Universitat Oberta de Catalunya

uoc.edu

Gestió borses de treball

Jaume Rosselló Jaume
Grau d'Enginyeria Informàtica
Desenvolupament web

Consultor/a Vicenç Font Sagrista
Professor/a responsable Santi Caballe Llobet

TFG | Juny 2021

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

FITXA DEL TREBALL FINAL

Títol del treball:	*Gestió de borses de treball*
Nom de l'autor:	*Jaume Rosselló Jaume*
Nom del consultor/a:	*Vicenç Font Sagrista*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*06/2021*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Desenvolupament web*
Idioma del treball:	*Català*
Paraules clau	*Borses de treball, java, spring boot*

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

L'objectiu del projecte es facilitar al departament de recursos humans la gestió de borses de treball que son publicades als distints butlletins oficials. L'aplicació està dirigida principalment a empreses publiques on estan obligades a fer una publicació amb la finalitat de cobrir la cobertura de llocs temporals o provisionals derivats de baixes, excedències, etc. També es pot utilitzar per a la empresa privada per a gestionar possibles candidats per llocs de feina determinats.

En aquesta primera fase de l'aplicació la finalitat es poder generar una llista de persones excloses i aptes on recollir les distintes puntuacions definides per distints criteris com pot ser experiència professional, idiomes, estudis, cursos, entrevista personal, etc.

Tota l'aplicació integrat dins un àmbit de seguretat per preservar la informació personal, on també pot accedir el candidat final per a fer consulta de les seves inscripcions a distintes borses i consultar la puntuació obtinguda.

L'aplicació es fàcilment escalable i afegir nous mòduls de control, com podria ser, gestió de cridades telefòniques, control d'ordre de contractació, motius d'exclusió, etc. Tot dins un àmbit de multi idioma perquè sigui fàcil la integració a altres comunitats o països.

La metodologia utilitzada es la de model-vista-controlador amb llibreries estàndard perquè funcioni a qualsevol navegador web amb accés a una base de dades postgreSQL.

En conclusió es te una major control de les dades, amb possibilitat de ampliar funcionalitats, amb una seguretat extra, i el punt a favor de la consulta del candidat i

poder eliminar un poc la opacitat de les administracions publiques sobre la població.

Abstract (in English, 250 words or less):

The aim of the project is to make it easier for the human resources department to manage job vacancies that are published in the various official bulletins. The application is aimed primarily at public companies where they are required to make a publication in order to cover the coverage of temporary positions arising from sick leave, leave of absence, etc. It can also be used for the private company to manage potential candidates for certain jobs.

In this first phase of the application, the purpose is to be able to generate a list of excluded and suitable people where to collect the different scores defined by different criteria such as professional experience, languages, studies, courses, personal interview, etc.

The entire application is integrated within a security area to preserve personal information, where the final candidate can also access to consult their registrations on different scholarships and check the score obtained.

The application is easily scalable and adds new control modules, such as phone call management, contract order control, exclusion grounds, etc. All within a multi-language environment so that integration into other communities or countries is easy.

The methodology used is that of model-view-controller with standard libraries so that it works in any web browser with access to a postgreSQL database.

In conclusion, there is greater control over the data, with the possibility of extending functionality, with extra security, and the point in favor of consulting the candidate and being able to eliminate a little the opacity of public administrations on the population.

Índex

1. Introducció 1
1.1 Context i justificació del Treball 1
1.2 Objectius del Treball 1
1.3 Enfocament i mètode seguit 2
1.4 Planificació del Treball 3
1.5 Tecnologies utilitzades 5
2. Anàlisi de requisits 7
2.1. Requeriments 7
2.2. Rols i permisos 8
2.3. Requeriments funcionals 8
2.4. Requeriments no funcionals 9
3. Disseny de l'aplicació 9
3.1. Disseny conceptual, diagrama de classes UML 9
3.2. Model lògic relacional 11
3.3. Casos d'us 12
3.4. Especificació casos d'us 13
3.5. Diagrama de components 21
3.6. Disseny 23
4. Desenvolupament de l'aplicació 28
4.1. Preparació de l'entorn 28
4.2. Capa de persistència 30
4.3. Capa de negoci 33
4.4. Capa de presentació 35
4.5. Capa de seguretat 37
4.6. Configuració general de l'aplicació 38
4.7. Posant en marxa l'aplicació 40
4. Conclusions 47
5. Glossari 49
6. Bibliografia 50
7. Annex – Guia d'instal·lació 51
7.1. Instal·lar JDK 8 51
7.2. Instal·lar PostgreSQL 53
7.3. Instal·lar maven 54
7.4. Configurar i executar la aplicació 55

Llista de figures

Il·lustració 1: Diagrama de Gantt 3
Il·lustració 2: Diagrama de Gantt part 2 4
Il·lustració 3: Diagrama de tecnologies 6
Il·lustració 4: Logotip GitHub 6
Il·lustració 5: Logotip Heroku 6
Il·lustració 6: Diagrama de classes UML 10
Il·lustració 7: Diagrama de casos d'us 12
Il·lustració 8: Diagrama de components 21
Il·lustració 9: Serveis dels components 22
Il·lustració 10: Pantalla login 23
Il·lustració 11: Pantalla índex - llista borses de treball 23
Il·lustració 12: Pantalla gestió borsa 24
Il·lustració 13: Pantalla Inscripcions 25
Il·lustració 14: Pantalla gestió de candidats 26
Il·lustració 15: Pantalla gestió de categories 27
Il·lustració 16: Esquema entitats 30
Il·lustració 17: Exemple entitat Borsa 30
Il·lustració 18: Esquema entitat-relació generat PostgreSQL 31
Il·lustració 19: Exemple repositori Borsa 32
Il·lustració 20: Llista mètodes per defecte del repositori 32
Il·lustració 21: Mètodes manuals dins repositori 32
Il·lustració 22: Mapa web segons rols 36
Il·lustració 23: Exemple fitxers internacionalització 39
Il·lustració 24: Comanda maven per arrancar l'aplicació 40
Il·lustració 25: Log d'inici de l'aplicació 40
Il·lustració 26: Pantalla accés aplicació 41
Il·lustració 27: Menú amb rol SIS_ADMIN 41
Il·lustració 28: Pantalla inicial borses de treball amb rol Admin 41
Il·lustració 29: Pantalla gestió de categories laborals 42
Il·lustració 30: Pantalla gestió de borsa 42
Il·lustració 31: Exemple pantalla candidats 43
Il·lustració 32: Pantalla gestió borses, botó accés inscripcions 43
Il·lustració 33: Pantalla gestió inscripcions a borsa 43
Il·lustració 34: Pantalla gestió borses amb rol Editor 44
Il·lustració 35: Exemple pantalla mèrits d'un candidat 45
Il·lustració 36: Pantalla gestió inscripcions. Llista candidats 45
Il·lustració 37: Pantalla consulta candidats 46
Il·lustració 38: Esquema cicle vida del software 47

1. Introducció

1.1 Context i justificació del Treball

L'aplicació que es vol desenvolupar ve donada de la necessitat trobada a alguns departaments de recursos humans, sobretot de l'administració pública a l'hora de gestionar les distintes sol·licituds de candidats a borses de treball, a la qual, segons les categories ofertes la llista de candidats pot ser molt extensa.

A dia d'avui hi ha departaments que per a la gestió actual de borses utilitzen fulls de càlcul on es traspassa la informació de forma manual, i alguns pics de forma duplicada, ja que si les dades personals ja existeixen a una borsa no seria necessari tornar a repetir, i a més el càlcul de puntuació de mèrits i experiència es calculen de forma manual amb la conseqüent pèrdua de temps i possibles errades de transcripció.

També un dels principals problemes trobats es la concurrència de persones fent feina damunt un mateix full de càlcul on es podien trobar problemes de pèrdua d'informació. Fins ara ho arreglaven només fent feina una persona a la vegada avisant abans d'accedir al fitxer del servidor.

1.2 Objectius del Treball

L'objectiu principal de la fase de l'aplicació que es farà al TFG es poder obtenir una llista definitiva de gent presentada a borses de l'administració on la ordenació es basarà segons els mèrits entregats per el candidat a l'administració on per a cada borsa es defineix una baremació distinta segons les necessitats de la categoria i lloc o llocs de feina que es volen omplir.

La primera fase pretén controlar un manteniment basic de borses de treball, controlat per distints rols d'usuaris. On un usuari administrador pot gestionar la borsa i controlar l'estat de la mateixa, definint el criteri de baremació de punts per a cada mèrit i titulació presentada per el sol·licitant. Un usuari tècnic que serà l'encarregat de incloure les dades al sistema com poden ser les dades personals i els distints títols presentats per la candidatura. El darrer rol serà nomes de consulta personal, on el treballador, amb un usuari i contrasenya que se li ha atorgat a l'hora de donar d'alta al sistema, podrà consultar l'estat de la seva petició.

A una segona fase, dins la gestió de categories i de persones dins les borses definides, també es podria afegir el control de les places de feina, sabent quina persona ocupa una determinada plaça, i des de quina borsa l'ha ocupada. També es pot afegir un control de cridades telefòniques, podent controlar en base a incidències d'un històric de la persona i dur control del temps i places ocupades.

1.3 Enfocament i mètode seguit

Per a dur a terme el treball es desenvoluparà una plataforma web, ajudant així a la concurrència de distints treballadors a l'hora damunt una mateixa borsa de treball. Ajudant així també a la concurrència d'informació de treballadors per així no duplicar informació com podia ocórrer als fulls de càlcul.

No es contempla fer una importació dels excels fets fins a la data ni fer cap migració d'informació des de altra sistema, ja que, com les borses tenen una data de caducitat, així com es vagin caducant, les noves s'afegiran al nou sistema amb una nova remesa de sol·licituds, ja que l'objectiu no es la renovació de borses, sinó de la creació de borses noves a cada caducitat. Però si alguna administració voldria usar aquesta aplicació amb el sistema de borses de renovació de punts a distints períodes, es podria afegir fer un mòdul d'importació de dades personalitzat, però en aquest cas es fora de l'àmbit del TFG.

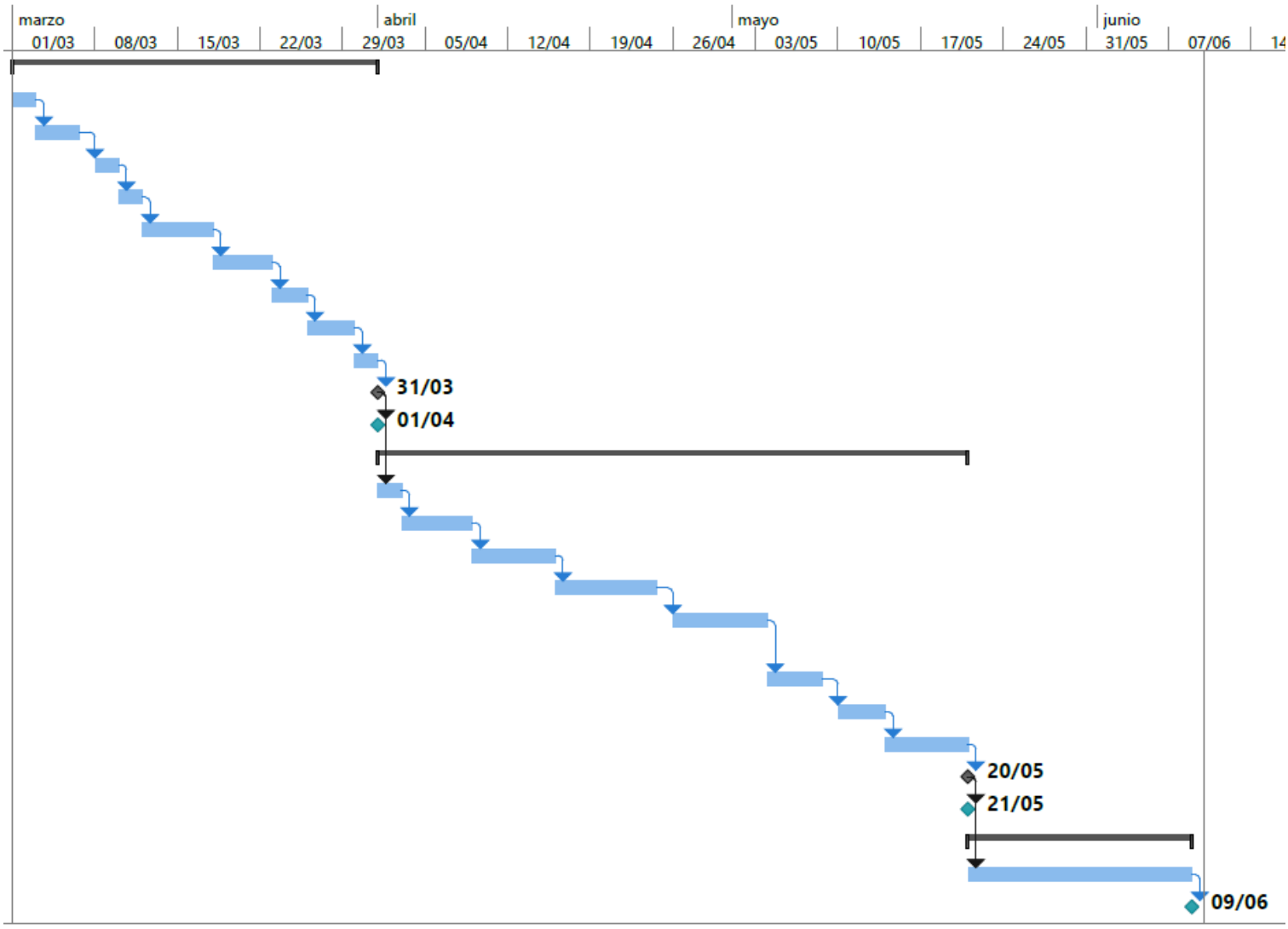
La introducció de dades es farà a traves d'un navegador web, on, per a fer aquest desenvolupament les proves es faran sempre amb la versió mes actualitzada del navegador Google Chrome.

