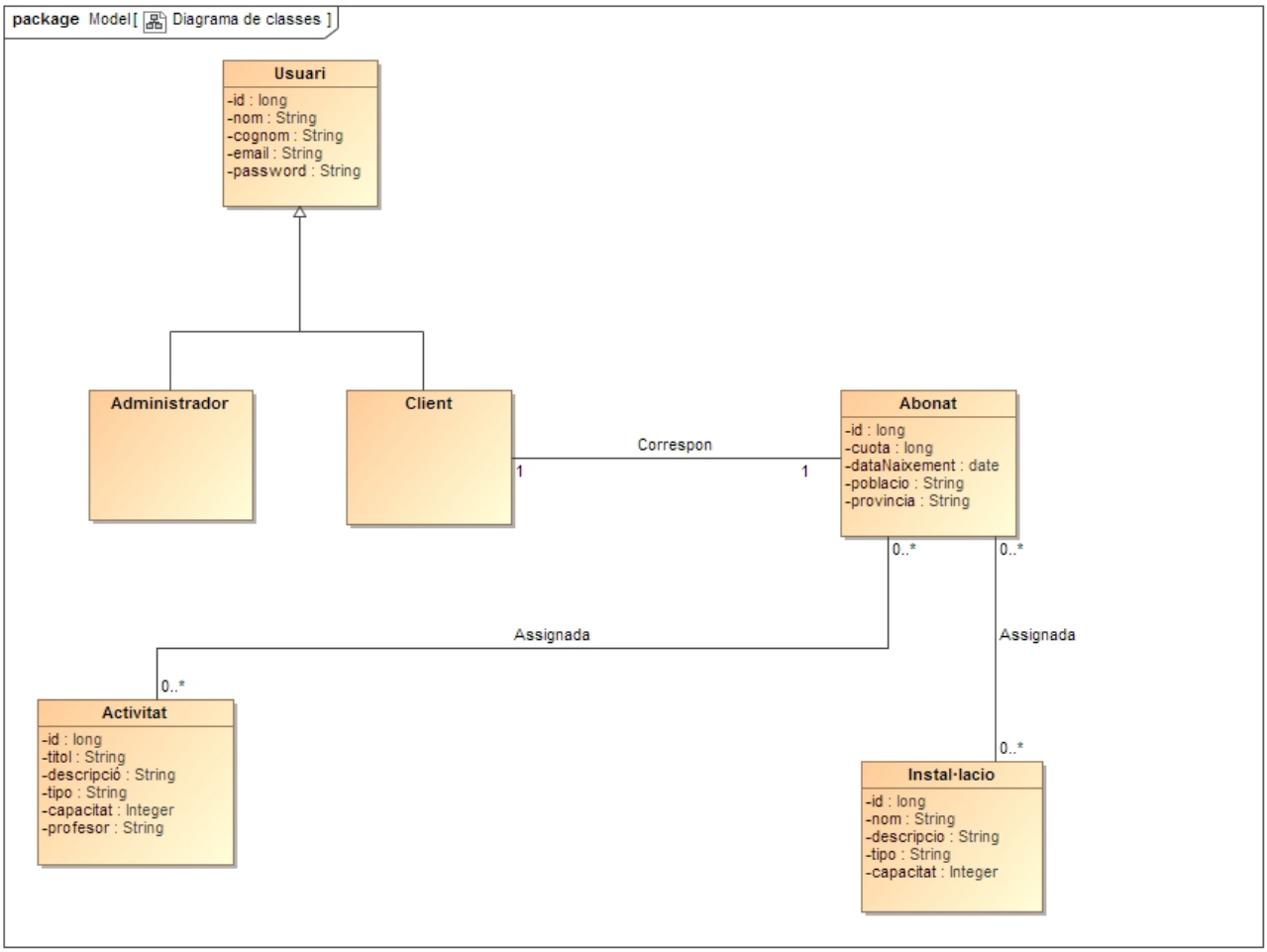
3.a: L'usuari introdueix dades incorrectes en el formulari
4.a: El sistema no troba l'usuari en les bases de dades i mostra un missatge d'error.

3.4 Model conceptual

En aquest apartat mostrarem el diagrama UML que correspon al model conceptual de la nostra futura aplicació.