1.4 Planificació del Treball

Per a la planificació del temps del desenvolupament de l'aplicació, s'ha agafat com a base el temps del trimestre i les fites d'entrega, i adaptar-los el mes possible a l'objectiu final.

Id	Nombre de tarea	Duración	Comienzo	Fin	Pred
1	**Anàlisi i disseny**	**27 días**	**lun 01/03/21**	**jue 01/04/21**	
2	Presa de requeriments	2 días	lun 01/03/21	mar 02/03/21	
3	Anàlisi dels requeriments	4 días	mié 03/03/21	sáb 06/03/21	2
4	Definició rols i permisos (actors)	2 días	lun 08/03/21	mar 09/03/21	3
5	Diagrama UML	2 días	mié 10/03/21	jue 11/03/21	4
6	Casos d'us	5 días	vie 12/03/21	mié 17/03/21	5
7	Anàlisi serveis (back)	4 días	jue 18/03/21	lun 22/03/21	6
8	Anàlisi interfície (front)	3 días	mar 23/03/21	jue 25/03/21	7
9	Diagrama de classes	3 días	vie 26/03/21	lun 29/03/21	8
10	Diagrama BBDD - ER	2 días	mar 30/03/21	mié 31/03/21	9
11	Anàlisi i disseny [Fita completa]	0 días	mié 31/03/21	mié 31/03/21	10
12	Fi lliurament PAC2	0 días	jue 01/04/21	jue 01/04/21	11
13	**Desenvolupament**	**43 días**	**jue 01/04/21**	**vie 21/05/21**	
14	Configuració entorn	2 días	jue 01/04/21	vie 02/04/21	11
15	Definició arquitectura	5 días	sáb 03/04/21	jue 08/04/21	14
16	Back - Creació entitats (BBDD)	6 días	vie 09/04/21	jue 15/04/21	15
17	Back - Creació serveis	8 días	vie 16/04/21	sáb 24/04/21	16
18	Front - Creació beans (controladors)	7 días	lun 26/04/21	lun 03/05/21	17
19	Front - Creació jsp	5 días	mar 04/05/21	sáb 08/05/21	18
20	Proves	4 días	lun 10/05/21	jue 13/05/21	19
21	Tests	6 días	vie 14/05/21	jue 20/05/21	20
22	Desenvolupament [Fita completa]	0 días	jue 20/05/21	jue 20/05/21	21
23	Fi lliurament PAC3	0 días	vie 21/05/21	vie 21/05/21	22
24	**Memòria i presentació**	**16 días**	**vie 21/05/21**	**mié 09/06/21**	
25	Preparar projecte i documentació	16 días	vie 21/05/21	mar 08/06/21	22
26	Fi lliurament Memòria	0 días	mié 09/06/21	mié 09/06/21	25

Il·lustració 1: Diagrama de Gantt



Il·lustració 2: Diagrama de Gantt part 2

1.5 Tecnologies utilitzades

Springboot 2.4.3[1]: entorn de desenvolupament basat en el framework de codi obert Spring, on no es necessari la instal·lació de cap servidor per a fer el desenvolupament d'una aplicació web.

Com a inici s'utilitzaran les dependències principals següents, podent anar afegint d'altres segons la necessitat:

- **Spring web**: creació de web, també RESTful, amb un model de MCV. Es la dependència que du un servidor apache inclòs dins el paquet.
- **Spring Data JPA**: Anotacions per a la persistència de dades a SQL amb l'ajuda del paquet hibernate.
- **Spring security**: Control d'accés a l'aplicació.

Maven 3.6.3[2]: eina per a gestió i construcció de projectes java, on principalment s'usarà per afegir les dependències necessàries al nostre projecte.

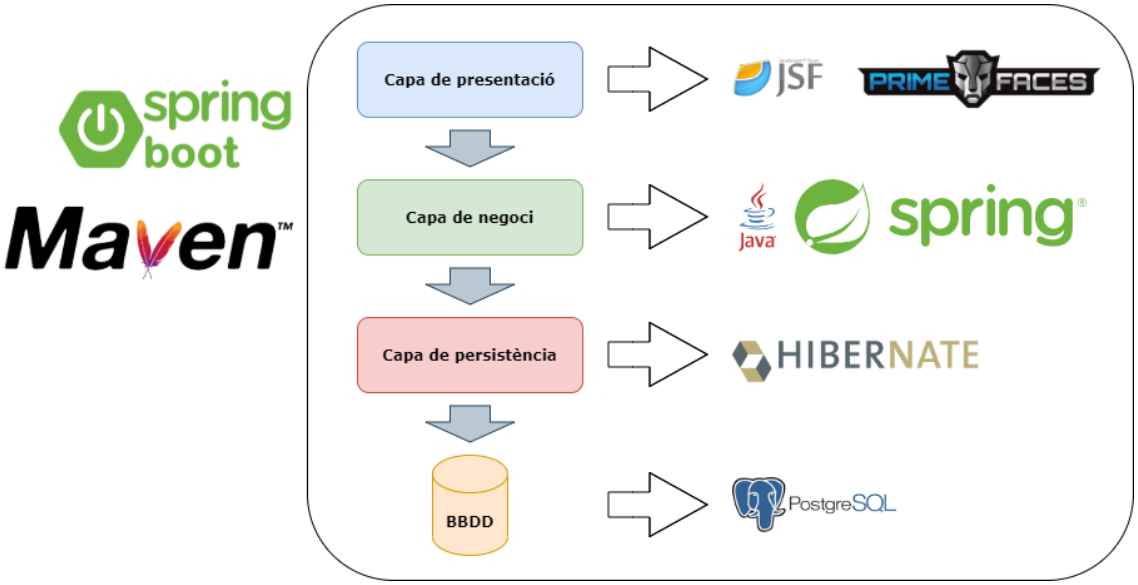
Primefaces 8.0[3]: biblioteca de components per a JavaServer Faces (JSF) de codi obert, on aplica una nova capa de components per a l'ajuda de mostrar i introduïa la informació al navegador.

Java SE Development Kit 8[4]: Versió java per al desenvolupament de l'aplicació.

Eclipse IDE Versió 2020-12[5]: aplicació per a l'edició i compilació per al desenvolupament de l'aplicació.

- S'ha afegit les **Spring Tools 4** per a poder controlar i compilar l'aplicació per així poder fer les proves.

- **PostgreSQL Versió 12**[6]: Gestió de base de dades relacional de codi obert.
- **GIT**: control de versions de codi obert
  - **Github.com**: servidor per a pujar el codi dels diferents canvis a la fase de desenvolupament.
  - **Git extensions**[7]: aplicació instal·lada al ordinador client on ajuda de forma visual la sincronització de codi entre el ordinador local de desenvolupament i el servidor.
- **Heroku.com**: Servidor de proves per a poder realitzar proves de l'aplicació. També es pot realitzar la instal·lació de les taules PostgreSQL.



Il·lustració 3: Diagrama de tecnologies

Il·lustració 4: Logotip GitHub

Control de versions

https://github.com/jaumet79/borsestreball.git

Il·lustració 5: Logotip Heroku

Entorn de proves

https://borsestreball.herokuapp.com

2. Anàlisi de requisits

2.1. Requeriments

Des del departament de recursos humans han de poder crear tantes borses de treball com siguin necessàries, però nomes pot haver una activa per categoria laboral.

Com a dades necessàries per definir la borsa es troba, una descripció, categoria laboral, la data de publicació a diferents entorns (internet, prensa, etc), data inici procés.

Per a cada borsa s'ha de definir una sèrie de criteris per a calcular la puntuació final. S'ha de definir si el criteri es un mèrit o un requeriment, on en el cas de no presentar-ho quedaria exclòs directament. També s'ha d'indicar una descripció, una puntuació per unitat del criteri i una puntuació màxima.

La definició de la borsa i els seus criteris nomes les podrà fer els usuaris amb el rol administrador.

Les borses han de passar per distintes fases:

- **Sol·licituds** > introducció de sol·licituds, i comprovació si es admès o exclòs segons documentació aportada.
- **Esmena sol·licituds** > temps per a esmenar alguna errada o reposició de la documentació necessària, però no es poden afegir noves sol·licituds
- **Mèrits** > afegir unitats de mèrits i càlculs segons baremació a les sol·licituds presentades i admeses.
- **Finalitzada** > borsa que no es pot modificar, però oberta per a cridar a nous treballadors.
- **Tancada** > ja no es pot usar aquesta borsa de treball.

El canvi de fase l'ha de definir un usuari amb rol administrador.

La gestió de persones a la borsa la pot fer un usuari amb rol Editor (el rol Administrador també).

Per als criteris que cada persona presenti, ha d'indicar el valor d'unitats on es multiplicarà per el valor del criteri per unitat. On la suma de tots aquests valors, faran la puntuació total de la persona a la borsa indicada.

Una vegada finalitzat el procés, s'ha de mostrar la llista definitiva de la gent admesa per ordre de puntuació.

Es defineix que tots els usuaris de l'aplicació utilitzaran un ordinador de sobretaula amb el navegador Chrome per defecte.

Es demana que l'aplicació sigui multi idioma, on per ara només es definirà en català i castellà, però amb la facilitat de poder afegir nous idiomes.

Els sol·licitants podran fer consulta de les seves dades amb el seu usuari i la seva data de naixement.

2.2. Rols i permisos

A continuació definim els distints rols definits a l'aplicació:

- **Admin**: Gestor dels usuaris que utilitzaran l'aplicació i encarregat de crear les borses amb els seus criteris. També pot fer las funcions d'editor i consultor.
- **Editor**: Encarregat d'afegir i gestionar les sol·licituds a l'aplicació. També pot fer la funció de consultor.
- **Consultor**: rol donat a la persona sol·licitant per a poder consultar l'estat de la seva participació a la borsa.

2.3. Requeriments funcionals

A continuació es mostren les funcionalitats extretes de l'anàlisi dels requeriments, amb un resum dels rols, explicats al punt anterior, que poden accedir a cada funcionalitat.

Codi	Requisit	Admin	Editor	Consultor
R1	Identificar-se a l'aplicació	x	x	x
R2	Sortir de l'aplicació	x	x	x
R3	Gestió usuaris (alta, baixa, modificació i consulta)	x		
R4	Gestionar categories (alta, baixa i modificació)	x		
R5	Gestionar borses (alta, baixa i modificació)	x		
R6	Canviar fase borsa	x		
R7	Gestionar criteris (alta, baixa i modificació)	x		
R8	Consultar borses	x	x	
R9	Gestionar persones (alta, baixa, modificació i consulta)	x	x	

R10	Assignar persona a borsa	x	x	
R11	Eliminar persona de borsa	x	x	
R12	Excloure persona de borsa	x	x	
R13	Gestionar mèrits a persona	x	x	
R14	Obtenir llista persones a borsa	x	x	
R15	Consultar dades personals selecció	x	x	x

2.4. Requeriments no funcionals

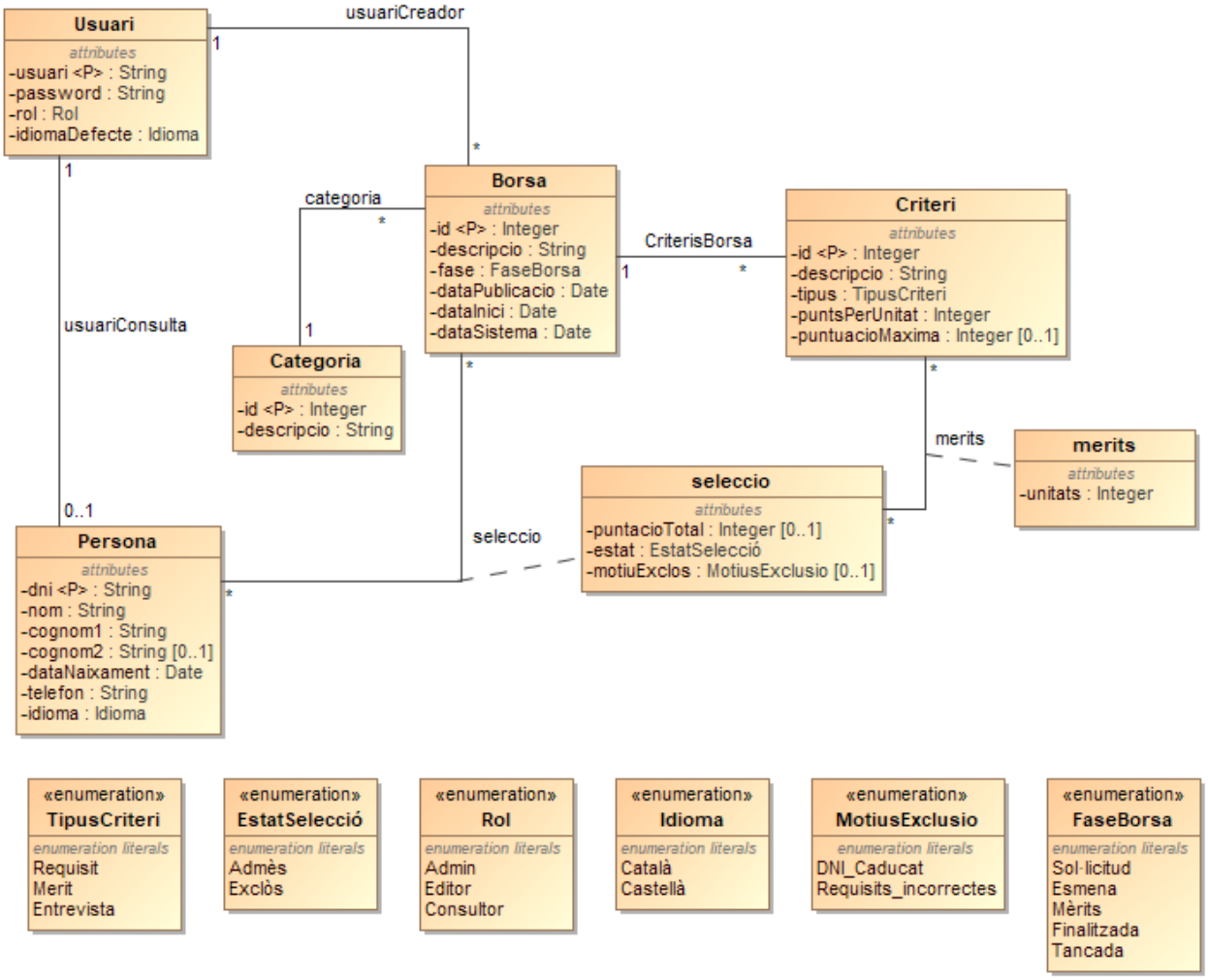
S'han detectat dos requisits no funcionals.

Codi	**Requisit no funcional**
N1	Optimitzat per a Google Chrome
N2	Aplicació multi-idioma (fase 1: català-castellà)
N3	Encriptació de la contrasenya dins la base de dades

3. Disseny de l'aplicació

3.1. Disseny conceptual, diagrama de classes UML

A partir de la informació recopilada procedim a fer un model de dades des del punt de vista de la informació, on descriu amb detall la informació de que tracta el sistema, incloent estructures, dades i restriccions trobades segons l'anàlisi inicial.



Il·lustració 6: Diagrama de classes UML

- La classe **Borsa** es la arrel de l'aplicació, on es defineixen les dades generals de la borsa, com pot ser l'estat, dates i una explicació. Per a següents fases es pot afegir més informació, com podria ser localització del lloc de feina, tipus de contracte, etc.
- La classe **Usuari** es la classe que gestiona els usuaris que poden accedir a l'aplicació, i amb quin nivell de rol accedeixen.
- La classe **Persona** es on es donen d'alta totes les persones que volen accedir a una borsa. Per a cada persona es definirà un **usuari** per a poder accedir amb un rol de consulta per a poder veure les seves dades.
- La classe **Categoria** emmagatzema les distintes categories laborals de les quals es poden crear borses.
- Per a cada borsa es definiran els seus **criteris** per indicar la puntuació de cada mèrit presentat i poder calcular una puntuació per persona.

- Dins la classe **Selecció** es gestionen les dades de la unió persona-borsa, on s'indica la puntuació per a cada sol·licitud segons el càlcul dels seus mèrits segons els criteris indicats.
- A la unió **Mèrits**, indica la puntuació definida entre la persona de la borsa i el criteri indicat.

3.2. Model lògic relacional

A continuació es fa una relació de les taules que es generaran a partir del disseny conceptual.

Usuari(usuari, **password**, **rol**, **idiomaDefecte**)

Persona(dni, **nom**, **cognom1**, cognom2, **dataNaixament**, **telèfon, idioma**, usuariConsulta)
{usuariConsulta} is foreign key to Persona

Categoria(id, **descripcio**)

Borsa(id, **descripcio**, **fase, dataPublicacio**, **dataInici**, **dataSistema**, **categoria**, **usuariCreador**)
{categoria} is foreign key to Categoria
{usuariCreador} is foreign key to Usuari

Criteri(id, **descripcio**, **tipus**, **puntsPerUnitat**, puntuacioMaxima)

CriterisBorsa(borsa, criteri)
{borsa} is foreign key to Borsa
{criteri} is foreign key to Criteri

Seleccio(borsa, persona, puntuacioTotal, **estat,** motiuExclos)
{borsa} is foreign key to Borsa
{persona} is foreign key to Persona

Merits(borsa, persona, criteri, **unitats**)
{borsa} is foreign key to Borsa
{persona} is foreign key to Persona
{criteri} is foreign key to Criteri

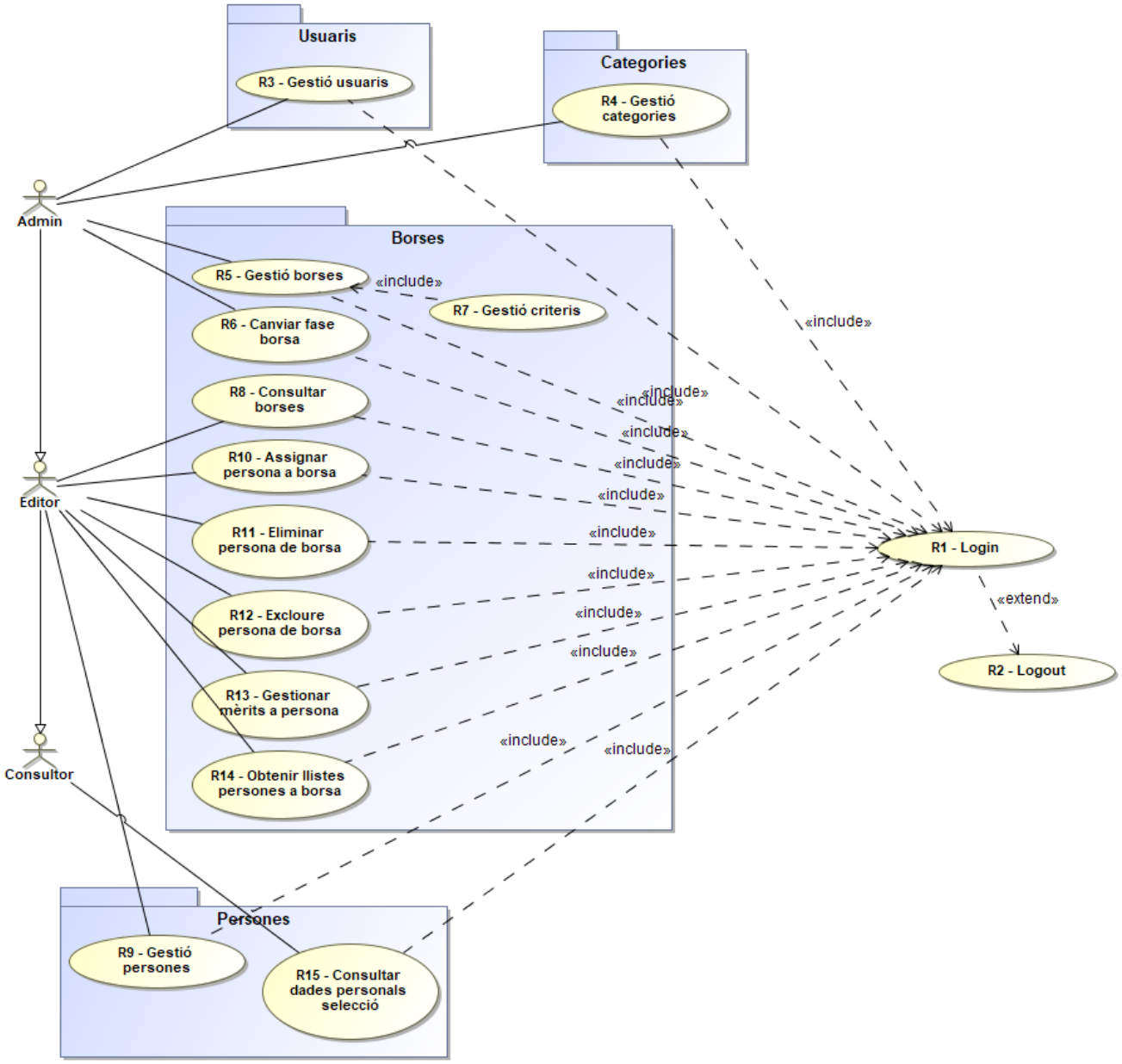
Nota:

- Les claus primàries estan subratllades amb línia contínua.

- Els atributs que no formen part de la clau i que no poden ser nuls apareixen en **negreta**.

3.3. Casos d'us

A continuació es mostra l'esquema dels casos d'us segons l'accés d'un rol determinat i les seves herències de permisos.



Il·lustració 7: Diagrama de casos d'us

3.4. Especificació casos d'us

Es mostra un anàlisi dels diferents casos d'us i les seves variants. Per a cada un s'ha definit l'usuari que les pot executar i si ha alguna variació segons el rol de l'usuari que accedeix.

- **Cas d'ús: R1 - Identificar-se a l'aplicació**
- **Actors**: Tots.
- **Descripció**: L'usuari indica el nom d'usuari i la seva contrasenya on el sistema retorna el rol que se l'hi ha assignat.
- **Condició prèvia**: Usuaris donats d'alta a BBDD.
- **Cas d'èxit**:
    1. Usuari indica nom i contrasenya
    2. Es comprova que l'usuari existeix a BBDD
    3. Retorna la informació de l'usuari amb el rol assignat.
- **Cas alternatiu 1**:
    2a. Usuari existeix però contrasenya incorrecte
    3a. Retorna error indicant contrasenya incorrecte.
- **Cas alternatiu 2**:
    2b. Usuari no existeix
    3b. Retorna error indicant usuari incorrecte.

- **Cas d'ús: R2 – Sortir de l'aplicació**
- **Actors**: Tots.
- **Descripció**: L'usuari tanca la sessió i surt de l'aplicació.
- **Condició prèvia**: Usuari identificat correctament.
- **Cas d'èxit**:
    1. Usuari surt de l'aplicació

- **Cas d'ús: R3 – Gestió d'usuaris**
- **Actors**: Administrador.
- **Descripció**: Afegir/Modificar usuari a l'aplicació
- **Condició prèvia**: Estar identificat com a usuari Administrador (un usuari de tipus Administrador es generarà de forma automàtica amb l'aplicació)
- **Cas d'èxit**:
    1. Usuari indica les dades d'un usuari.
    2. El sistema comprova si ja existeix a BBDD, i no en troba.
    3. Es guarda l'usuari i contrasenya a la BBDD
- **Cas alternatiu 1:**
    2a. El sistema comprova si ja existeix a BBDD i troba una duplicitat
    3a. Retorna error i no guarda a BBDD

- **Cas d'ús: R3 – Gestió d'usuaris**
- **Actors**: Administrador.
- **Descripció**: Donar de baixa usuari de l'aplicació
- **Condició prèvia**: Estar identificat com a usuari Administrador.
- **Cas d'èxit**:
    1. Usuari Administrador indica l'usuari a donar de baixa.
    2. Es modifica l'usuari indicant usuari donat de baixa (baixa lògica)

- **Cas d'ús: R4 – Gestió de categories**
- **Actors**: Administrador.
- **Descripció**: Afegir/Modificar categories a l'aplicació
- **Condició prèvia**: Estar identificat com a usuari Administrador.
- **Cas d'èxit**:
    1. Usuari indica les dades d'una categoria.
    2. El sistema comprova si ja existeix a BBDD, i no en troba.
    3. Es guarda la categoria a BBDD.
- **Cas alternatiu 1:**

    2a. El sistema comprova si ja existeix a BBDD i troba una duplicitat

    3a. Retorna error i no guarda a BBDD

- **Cas d'ús: R4 – Gestió de categories**
- **Actors**: Administrador.
- **Descripció**: Eliminar categoria de l'aplicació
- **Condició prèvia**: Estar identificat com a usuari Administrador.
- **Cas d'èxit**:
    1. Usuari indica la categoria a eliminar
    2. Sistema comprova si la categoria està assignada a alguna borsa, si no està assignada,
    3. La categoria es eliminada de BBDD
- **Cas alternatiu 1:**

    2a. Sistema comprova si la categoria està assignada a alguna borsa, si està assignada,

    3a. Retorna error i elimina la categoria de BBDD

- **Cas d'ús: R5 – Gestió de borses**
- **Actors**: Administrador.
- **Descripció**: Alta i modificació de borsa
- **Condicions prèvies**:
    1. Estar identificat com a usuari Administrador.
    2. Tenir la categoria a afegir a la borsa donada d'alta.