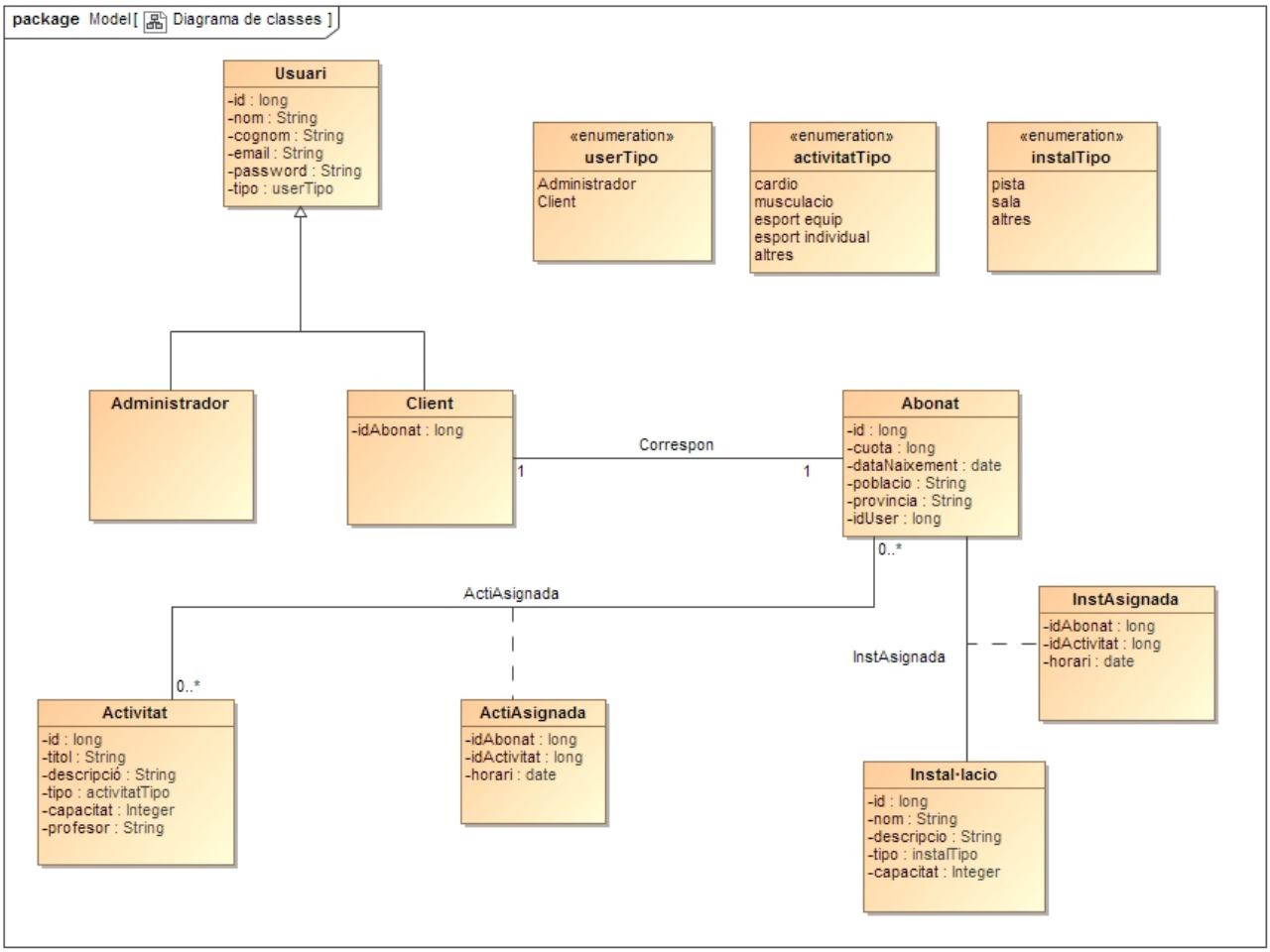
3.4.1 Model de dades no normalitzat

Presentarem ara el model previ a la seva normalització. Com podrem observar, les relacions entre taules no s'han convertit en altres taules i no existeixen encara les claus foranes. Necessitarem també diferenciar alguns tipus de classes.



3.4.2 Model de dades normalitzat

Ara presentarem el model després de la seva normalització. Aquest serà el que s'utilitzarà com a model per a la implementació:



4.Disseny

4.1 Arquitectura de l'aplicació

Habitualment les aplicacions web JavaEE s'estructuren en tres capes: la capa web, la de negoci i la d'accés a dades. En la de negoci sol haver-hi una «sub-capa» de servicis (Business Objects) que ofereixen això, serveis, a la capa web o a clients remots. A més, ací se sol considerar que està el model del domini. Finalment, en la capa d'accés a dades normalment tinguem els DAOs, que s'encarreguen de la persistencia.

L'arquitectura de les aplicacions generades per Roo per defecte és molt més simple que el que acabem de descriure. Bàsicament té dues capes, la web i la de les entitats o model de domini. Les entitats es converteixen en una espècie de «superentitats» que s'encarreguen no sols de modelar el domini sinó també de tasques com la seva pròpia persistència, la validació de les seves dades i inclús la seva serialització en JSON. Aquesta idea de que els mateixos objectes de domini s'encarreguen de la seva persistència no és exclusiva de Roo, en realitat és molt típica de les plataformes RAD com Rails, Django, Grails i altres, i es coneix habitualment com Active Record.

En alguns casos ens interessarà tenir en l'aplicació serveis i/o DAOs, Encara que no els utilitzi per defecte, Roo també pot generar-los si li ho demanen.

Les claus de l'Arquitectura Spring Roo són:

- L'interpret de comandaments.
- Les anotacions @Roo, amb nivell de retenció de codi (RetentionPolicy = SOURCE), és a dir, descartades pel compilador.
- Les introduccions AspectJ (AspectJ inter-type declarations – IDT)

- Un model de metadades que facilita el desenvolupament dels mòduls add-on.
- Totalment compatible amb l'edició del codi de l'aplicació. Ja que, el codi generat i mantingut per Roo es troba a IDTs, si editem o canviem les classes java, Roo descobrirà els canvis i adaptarà els IDTs a la nova situació