- **Cas d'èxit**:
    1. Usuari indica les dades a afegir.
    2. El sistema comprova si ja existeix alguna borsa de la categoria indicada no està finalitzada o tancada.
    3. Si no n'hi ha cap, la borsa es guardada a la BBDD.
- **Cas alternatiu 1:**

    3a. Si n'hi ha cap borsa oberta de la mateixa categoria, retorna error i no guarda a BBDD

- **Cas d'ús: R5 – Gestió de borses**
- **Actors**: Administrador.
- **Descripció**: Eliminació de borsa
- **Condicions prèvies**:
    1. Estar identificat com a usuari Administrador.
    2. Tenir la borsa donada d'alta.
- **Cas d'èxit**:
    1. Usuari indica la borsa a eliminar.
    2. El sistema comprova si te alguna selecció de persones donada d'alta, si no en te.
    3. Es comprova si te criteris donats d'alta, si es així, els elimina automàticament a la mateixa transacció.
    4. Elimina la borsa.
- **Cas alternatiu 1:**

    2a. El sistema comprova si te alguna selecció de persones donada d'alta, i si en te.

    3a. Retorna error i no elimina la borsa de BBDD.

- **Cas d'ús: R6 – Canviar fase borsa**
- **Actors**: Administrador.
- **Descripció**: Es canvia la fase de borsa.
- **Condicions prèvies**:
    1. Estar identificat com a usuari Administrador.
    2. Tenir la borsa donada d'alta.
- **Cas d'èxit**:
    1. Usuari indica la borsa a modificar fase.
    2. El sistema comprova a quina fase es troba, i només deixa escollir una fase posterior.
    3. Es guarda la nova fase a BBDD.

- **Cas d'ús: R7 – Gestió de criteris**
- **Actors**: Administrador.
- **Descripció**: Afegir/Modificar criteri
- **Condició prèvia**:
    - Estar identificat com a usuari Administrador.
    - Tenir la borsa a indicar criteris donada d'alta.
- **Cas d'èxit**:
    1. Usuari selecciona una borsa per a afegir criteris.
    2. Usuari indica les dades de criteri
    3. Sistema comprova si hi ha algun registre duplicat per la mateixa descripció i borsa, i no en troba
    4. Si la modificació de criteri ha modificat el valor de puntsPerUnitat, es fa un re-càlcul de totes les persones assignades a la borsa, i es modifica el valor puntuació total de la selecció.
    5. El criteri es registrat a la BBDD
- **Cas alternatiu 1:**

    2a. Sistema comprova si hi ha algun registre duplicat per la mateixa descripció i borsa, i si en troba

    3a. Retorna error i no guarda a BBDD

- **Cas d'ús: R7 – Gestió de criteris**
- **Actors**: Administrador.
- **Descripció**: Eliminar criteri
- **Condició prèvia**:
    - Estar identificat com a usuari Administrador.
    - Tenir criteris donats d'alta assignats a una borsa
- **Cas d'èxit**:
    1. Usuari selecciona una borsa per a modificar els criteris.
    2. Usuari indica el criteri a eliminar.
    3. Sistema comprova si hi ha alguna persona de la borsa a la qual se li ha donat puntuació en base a aquest criteri, i no en troba.
    4. El criteri es eliminat de la BBDD

- **Cas alternatiu 1:**

    3a. Sistema comprova si hi ha alguna persona de la borsa a la qual se li ha donat puntuació en base a aquest criteri, i en troba.

    3a. Retorna error i no guarda a BBDD

- **Cas d'ús: R8 – Consulta borses**
- **Actors**: Administrador i Editor.
- **Descripció**: Consulta llista i borsa individual
- **Condicions prèvies**:
    - Estar identificat com a usuari Administrador o Editor
    - Tenir la borsa donada d'alta.
- **Cas d'èxit**:
    1. Usuari indica la borsa a consultar.
    2. Retorna la informació de la borsa.
- **Cas alternatiu 1:**

    1a. Usuari no comprova la borsa a consultar

    2a. Retorna una llista de totes les borses donades d'alta.

- **Cas d'ús: R9 – Gestió de persones**
- **Actors**: Administrador i Editor.
- **Descripció**: Afegir i modificar persones al sistema.
- **Condicions prèvies**:
    - Estar identificat com a usuari Administrador o Editor
- **Cas d'èxit**:
    1. Usuari indica el DNI de la persona a afegir.
    2. El sistema cerca si ja existeix el DNI a la BBDD.
    3. Si no existeix, l'usuari indica les dades personals.
    4. El sistema guarda correctament la persona a la BBDD.
    5. El sistema crea un registre a la taula usuari on usuari=DNI, password=data de naixement, rol=CONSULTOR, idiomaDefecte=idiomaPersona
- **Cas alternatiu 1:**

    3a. Si existeix, el sistema retorna les dades de la persona.

    4a. L'usuari indica alguna modificació de les dades de la persona.

    5a. El sistema guarda correctament la persona a BBDD.

- **Cas d'ús: R10 – Assigna persona a borsa**
- **Actors**: Administrador i Editor.
- **Descripció**: Assignar persona a una borsa determinada
- **Condicions prèvies**:
    - Estar identificat com a usuari Administrador o Editor
    - Tenir la borsa donada d'alta al sistema.
    - Tenir la persona donada d'alta al sistema.
    - La borsa ha d'estar a la fase de sol·licituds.
- **Cas d'èxit**:
    1. Usuari selecciona borsa a la qual es vol afegir una persona.
    2. Usuari indica el DNI de la persona afegir.

3. Si existeix, afegeix la persona a la borsa.

- **Cas alternatiu 1:**

  3a. Si no existeix, el sistema retorna un error i no afegeix cap persona a la borsa.

- **Cas d'ús: R11 – Eliminar persona de borsa**
- **Actors**: Administrador i Editor.
- **Descripció**: Eliminar la assignació d'una persona d'una borsa determinada
- **Condicions prèvies**:
  - Estar identificat com a usuari Administrador o Editor
  - La persona ha d'estar assignada a la borsa.
  - La borsa ha d'estar a la fase de sol·licituds.
- **Cas d'èxit**:
  1. Usuari selecciona borsa de la qual es vol eliminar a la persona.
  2. Usuari indica el DNI de la persona a eliminar
  3. Si existeix, comprova si te mèrits assignats a aquesta persona
  4. Si te mèrits, els hi elimina automàticament
  5. El sistema elimina la assignació de la persona a la borsa indicada.
- **Cas alternatiu 1:**

  3a. Si no existeix, el sistema retorna un error indicant que no s'ha trobat la persona.

- **Cas d'ús: R12 – Excloure persona de borsa**
- **Actors**: Administrador i Editor.
- **Descripció**: Procés d'exclusió d'una persona d'una borsa determinada
- **Condicions prèvies**:
  - Estar identificat com a usuari Administrador o Editor
  - Tenir la borsa donada d'alta al sistema.
  - Tenir la persona donada d'alta al sistema.
  - Tenir la persona assignada a la borsa i amb estat admès.
  - La borsa ha d'estar a la fase de sol·licituds o esmena de sol·licituds.
- **Cas d'èxit**:
  1. Usuari selecciona borsa de la qual excloure a la persona.
  2. Usuari selecciona la persona a la qual es vol excloure.
  3. S'ha d'indicar un motiu d'exclusió.
  4. Es modifica la puntuació a 0.

- **Cas d'ús: R13 – Afegir mèrits a persona**
- **Actors**: Administrador i Editor.
- **Descripció**: Indicar mèrits a una persona dins una borsa determinada.
- **Condicions prèvies**:

    - Estar identificat com a usuari Administrador o Editor
    - Tenir la borsa donada d'alta al sistema.
    - Tenir la persona donada d'alta al sistema.
    - Tenir la persona assignada a la borsa.
    - La borsa ha d'estar a la fase de mèrits.
- **Cas d'èxit**:
    1. Usuari selecciona borsa a la qual es vol afegir una persona.
    2. Usuari selecciona la persona a la qual es volen modificar els mèrits.
    3. S'introdueixen els punts per cada mèrit
    4. Es pitja el botó desar
    5. Es fa un càlcul multiplicant criteri.puntsPerUnitat x merits.unitat, i si el càlcul es major al camp puntuacióMaxima es modifica per aquest valor
    6. Es guarden els valors a la taula de selecció.

- **Cas d'ús: R14 – Obtenir llista persones a borsa**
- **Actors**: Administrador i Editor.
- **Descripció**: Obtenir llista de les persones assignades a una borsa, amb filtre d'admès i exclòs, i distints criteris d'ordenació (puntuació, cognom)
- **Condicions prèvies**:
    - Estar identificat com a usuari Administrador o Editor
    - Tenir la borsa donada d'alta al sistema.
- **Cas d'èxit**:
    1. Usuari selecciona la borsa de la qual vol mostrar la llista de persones.
    2. Usuari selecciona els filtres per a mostrar llista i criteris d'ordenació.
    3. Es mostra llista per pantalla amb l'opció de poder treure versió per imprimir.

- **Cas d'ús: R15 – Consultar dades personals selecció**
- **Actors**: Administrador, Editor i Consultor.
- **Descripció**: Consultar les dades personals i de assignacions a llistes amb una vista resum.
- **Condicions prèvies**:
    - La persona ha d'estar donada d'alta al sistema.
- **Cas d'èxit**:
    1. Usuari Consultor s'identifica al sistema.
    2. El sistema cerca la informació de l'usuari consultor i comprova a les borses que ha participat i amb fase finalitzada o tancada.
    3. Se mostra una pantalla informativa amb el resum de les seves assignacions a llistes, estats, posició i puntuacions finals.

- **Cas alternatiu 1:**

  1a. Usuari Administrador o editor s'identifiquen al sistema.

  2a. Usuari es dirigeix a pantalla de consulta.

  3a. Indica DNI de la persona a consultar.

  4a. El sistema cerca la informació de l'usuari comprovant a les borses que ha participat (sense tenir en compte l'estat)

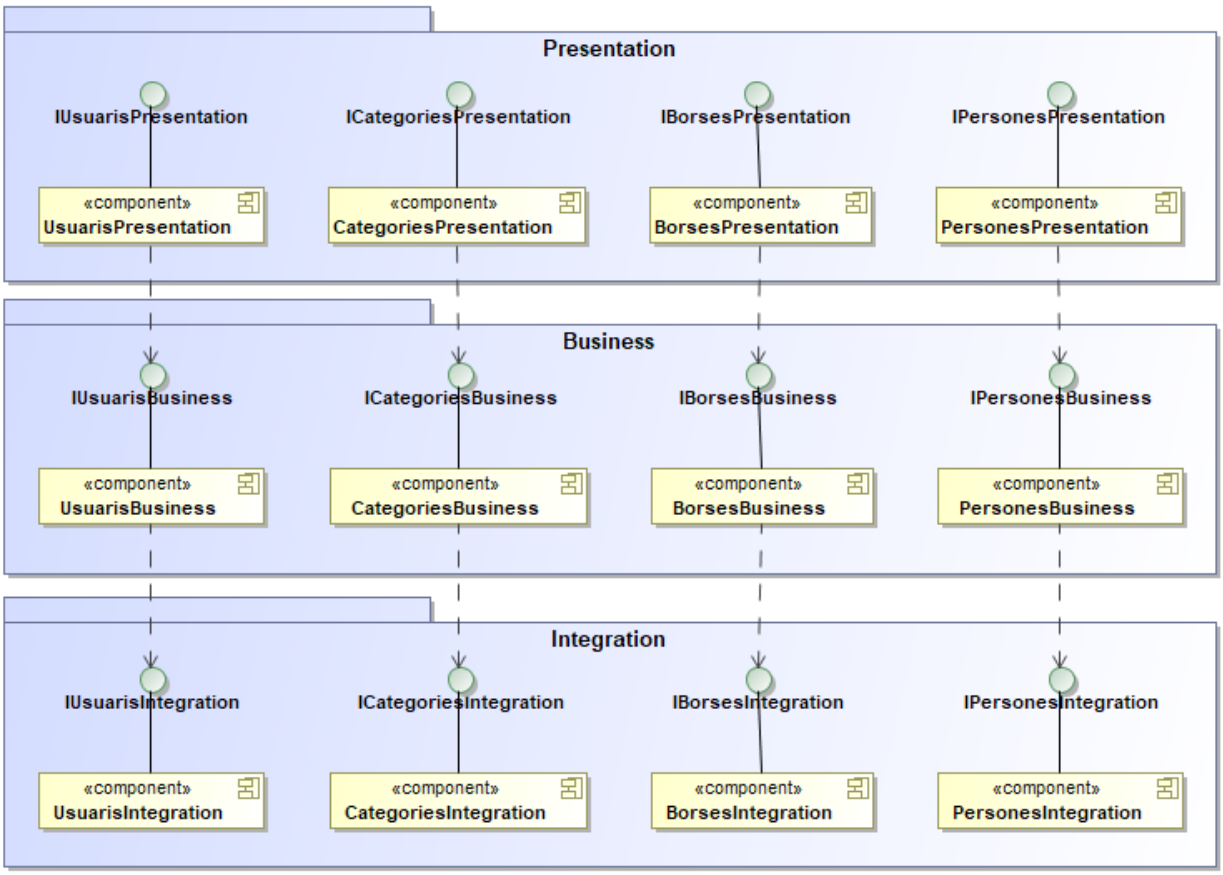
  5a. Mostra la pantalla informàtica com el cas d'èxit.

- **Cas alternatiu 2:**

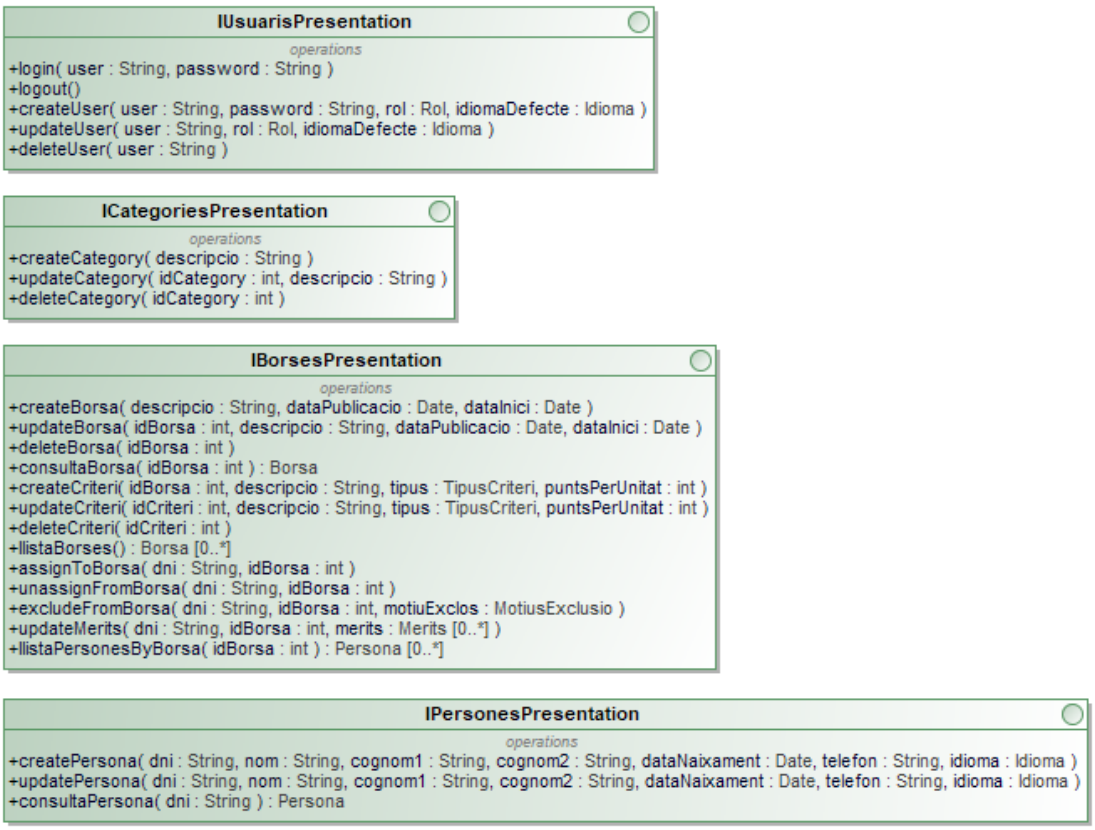
  4b. Si no troba el DNI indicat, mostra error de persona no trobada.

3.5. Diagrama de components

S'utilitzarà una arquitectura de tres capes model-vista-controlador on es crearan 4 grans grups de control: **usuaris**, encarregada de control d'usuaris i gestió, **categories**, **borses**, que seria la que controlaria la gran carrega de feina de l'aplicació i **persones.**



Il·lustració 8: Diagrama de components



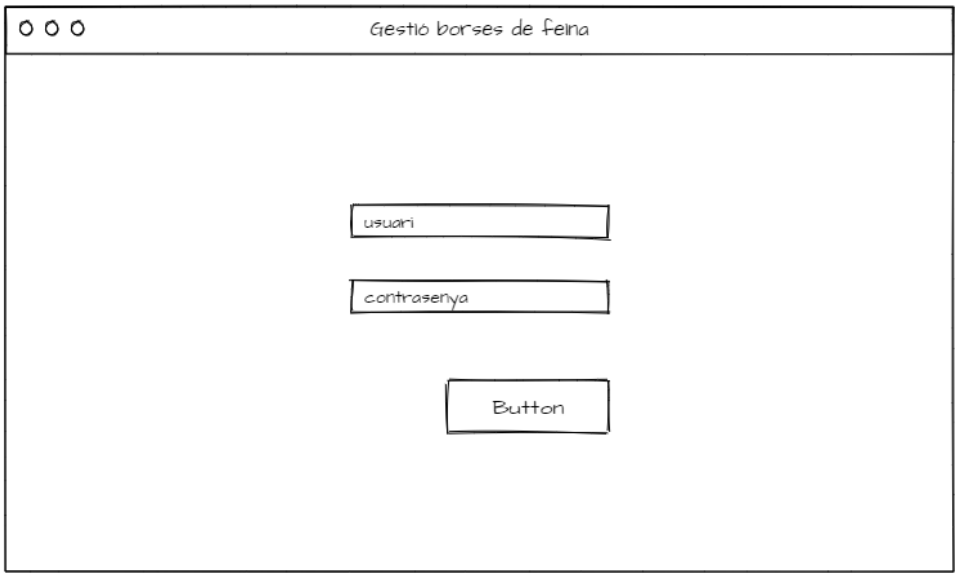
Il·lustració 9: Serveis dels components

Només es mostra la capa de presentació com exemple, la capa de negoci es igual a la de presentació, i la d'integració pot variar degut a l'ajuda que pot aportar la capa hibernate damunt jpa de java.

3.6. Disseny

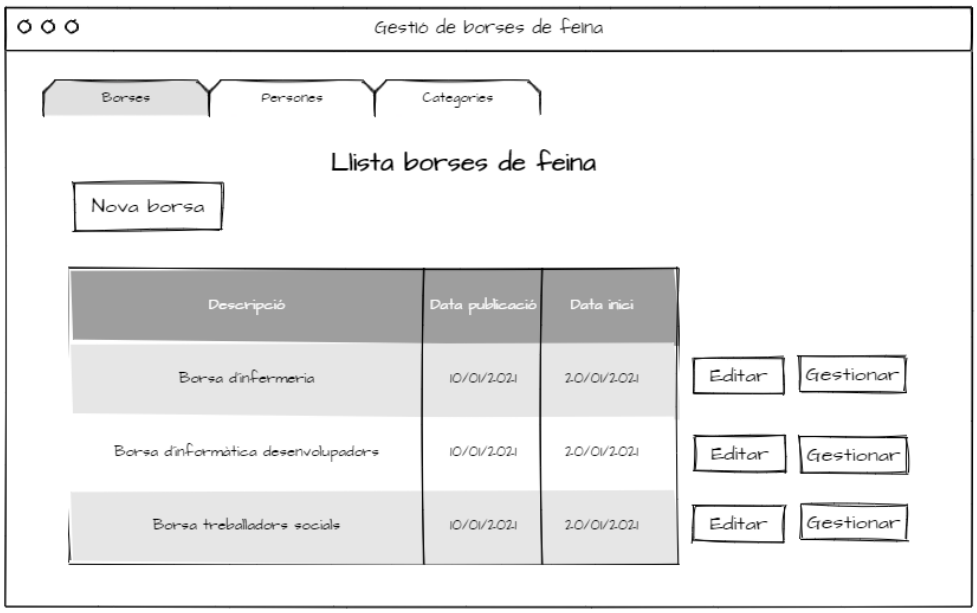
En aquest apartat s'ha fet un petit esquema de les pantalles per a formar un exemple per al disseny de las pagines a l'hora del desenvolupament.

Per entrar a l'aplicació es necessari un usuari i contrasenya que gestionen els usuaris Administradors.



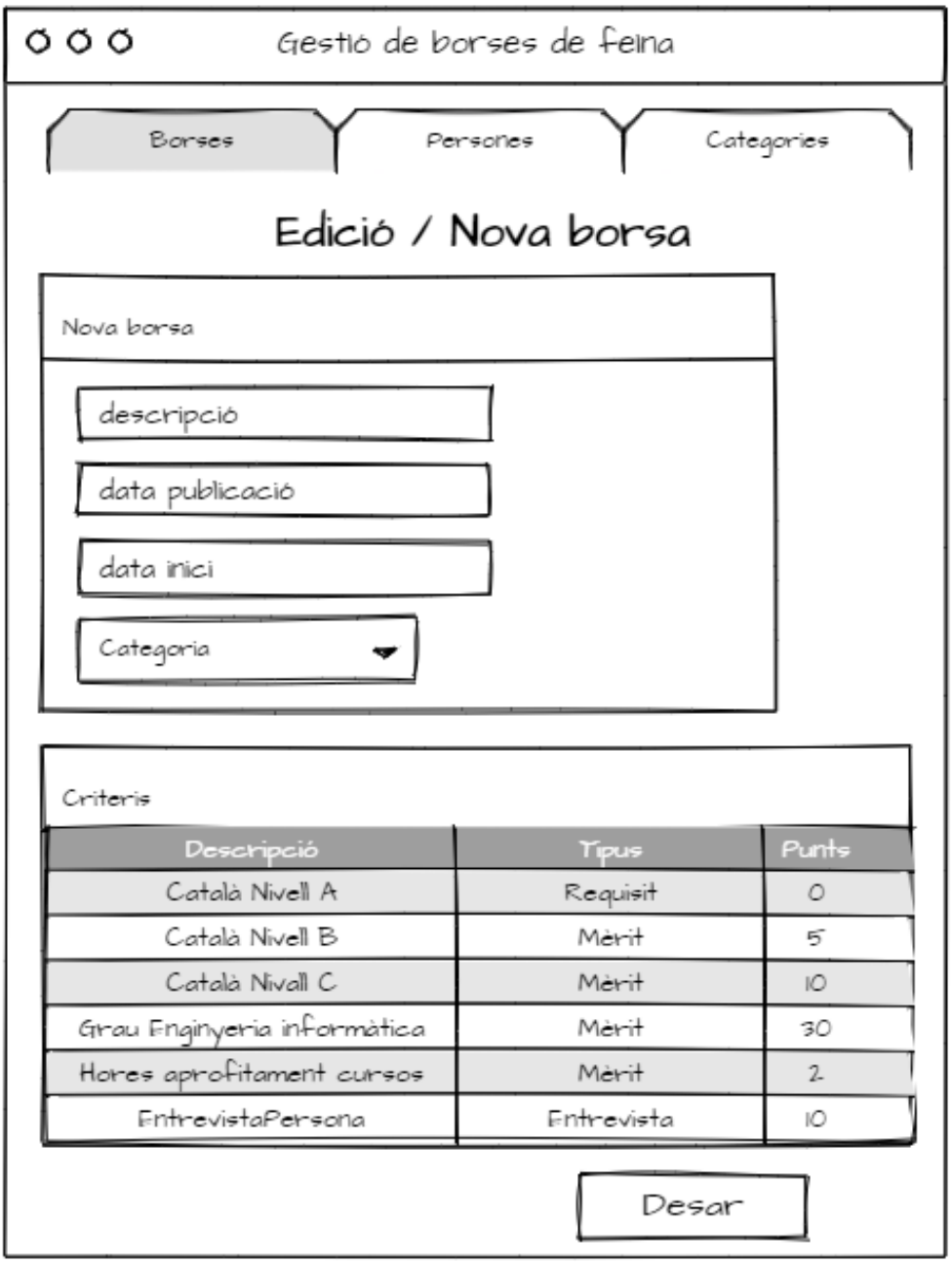
Il·lustració 10: Pantalla login

Una vegada introduits dins l'aplicació, a la pantalla principal ens mostren les borses ja donades d'alta amb les distintes opcions per a cada una.