4.1.1 Introduccions de AspectJ

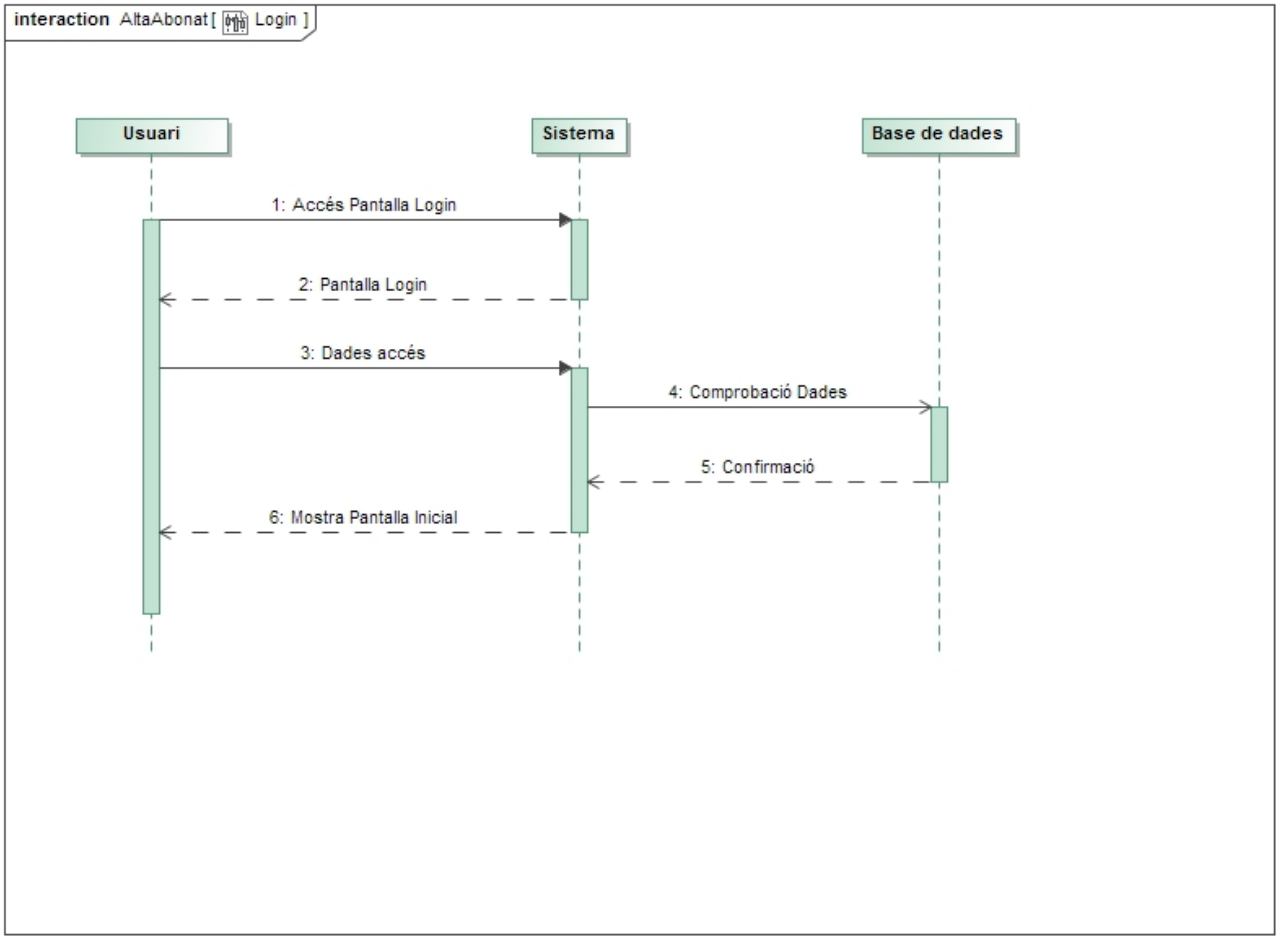
Fins ara a tots els generadors de codi s'ha pogut comprovar que estàvem canviant duros per pessetes. Si utilitzaves un generador de codi per a crear els models d'entitats de la base de dades, per exemple, el model generat mancava de les característiques pròpies d'un model natural OO i acabava comprometent el projecte. El control sobre el codi generat era escas i dificilment es podía especificar les jerarquies o les interfícies que el codi generat debia implementar.

Spring Roo evita aquests inconvenients d'una forma extremadament elegant. Mitjançant l'ús d'IDTs. Tot el codi que genera Roo es troba separat de les classes Java sobre les quals el programador desenvolupa, en les introduccions de AspectJ (amb extensió .aj). En temps de compilació s'uneixen les classes Java a les introduccions per crear un únic fitxer .class. Com la classe és idèntica a com seria si el programador hagués escrit tot el codi ell mateix, tots els avantatges del llenguatge Java i dels IDEs de desenvolupament funcionen a la perfecció. El programador pot ignorar per complet del codi generat en els IDTs. Aquests contenen conceptes i funcionalitats que s'han delegat a Roo i aquest és l'encarregat de mantenir-les.