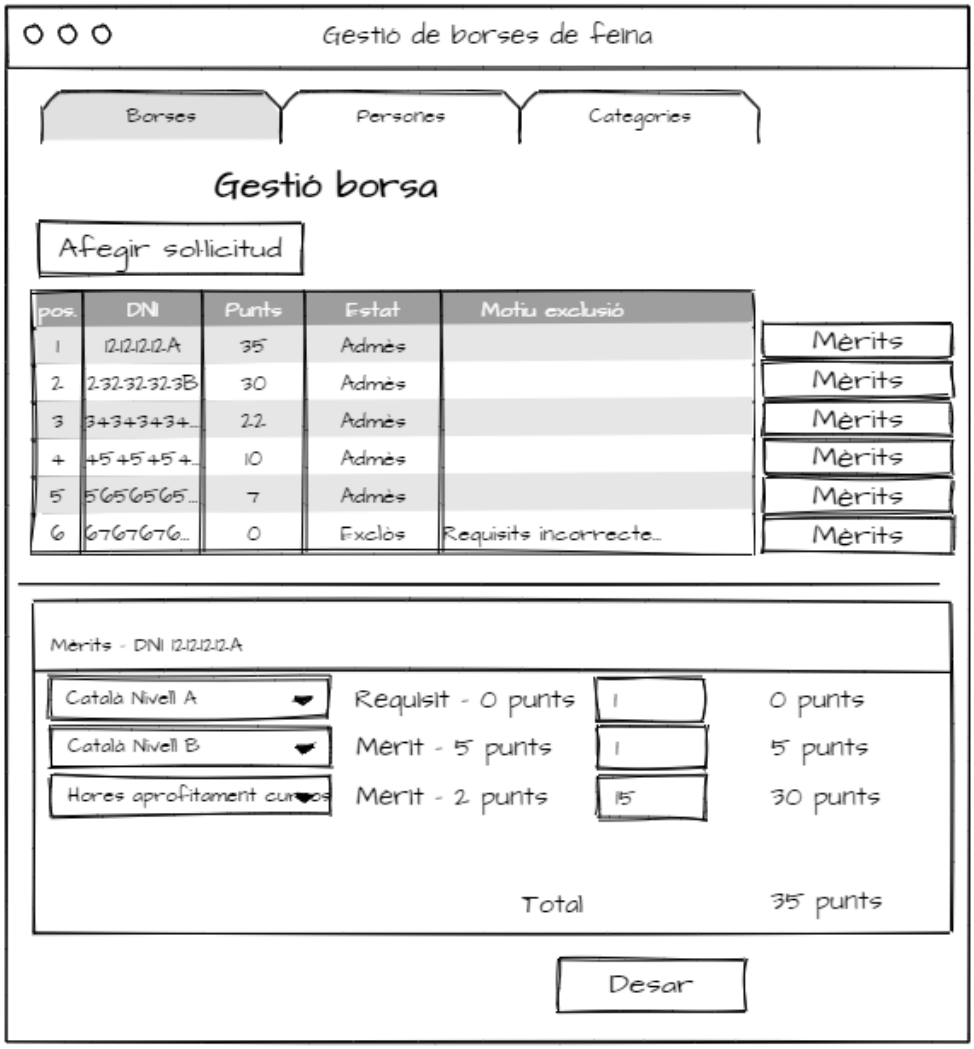
Il·lustració 11: Pantalla índex - llista borses de treball

A la següent pantalla es poden crear i modificar les borses de treball, indicant sobre el formulari principal les dades de la borsa, i a la taula inferior els distints criteris i puntuacions.



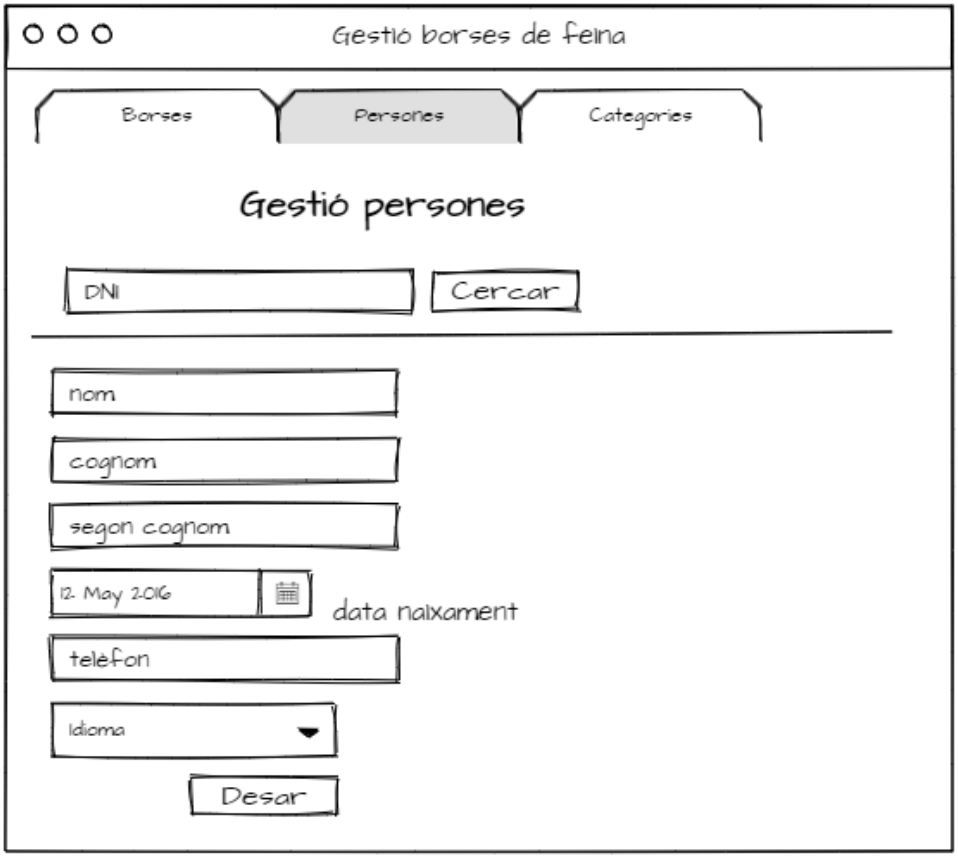
Il·lustració 12: Pantalla gestió borsa

Una vegada creada la borsa, a la següent pantalla es controlen les sol·licituds a afegir, així com els mérits aportats per la persona sol·licitant



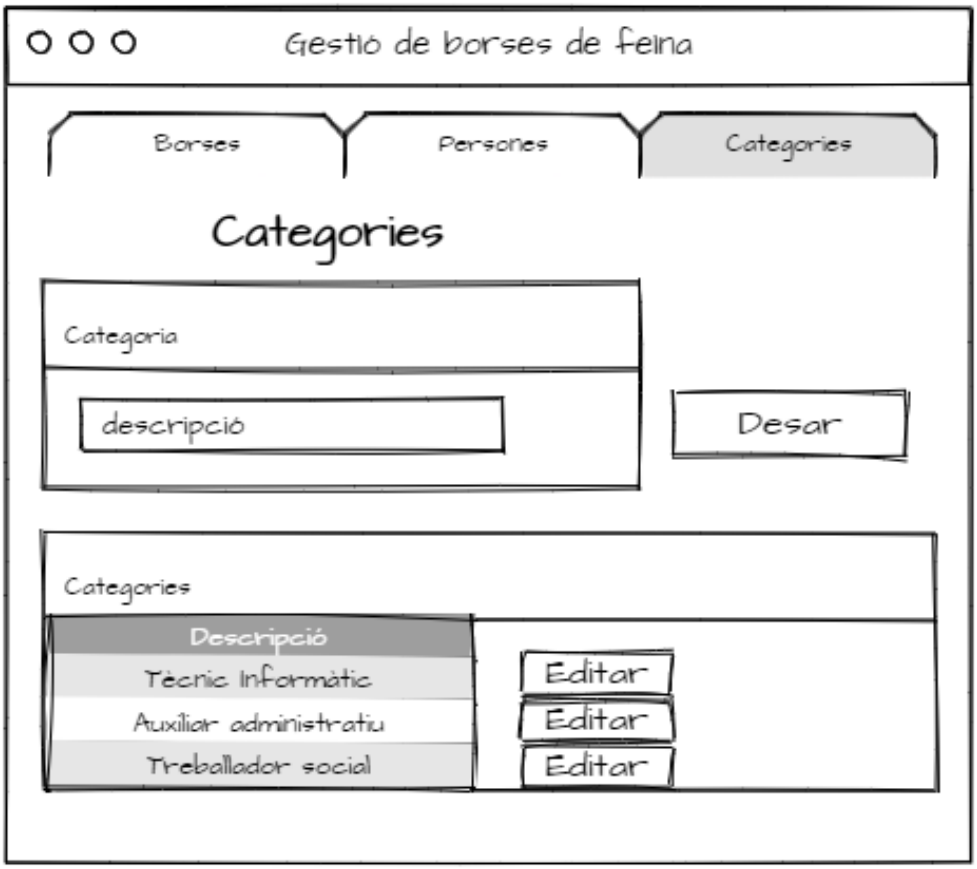
Il·lustració 13: Pantalla Inscripcions

Per a poder afegir sol·licituds a les borses de treball, primer s'han de registrar les persones a l'aplicació per a poder afegirles a distintes borses.



Il·lustració 14: Pantalla gestió de candidats

A la següent pantalla es defineixen les distintes categories laborals de les quals es poden crear borses



Il·lustració 15: Pantalla gestió de categories

4. Desenvolupament de l'aplicació

4.1. Preparació de l'entorn

Al final de la documentació s'ha afegit un annex per a poder instal·lar el necessari per a poder executar l'aplicació, per això es farà només un petit resum de les passes seguides per a posar en marxa el primer "hola mon" de l'aplicació:

El primer pas es instal·lar una versió de JDK de Java de la versió 8, en el nostre cas hem descarregat la versió 1.8.291 i s'han configurat les variables del sistema.

El següent, instal·lar la versió 13 de postgreSQL i crear una base de dades buida amb el nom "borsestreball".

A continuació instal·lar les llibreries maven utilitzades per, una vegada ben configurat, davallar totes les llibreries necessàries per a fer córrer l'aplicació. La versió utilitzada es la 3.8.1. També s'han de configurar unes variables al sistema.

Com a aplicació principal s'utilitza Spring Boot, ja que es una infraestructura creada per Spring on elimina la major part de la feina de configuració i es pot centrar mes en el desenvolupament. Encara que la configuració inicial es molt bàsica, afegir nova configuració es molt senzill, ja que està tot centralitzat a una classe especial de configuració.

Spring proporciona una eina configurable[8] per a poder generar una aplicació base, dintre de les opcions que es poden triar son: generar un projecte Maven o Gradle, també es pot configurar el llenguatge de programació entre Java, Kotlin i Groovy. Es pot triar també la versió de Spring Boot que es vol escollir, entre las darreres estables i betes.

L'eina també deixa triar l'empaquetat, com a Jar, amb un servidor web apache contingut a les llibreries, per a poder executar des de qualsevol equip sense tenir que instal·lar un servidor web. O l'empaquetat War per a poder afegir-ho a qualsevol servidor web apache.

I per acabar deixar triar entre les versions 8, 11 i 16 de Java.

En el moment de crear la aplicació, Spring et mostra una llista de "Eines de desenvolupament" on es poden trobar llibreries que es poden afegir a l'aplicació bàsica per a personalitzar-la a les necessitats pròpies de cada projecte. En el nostre cas hem afegit les següents:

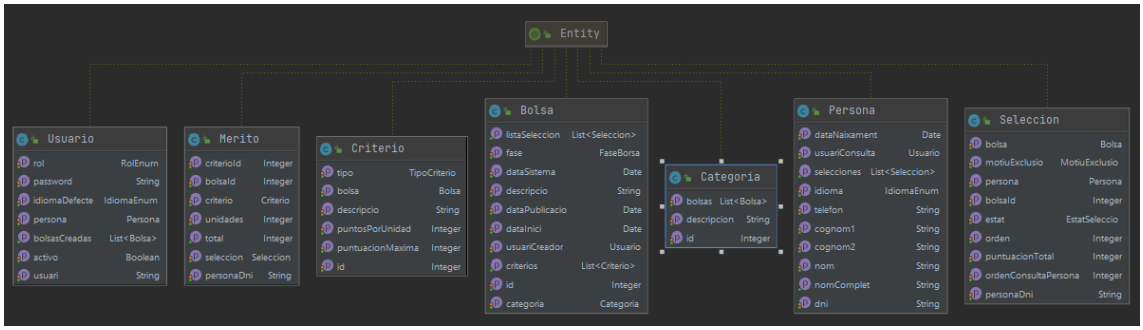
- **Spring Web**: construcció de web, RESTful, utilitzant Spring MVC. Aquesta llibreria du incorporada una versió de Apache Tomcat mínima per a poder executar l'aplicació en qualsevol moment sense necessitat de tenir un Tomcat instal·lat al sistema. Encara que també es pot crear un war per a poder afegir a un Tomcat ja existent.

- **Spring Data JPA**: capa de persistència per a SQL per a comunicació amb la base de dades, utilitzant llibreries de Hibernate.
- **Spring Security**: Capa altament personalitzable de control d'accés per a aplicacions Spring. Deixa configurar l'accés amb una BBDD i control d'encriptació de informació.
- A posteriori s'han hagut que anar afegint noves llibreries per a diferents utilitats, com per exemple:
    - Llibreria postgreSQL: necessària per a que la capa JPA pugi entendre la configuració de connexió a postGreSQL.
    - Primefaces i themes de primefaces: Llibreries d'interface gràfica per a la vista de JavaServer Faces i temes de visualització gratuïts.
    - També s'ha d'afegir la versió de JSF que es vol utilitzar, en el nostre cas, utilitzem la versió 2.2.20.