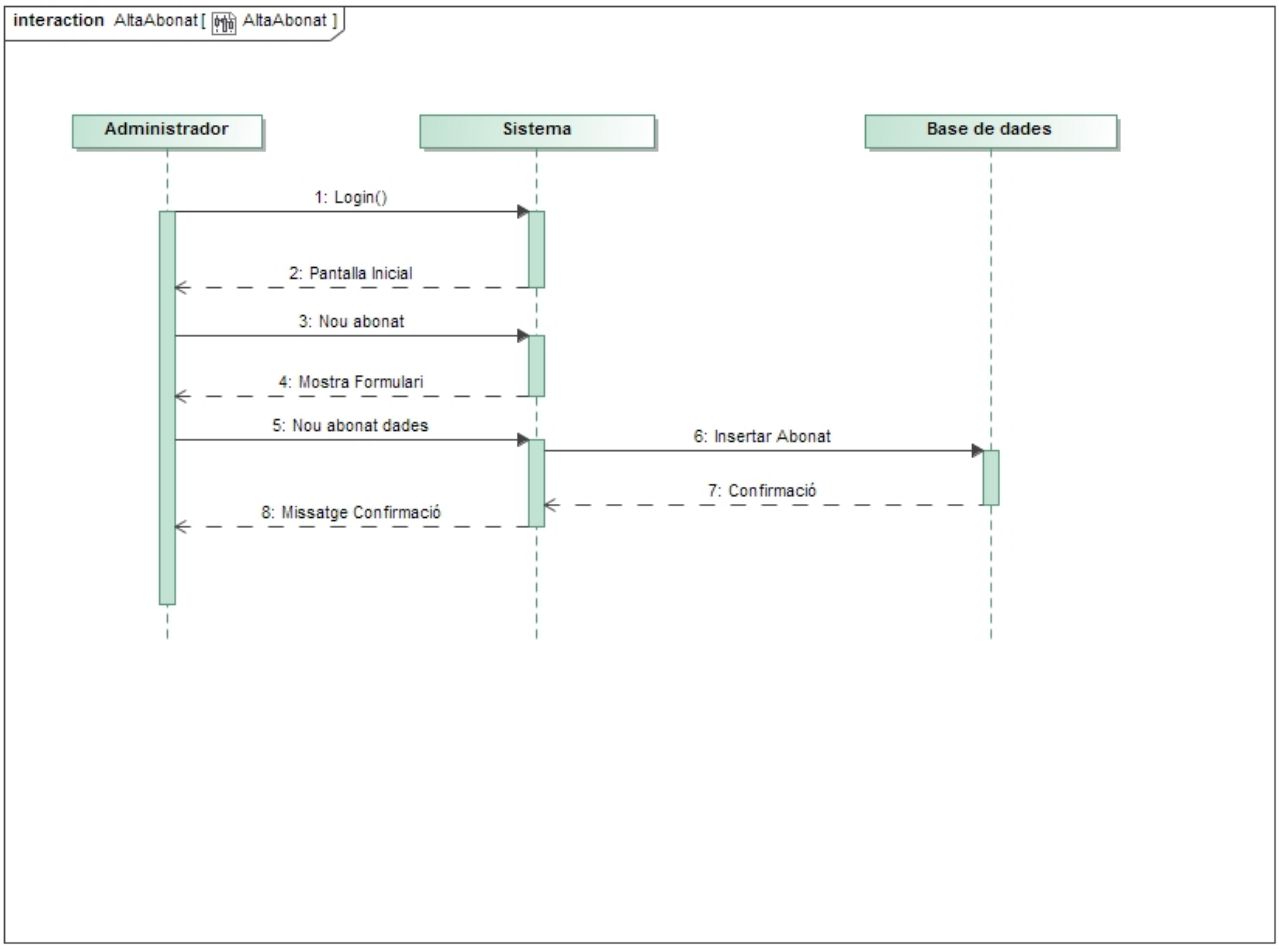
4.2 Diagrames de seqüència

Ja que el nostre aplicatiu es basa en altes, baixes i modificacions, mostrarem un diagrama de seqüència de cada tipus, per no repetir el de totes les entitats. A més també mostrarem un cas d'assignació i la seqüencia de login:

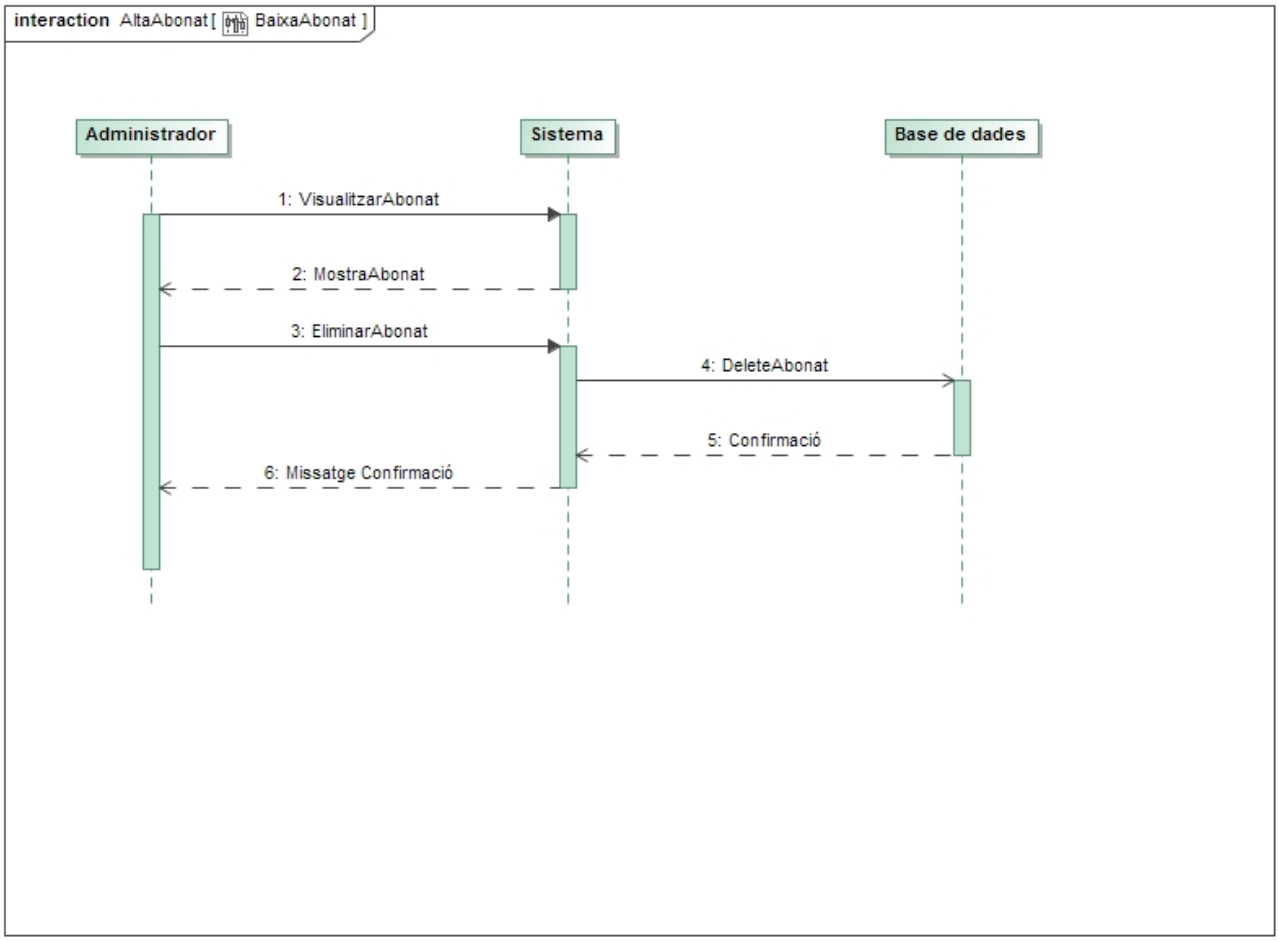
- Diagrama de seqüència Login:



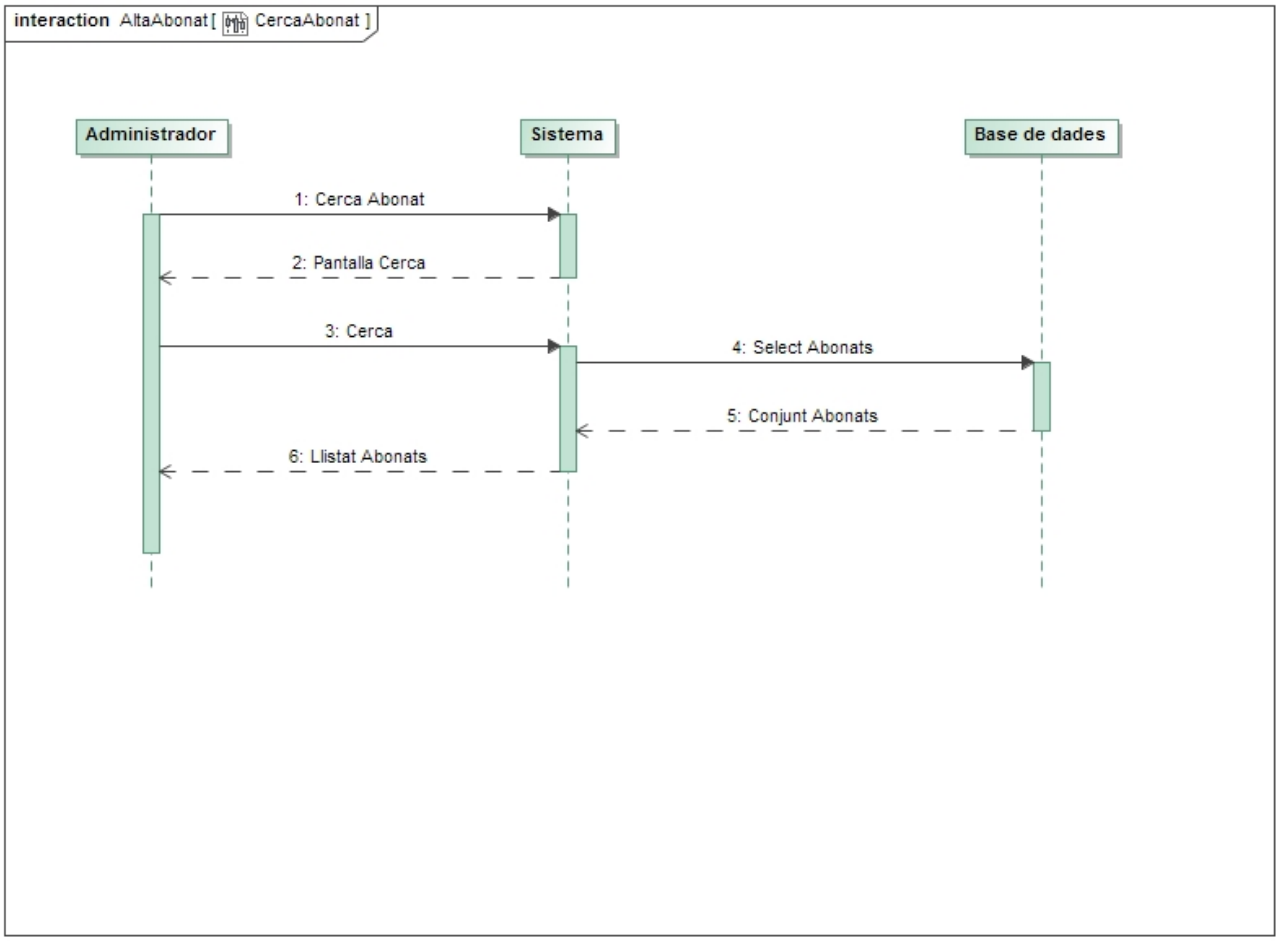
- Diagrama de seqüència Alta Abonat:



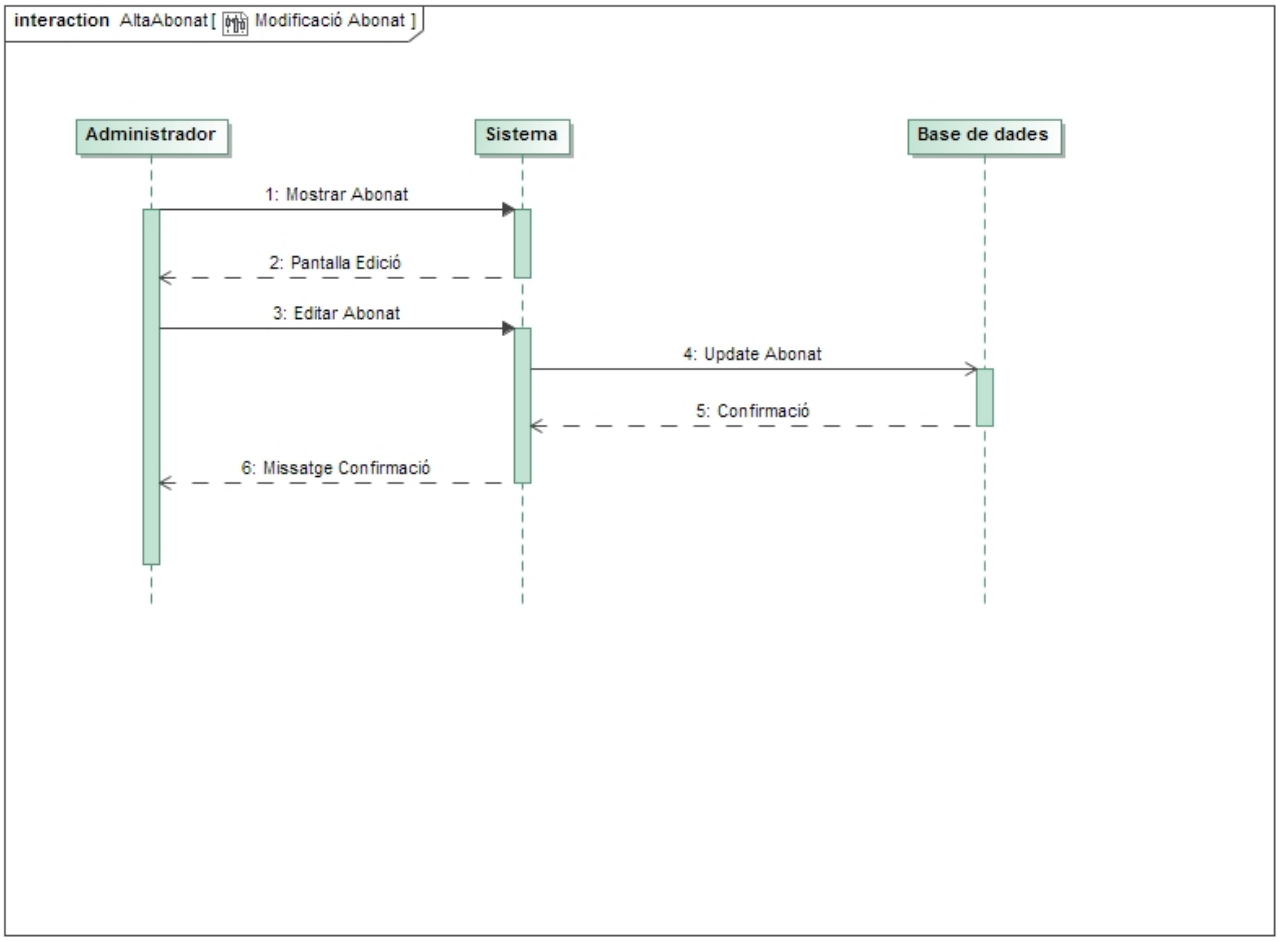
- Diagrama de seqüència Baixa Abonat:



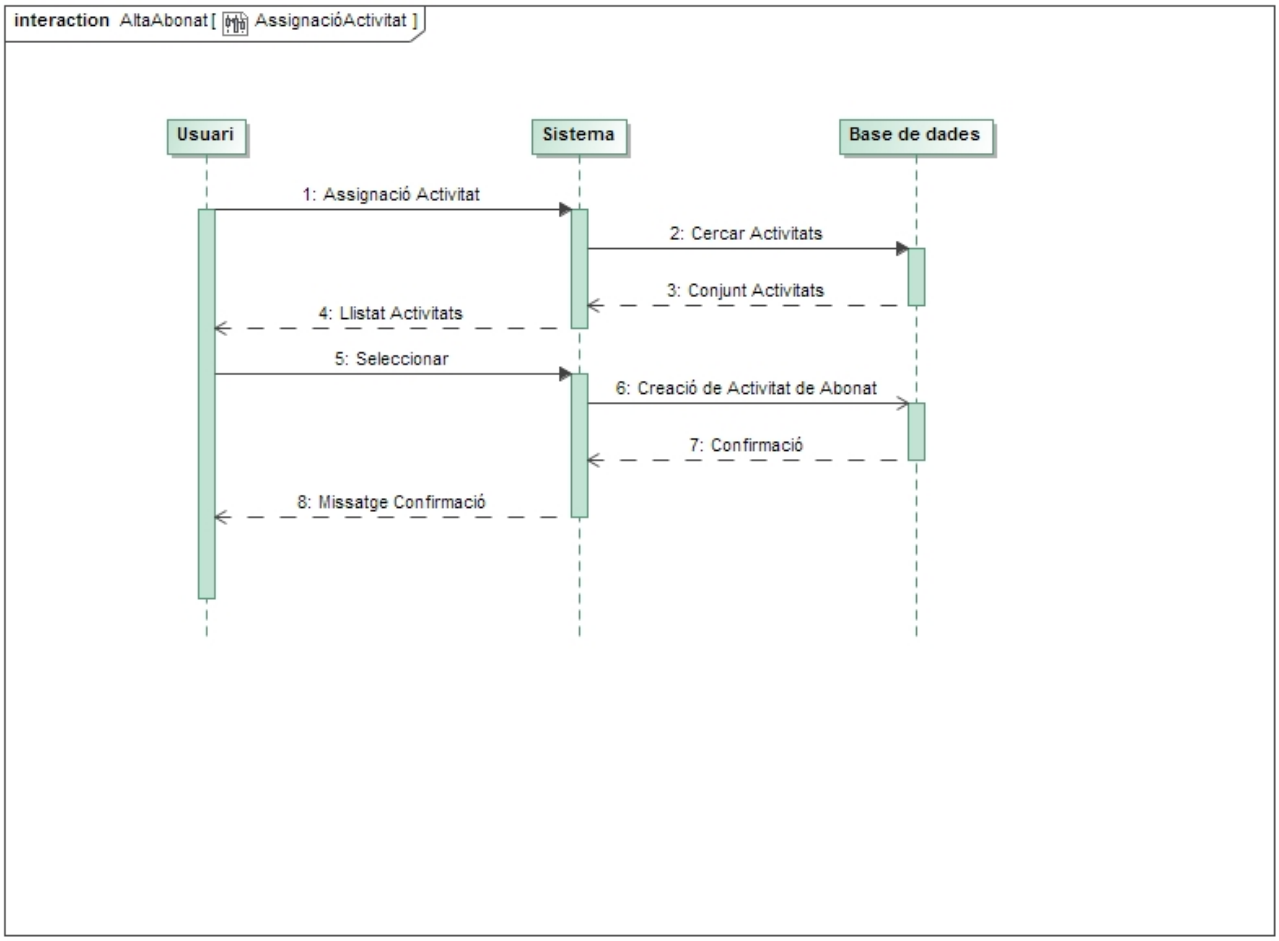
- Diagrama de seqüència Cerca Abonat:



- Diagrama de seqüència Modificació Abonat:



- Diagrama de seqüencia Assignació de Activitat:



5. Implementació

5.1 Instal·lació i configuració

Quan finalitzem aquest apartat comptarem amb un entorn de desenvolupament preparat per a la codificació del nostre aplicatiu.

Partint d'un entorn base amb Windows 7 de 64 bits, ha sigut necesària la instal·lació i/o configuració del següent llistat de funcionalitats:

- **Versió 8 del JDK de Java**: Segons la documentació inicial aportada per Spring son valides les llibreríes JDK 1.6 y 1.7. Pero amb les últimes actualitzacions s'accepta també la 1.8 que finalment es la que hem utilitzat.

- I**nstal·lació del IDE Springsource Tool Suite (STS)**: Spring ha desenvolupat aquesta extensió d'Eclipse que ofereix suport per treballar amb aplicacions Spring, entre aquestes es troba Spring Roo. Entre altres funcionalitats, la més important és la incorporació d'un interpret de comandos de Roo que permet executar ordres sense necessitat d'eixir del mateix IDE.

- **Configuració consola ROO**: Per obtenir la nomenada consola Roo, és necessari la descàrrega del plugin «Roo Extension», una volta descarregat, l'associarem amb la versió de Spring Roo que prèviament haurem descarregat de la pàgina oficial http://projects.spring.io/spring-roo/.

- **Configuració Maven**: És necessària la instal·lació de Maven2 per a la gestió de totes les llibreries necesàries del nostre aplicatiu.

Amb tot açò, ja disposarem de l'entorn necessari per a la codificació de l'aplicatiu amb Spring Roo.

5.2 Anàlisi de la creació d'entitats en Spring Roo

Encara que inicialment decidirem crear totes les entitats desde Spring Roo, finalment i després de l'estudi de la tecnologia, ens trobarem amb la funcionalitat de la reingenieria inversa.

Aqueste ens permet, creant la base de dades previament, obtindre tot el mapejat de la entitats amb les següents ordres desde la linia de comandament de la consolo Roo i petites modificacions al fitxer de configuració.

L'ordre a executar és la següent:

```
jpa setup --provider HIBERNATE --database POSTGRES
--databaseName UOCFitness –username postgres –password
postgres
```

Amb aquesta ordre automàticament se'ns crearàn els fitxers necessaris de: configuració de l'aplicació, la persistència i el database.properties, aquest és el que s'ha de modificar indicant l'usuari i password de connexió.

Una volta configurada la base de dades, ja podem procedir a la creació de les entitats de la nostra base de dades amb la nombrada reingenyeria inversa.

Per a poder realitzar aquesta es necessari la posada en funcionament de la llibreria de connexió de l'aplicatiu amb la base de dades postgres i per això s'executa la següent ordre manualment:

```
osgi start –url file:///C:/sts-bundle/postgresql-
9.4.1212.jre6.jar
```

Sols son necessaries dues ordres per realitzar la reingenyeria inversa i obtindre el mapejat de totes les clases necessaries en la nostra aplicació:

```
database introspect –schema public
```

```
database reverse engineer –schema public –package
~.domain
```

5.3 Anàlisi de la creació de la vista-controlador MVC en Spring Roo

Una volta configurades les entitats, la creació de controladors i les vistes d'aquestes es realitzen d'una forma molt senzilla.