I amb totes aquestes llibreries ja tenim preparada la primera execució de l'aplicació.

4.2. Capa de persistència

Per a poder utilitzar la capa de persistència, el primer que s'han de crear es el model de dades.



Il·lustració 16: Esquema entitats

Cada model ha d'utilitzar etiquetes especials de configuració de persistència (javax.persistence.*) i de hibernate (org.hibernate.annotations.*)

Posarem un exemple de la entitat Bolsa, ja que es pot dir que la principal de l'aplicació i s'explicaran les anotacions utilitzades:

```
26 /**
27  * Entitat borsa
28  *
29  * @author Jaume
30  */
31 @Entity
32 public class Bolsa {
33
34     @Id
35     @GeneratedValue
36     private Integer id;
37
38     @Column(length = 50, nullable = false)
39     private String descripcio;
40
41     @Column(nullable = false, length = 15)
42     @Enumerated(EnumType.STRING)
43     private FaseBorsa fase = FaseBorsa.SOLICITUD;
44
45     private Date dataPublicacio;
46
47     private Date dataInici;
48
49     private Date dataSistema;
50
51     @OneToMany(mappedBy = "bolsa", fetch = FetchType.EAGER, cascade = CascadeType.ALL, orphanRemoval = true)
52     @OrderBy(clause = "tipo ASC")
53     private List<Criterio> criterios;
54
55     @ManyToOne
56     @JoinColumn(name = "categoria", referencedColumnName = "id", nullable = false)
57     private Categoria categoria;
58
59     @ManyToOne
60     @JoinColumn(name = "usuariCreador", referencedColumnName = "usuari", nullable = false)
61     private Usuario usuariCreador;
62
63     @OneToMany(mappedBy = "bolsa")
64     @LazyCollection(LazyCollectionOption.FALSE)
65     @OrderBy(clause = "estat, puntuacionTotal DESC")
66     private List<Seleccion> listaSeleccion;
```

Il·lustració 17: Exemple entitat Borsa

Per a que Spring sàpiga que es una entitat de base de dades, s'ha d'identificar amb l'anotació @Entity, a partir d'allà es podran utilitzar les altres anotacions especials per a cada camp.

@Id @GeneratedValue > Autogeneració de l'identificador únic de la entitat.

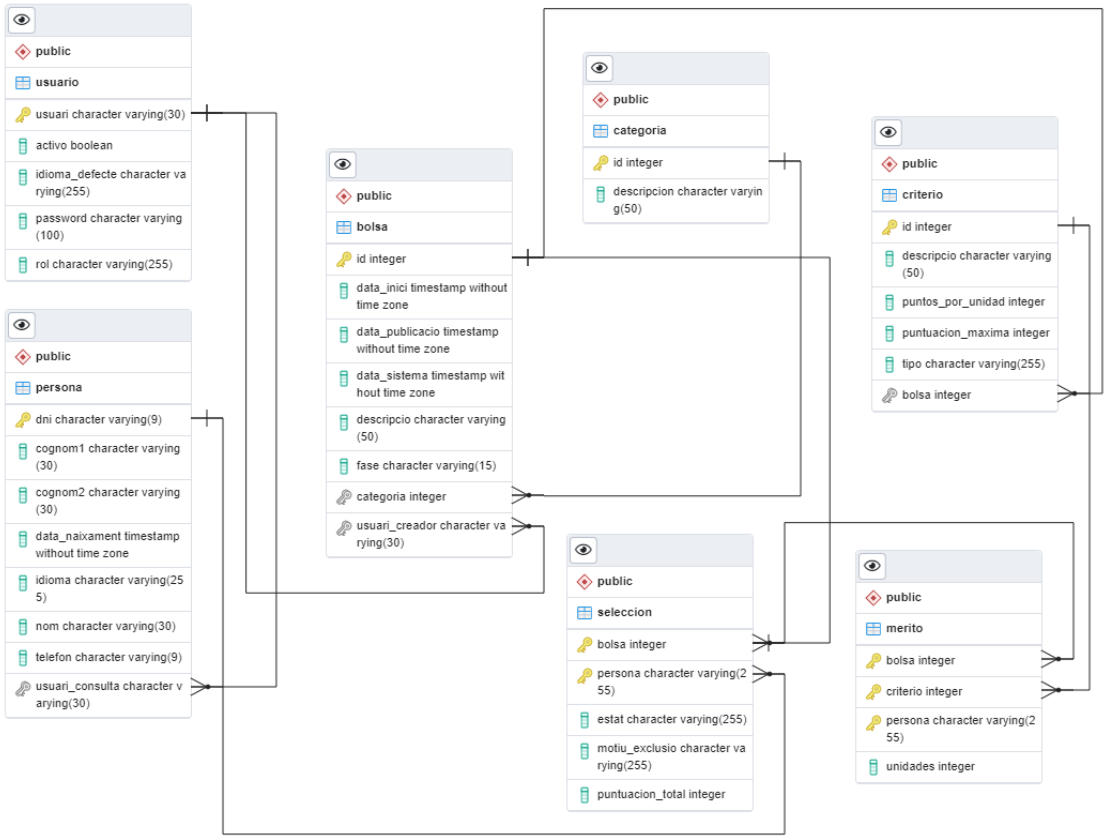
@Column > propietats de la columna, com pot ser la grandària, si accepta valors nuls, nom, numèric amb una certa precisió.

@Enumerated > quan la columna de la taula utilitza valors d'un enumerador, on ajuda a fer la conversió per a poder utilitzar-lo.

@OneToMany, @ManyToOne, @OneToOne > tipus de relació entre entitats.

@JoinColumns i @IdClass > Anotacions especials per a poder fer claus primàries amb mes d'un camp, i després poder relacionar-ho amb un altra entitat per aquells camps. (Veure entitat Merito).

Una vegada definides totes les entitats, en el moment que s'executa l'aplicació, fa una connexió amb la base de dades i comprova si totes les entitats son correctes amb la definició de taules. En cas de no ser així o no existir les taules, hibernate intentarà crear les taules per a poder fer feina. O en el cas que hagi sigut una modificació de taula on ja existeixen dades, podria donar un error i no s'executarà l'aplicació degut a que no s'ha pogut modificar la estructura.



Il·lustració 18: Esquema entitat-relació generat PostgreSQL

Una vegada definides les entitats, s'ha de definir la capa de persistència en si, que ens ajudarà a fer les consultes i gestió de les dades contra la base de dades.

En el nostre cas el que farem es crear una interfície pròpia que hereti de la classe especial d'accés a base de dades del framework de Spring, anomenada JpaRespository.

Per exemple, per a la Entitat Bolsa, crearem el repositori de la següent manera:

```
import org.springframework.data.jpa.repository.JpaRepository;

/**
 * Clase accés a la taula Bolsa
 *
 * @author Jaume
 */
public interface IBolsaRepo extends JpaRepository<Bolsa, Integer> {


}
```

Il·lustració 19: Exemple repositori Borsa

En el moment que es defineix que el repositori es del tipus Bolsa, i també s'ha de definir el tipus de clau primària, que en el nostre cas es un Integer, i ja es pot accedir a una sèrie de mètodes i utilitats per l'accés a base de dades, com es mostra a la imatge de la dreta.

```
count() : long - CrudRepository
count(Example<S> example) : long - QueryBy
delete(Bolsa entity) : void - CrudRepository
deleteAll() : void - CrudRepository
deleteAll(Iterable<? extends Bolsa> entities) : v
deleteAllInBatch() : void - JpaRepository
deleteById(Integer id) : void - CrudRepository
deleteInBatch(Iterable<Bolsa> entities) : void
equals(Object obj) : boolean - Object
exists(Example<S> example) : boolean - Quer
existsById(Integer id) : boolean - CrudReposito
findAll() : List<Bolsa> - JpaRepository
findAll(Example<S> example) : List<S> - JpaR
findAll(Pageable pageable) : Page<Bolsa> - Pa
findAll(Sort sort) : List<Bolsa> - JpaRepository
findAll(Example<S> example, Pageable pagea
findAll(Example<S> example, Sort sort) : List<
findAllById(Iterable<Integer> ids) : List<Bolsa>
findById(Integer id) : Optional<Bolsa> - CrudR
findOne(Example<S> example) : Optional<S>
flush() : void - JpaRepository
```

Il·lustració 20: Llista mètodes per defecte del repositori

A mes, aquesta capa te la funcionalitat que es poden afegir nous mètodes d'accés a la base de dades, on, nomes indicant la definició del mètode amb una nomenclatura especial, ja no es necessari crear una implementació amb consulta SQL, ja que Spring Jpa ho fa automàticament. Per exemple, a la interfície de la classe Categoria, s'han definit dos mètodes especials:

```
public List<Categoria> findAllByOrderByDescripcionAsc();

public List<Categoria> findAllByDescripcion(String descripcion);
```

Il·lustració 21: Mètodes manuals dins repositori

On defininit amb la nomeclatura especial "findAllBy" "OrderBy" mes el nom del camp definit a la entitat, i per acabar "Asc" per a definir l'ordre ascendent.

En el segon cas, com s'ha definit que cerqui a tota la taula per descripció, el primer paràmetre ha de ser la descripció.

Està clar que una consulta a base de dades es pot complicar molt en el cas d'accedir a base de dades molt grans, per aquest cas, ja es tendría que definir una

clase que implementàs la interficie i fer la consulta SQL i una conversió a les dades necessaries, poguent ser mitjançant DTO, per a una millor optimització de les dades.

A aquesta aplicació s'ha pogut utilitzar en tot moment la definició de consultes simples, siguent molt efectiva la capa de JpaRepository afegida per Spring.

Tota aquesta definició de claus especials es poden veure a la pàgina de Spring[9].

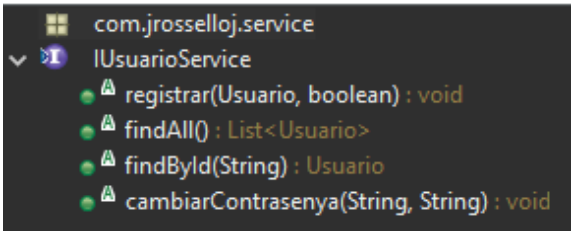
4.3. Capa de negoci

La capa de negoci es on es concentra tota la lògica de negoci de la nostra aplicació. Des de la definició a la part d'anàlisi fins al final del desenvolupament hi ha hagut variàncies degut a varis factors, com poden ser, falta de coneixement de la tecnologia i falta d'experiència.

Anem a revisar i analitzar els canvis paquet per paquet, i comencem per el paquet de Usuaris:

Definició inicial

Definició final



Els mètodes *login*(...) i *logout*() l'hem definit a altra servei degut a la definició especial que te la capa de seguretat i s'explicarà mes endavant.

Els mètodes *createUser* i *updateUser* s'han ajuntat dins un mateix mètode anomenat registrar, ja que la execució interna de JPA es la mateixa i es pot reaprofitar el codi.

Hem obviat el mètode *deleteUser* per a no eliminar possible historial dins la base de dades, a canvi s'ha definit una elminació lògica amb un paràmetre booleà dins la entitat *Usuario*.

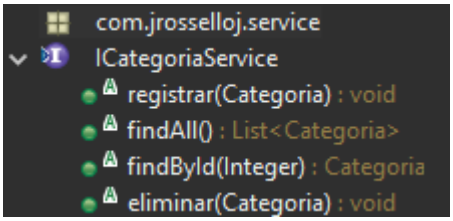
Els mètodes *findAll* i *findById* son consultes que no es varen tenir en compte a la fase d'analisí, molt essencials per a mostrar els usuaris per pantalla.

També es va afegir el mètode especial *cambiarContrasenya*(...) per la peculiaritat del tipus d'encriptació de les contrasenyes, on s'explicarà a la capa de seguretat.

Anem a revisar la definició de la capa de negoci de Categories:

Definició inicial

Definició final

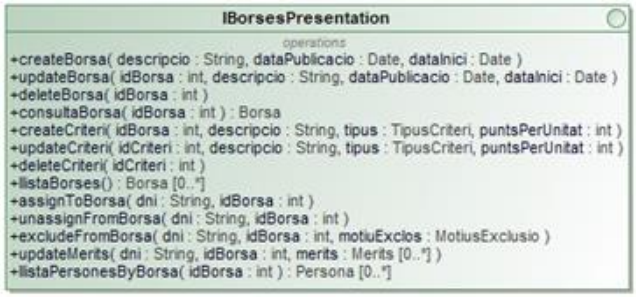


Com a la definició de usuaris, els mètodes de crear i modificar s'han unificat al mètode *registrar*(). I en aquest cas hem modificat el nom *deleteCategory* per *eliminar*().