Primerament utilitzarem el comandament per a la creació dels controladors de l'entitat:

```
web mvc setup
```

Seguidament crearem les vistes per a cada entitat:

- Primerament posem el focus en l'entitat que volem:

```
focus –class ~.domain.usuari
```

- Després procedim a la creació:

```
web mvc scaffold –class
uoc.fitness.web.UsuariController
```

Amb aquests comandaments ja tindríem l'aplicació base creada per a la seva utilització. Evidentment serà necessari adaptar-la a les nostres neccessitats pròpies.

5.4 Anàlisi de la configuració de seguretat en Spring Roo

Spring Roo també ens ofereix la possibilitat d'afegir una configuració de control d'accés a la nostra aplicació. Ens crearà un accés de tipus login i també podrem restringir l'entrada a alguns apartats segons el role d'usuari.

Per a l'instal·lació del Spring Secutity addon utilitzarem el següent comandament:

~.web roo> security setup

Posteriorment desde el STS: «Run As > Maven install» per a descarregar les llibreries que hem incorporat al nostre projecte en executar el «security setup».

Podrem observar que, a més de actualitzar-se els fitxers existents i afegir les llibreries necessàries, també s'han creat dos nous fitxers:

- login.jspx: nova pantalla de login
- applicationContext-security.xml: per a la configuració dels controls d'accés.

Segons els roles que tinguem creats podrem configurar l'accés als elements de la nostra aplicació, de la següent forma:

```
<intercept-url pattern="/nomEntitat/**"
access="hasRole('ROLE_ADMIN')"/>
```

5.5 Anàlisi de la configuració multiidioma en Spring Roo

Incorporar multi idioma a la nostra aplicació es pot realitzar d'una forma molt senzilla amb Spring Roo, simplement haurem d'executar el següent comandament per a cada idioma que volem afegir:

~.web roo> mvc language –code es

Es crearà un nou arxiu per emmagatzemar les traduccions dels missatges a l'espanyol, message_es.properties.

Per a completar les traduccions dels camps de la nostra aplicació, necessitarem la creació d'un nou fitxer, aplication_es.properties.

6. Conclusions

Personalment, l'elecció de realitzar el TFG relacionat amb J2EE ha sigut motivada pel propi interès de profunditzar en una tecnologia ja coneguda per mi, al ser habitual al meu dia a dia laboral.

A més, el TFG m'ha suposat aplicar molts dels coneixements adquirits a les diferents assignatures del grau, sent molt més presents aquelles relacionades amb l'enginyeria del programari o les tècniques de desenvolupament d'aplicacions. També durant l'execució del projecte s'han adquirits nous coneixements en l'eina de desenvolupament automàtic Spring Roo i la seva variant GvNix.

Des d'un punt de vista més professional, a més de l'aprenentatge de la nova tecnologia, desenvolupar un projecte des de zero amb totes les etapes d'anàlisi, disseny i implementació, m'han ajudat a vorer el món del desenvolupament des d'un altre punt de vista diferent del programador. Aquesta experiència m'ajudarà en un futur per a millorar la meva carrera professional i ampliar coneixements.

7. Requisits no implementats

Pel que fa al requisit per el qual els usuaris que insertàvem eren els que podrien utilitzar per accedir a l'aplicació, ens ha sigut possible configurar correctament Spring Security.

Encara que segons tota la documentació analitzada les modificacions a realitzar en el arxiu applicationContext-security.xml eren senzilles i amb unes consultes simples era suficient, no s'ha aconseguit connectar amb la base de dades.

Després de cercar informació sobre aquest problema, una possible causa podria ser el tipus de llenguatge de base de dades, PostgreSql, i el seu tipus de dades. Encara que s'ha intentat trobar el tipus de dades correctes no s'ha trobat amb la solució.

Expliquem un poc com es deuría d'haver connectat i com hem deixat dos casos d'exemple per que es puga probar la funcionalitat de roles:

```
<!-- Configure Authentication mechanism -->
<authentication-manager alias="authenticationManager">
    <!-- SHA-256 values can be produced using 'echo -n your_desired_password | sha256sum' (using normal *nix environments) -->
    <authentication-provider>
        <password-encoder hash="sha-256" />
         <!-- <jdbc-user-service data-source-ref="dataSource"

               users-by-username-query="
                 SELECT nom AS username, password, true
                 FROM usuari
                 WHERE nom=?;"

               authorities-by-username-query="
                SELECT nom AS username, 'ROLE_USER' AS authority
                FROM usuari
                WHERE nom=?;"

        />
        -->
         <user-service>
            <user name="admin" password="8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918" authorities="ROLE_ADMIN" />
            <user name="user" password="04f8996da763b7a969b1028ee3007569eaf3a635486ddab211d512c85b9df8fb" authorities="ROLE_USER" />
        </user-service>

    </authentication-provider>
</authentication-manager>
```

La part comentada són les consultes que, teòricament, haurien de ser suficients per connectar els users de base de dades amb el login de Spring Security. Finalment s'ha optat per deixar dos usuaris d'exemple: «admin», «user» per poder utilitzar la funcionalitat d'accés amb login i de roles, com podem vorer s'utilitza una codificació sha-256 per al password dels usuaris.