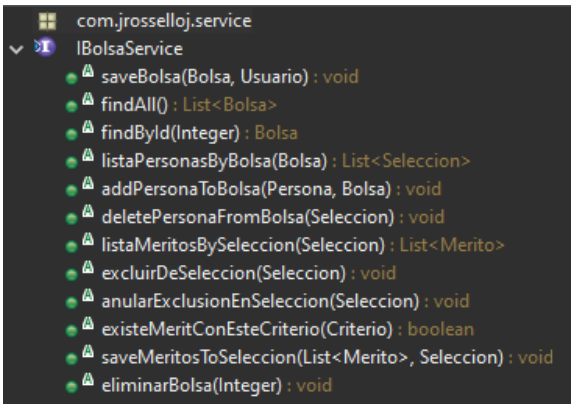
S'han afegit els mètodes *findAll* i *findById* per el mateix motiu que amb usuaris.

Continuem amb la definició de Borses:

Definició inicial



Definició final



Els mètodes *createBorsa* i *updateBorsa* s'han unificat a *saveBolsa*, així com els mètodes de CRUD de criteris (*createCriteri*, *updateCriteri* i *deleteCriteri*) ja que també utilitzarem el mètode *saveBolsa* amb la llista de criteris assignats, i la capa de persistència automàticament afegeix, modifica o elimina segons sigui necessari.

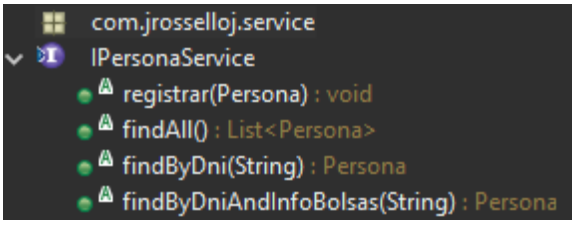
La definició ha canviat als següents mètodes: *delete* per *eliminarBolsa*, *consulta* per *findById*, *llistaBorses* per *findAll*, *assignToBorsa* per *addPersonaToBolsa*, *unassignFromBorsa* per *deletePersonaFromBolsa*, *excludeFromBorsa* per *excluirDeSeleccion*, *updateMerits* per *saveMeritosToSeleccion*, *llistaPersonesByBorsa* per *listaPersonasByBolsa*.

S'han definit tres mètodes nous: *listaMeritosBySeleccion*, per a consultar els mèrits de una persona a una borsa determinada (selecció). *anularExclusionEnSeleccion* per a tornar a incloure una persona dins una selecció si aquesta ha sigut exclosa per algun motiu. I *existeMeritConEsteCriterio* per a control d'eliminació de criteris d'una borsa si alguna persona de la selecció ja te puntuació definida damunt aquest criteri.

I per acabar, analitza'm la definició de Persones

Definició inicial

Definició final



Com a tots els casos, s'ha unificat *create* i *update* dins registrar. El mètode *consultaPersona* s'ha dividit en dos casos: *findByDni* i *findByDniAndInfoBolsas* utilitzada per a la consulta de l'usuari i s'afegeix la informació de la posició que ocupa a cada llista que està inscrit. La informació extra de l'ordenació es fa amb una anotació especial dins la entitat que es *@Transient*, on es poden definir valors sobre propietats de l'entitat i que no es sincronitzaran amb base de dades, i així poder utilitzar-les en temps d'execució.

Per a crear les implementacions dels serveis, primer s'ha de crear una interficie amb la definició dels mètodes i després crear la implementació d'aquests mètodes on, a aquesta nova clase, s'ha de definir amb l'anotació *@Service* per a definir que es un servei i que es pot instanciar amb un altra anotació especial *@Autowired*, on automaticament es crea una instancia de la definició dels mètodes del servei.

S'ha definit la excepció BorsesTreballException per a control d'errors on es pot incloure un missatge d'error controlat per a poder mostrar en pantalla.

4.4. Capa de presentació

La capa de presentació està dividida en dues parts, la part visual definida en arxius amb extensió xhtml amb el framework de primeFaces que es una extensió del framework definit per java JSF. I la segona part, la part controladora definida amb components Bean on es defineixen els atributs necessaris per enviar a la pantalla de presentació, on a la vegada fa l'enllaç amb la capa de negoci.

Els Components de vista es defineixen amb les anotacions especials de Spring *@Component* que serveix per a identificar a quines classes es poden accedir des de les pantalles xhtml, on es podrà accedir als distints atributs i mètodes indicant el nom del bean un punt i el nom de l'atribut o mètode a accedir.

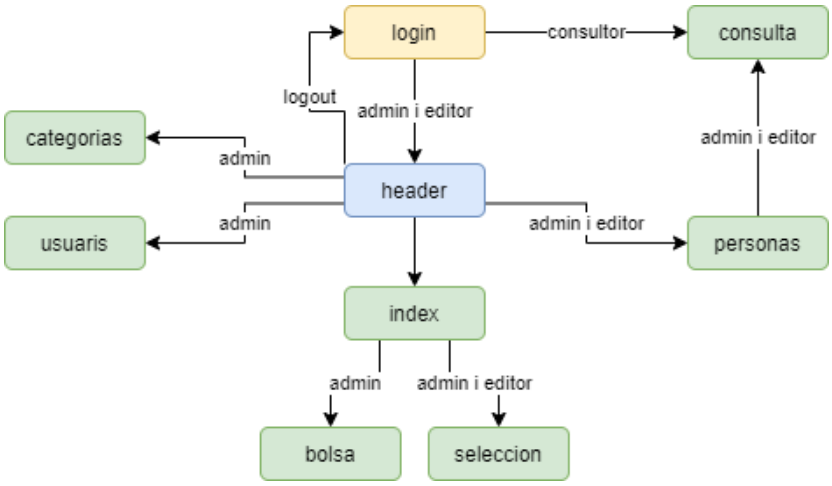
A la definició dels distints components s'ha d'indicar quin es l'abast de "vida" que tindrà. Per aquesta aplicació utilitzarem les visibilitats de vista i de sessió. Per a la definició de vista, utilitzarem la anotació *@Scope("view")*, on la sessió de la vista només existirà el moment que es mostri una pagina xhtml determinada, on es mantindrà l'abast mentre existeixi la pàgina i es facin cridades ajax (primefaces utilitza

aquest tipus de cridades per a actualitzar la informació dels seus components). Una vegada que es canvia de pàgina la vida d'aquella sessió desapareix, i es crearia una de nova a la nova visita de la pàgina.

Per a l'abast de Sessió, s'afegeix l'anotació *@SessionScope*, on l'abast d'aquest component es manté durant tota la sessió que te obert l'usuari des de que ha iniciat la sessió amb el seu nom i la contrasenya. A la nostra aplicació només tenim un component d'aquest tipus anomenat SessionBean on tenim alguns mètodes especials com es, recuperar el nom d'usuari de la sessió, els rol que te definit, quin idioma vol mostrar a la pantalla, etc.

La part visual està definit per distintes pantalles, on es comença la navegació des de la pantalla de login, on, segons el rol de l'usuari es mostra la pantalla d'índex o si es usuari de consulta va directe a mostrar la seva informació personal.

Es mostra un mapa web de les distintes pantalles que es pot accedir segons el rol identificat segons l'usuari.



Il·lustració 22: Mapa web segons rols

Totes les pàgines a excepció de la pantalla de login, estan definides amb un fitxer *template.xhtml*, on a la part superior es mostra una capçalera amb menú (*header.xhtml*) per a poder accedir directament i des de qualsevol vista de la aplicació a distintes pàgines. I segons els permisos del rol definit, es mostraran mes o menys opcions segons correspongui.

Per el desenvolupament de la part visual amb xhtml s'utilitza la tecnologia primefaces on no entrarem en explicacions tècniques, on principalment he consultat la pàgina d'exemples de primefaces [10] per a poder afegir els components corresponents i aprofitar al màxim el framework visual.

Per a poder convertir objectes a Strings i a l'inrevés per a poder mostrar, escollir i guardar la informació necessitem crear un nou component amb una herència de Converter i amb unes anotacions especials *@FacesConverter*, on definint dos

mètodes especials, *getAsObject* i *getAsString*, converteixen un objecte a *String* i a l'inrevés. En el nostre cas només ens ha fet falta per a la conversió de Categories, per a poder mostrar una llista de categories i poder-ne triar una i que es guardi la escollida a base de dades. Veure CategoriaConverter.

4.5. Capa de seguretat

Per a la seguretat de l'aplicació usarem les llibreries de Spring de Seguretat i accedirem amb un usuari i contrasenya guardades a la base dades on també ens retornarà una informació addicional com es l'idioma en que es mostrarà l'aplicació, i el rol d'accés a l'aplicació.

Una vegada afegides les llibreries dins el fitxer pom.xml de configuració de maven, hem de configurar l'aplicació per indicar com volem fer l'accés a la aplicació, on es guardaran els usuaris, quina encriptació volem usar per la contrasenya, etc.

Per a la configuració inicial de seguretat s'ha creat la classe *SecurityConfig* que hereta de *WebSecurityConfigurerAdapter* i on, heretant ja serveis de configuració basics de seguretat, es altament configurable i escalable. En el nostre cas les definicions que hem fet son:

- Al mètode *configure(HttpSecurity http)* es defineix quin es l'àmbit de la seguretat, en el nostre cas es tota l'aplicació menys la pàgina de login, on te la configuració especial indicant *loginPage* on s'escriurà l'usuari i la contrasenya i un botó de enviament de formulari. I també es defineix la ruta a la que accedirà una vegada fet *logout* des de qualsevol pàgina, i on l'aplicació retornarà sempre a la pàgina de login, obligant a tornar a escriure usuari i contrasenya si vols tornar a accedir a l'aplicació.

- Al mètode *configure(AuthenticationManagerBuilder auth)* es defineix de quina manera es comprova si la informació introduïda es correcte i si l'accés ha sigut correcte. L'aplicació l'hem configurada per a comprovar els usuaris dins la taula usuaris de la base de dades. Tota la comprovació es fa a un Servei extern on hereta de *UserDetailsService* i fent la comprovació d'usuari i contrasenya retorna la informació relativa a l'usuari, o en canvi retorna una excepció en el cas que l'usuari o la contrasenya fossin incorrectes.

- Al mateix mètodes s'ha configurat un usuari especial en memòria anomenat ADMIN_SIS, per a la primera vegada que s'executa l'aplicació, on només deixarà accedir a la pantalla de usuaris i poder definir algun usuari administrador per a poder començar a fer feina amb l'aplicació.

Totes les contrasenyes es guarden i es comproven amb un mètode d'encriptació anomenat BCryptPasswordEncoder que serveix per guardar les contrasenyes encriptades, i fer la comprovació amb el mateix mètode.

Per a guardar la informació de l'usuari una vegada accedit a l'aplicació s'ha creat la classe *MyUserDetails* heretant de *User*, on apart de guardar la informació bàsica de l'usuari, també es guarda l'idioma per defecte que ha escollit quan s'ha creat l'usuari, i així poder mostrar l'idioma correctament a l'aplicació.

4.6. Configuració general de l'aplicació

Per a la configuració Spring ha centralitzat la gran majoria de configuració dins una classe que implementa una interfície especial de configuració anomenada *WebMvcConfigurer* i una anotació especial @SpringBootApplication per identificar d'on carregar la configuració inicial.

La configuració es basa en la implementació de @Bean segons els distints paràmetres a implementar. Com a principals configuracions podem definir:

- Amb *ServletRegistrationBean facesServletRegistration()* definim la terminació que tindran els nostres fitxers de web i la pàgina principal una vegada accedit a l'aplicació.
- Amb *ServletContextInitializer servletContextInitializer()* definim els paràmetres inicials de l'aplicació. En el nostre cas s'ha definit el tema a utilitzar amb primefaces i la solució a un bug detectat a l'hora de mostrar les dates a segons quins components de primeFaces.
- Amb *CustomScopeConfigurer viewScope()* es pot configurar la definició de l'àmbit de vista.

Per a poder definir l'aplicació com a multi llenguatge s'han de crear tants de fitxers de propietats, fitxer amb una llista de clau=valor, com idiomes es vulguin mostrar. En el nostre cas hem creat 2 fitxers, un per català i un altra per castellà (*messages_ca.properties* i *messages_es.properties*) on ambdós fitxers tenen el mateix nombre de propietats amb la seva traducció. En aquest cas la configuració s'ha de crear per dues bandes, una per a poder utilitzar-les directament al codi xhtml i un altra per a poder utilitzar-les a al codi java.

Per a la configuració xhtml s'ha de modificar el fitxer faces-config.xml on es definirà la llibreria per a utilitzar les etiquetes, en el nostre cas utilitzarem la de Spring: *SpringBeanFacesELResolver*. També s'indiquen els idiomes configurats i la ruta dins la carpeta *resources* amb el nom base del fitxer de propietats, en el nostre cas "i18n.messages".

En canvi, per a poder utilitzar les propietats dins el codi java, hem de tornar a la classe de configuració de Spring i afegir el mètode *MessageSource messageSource()* retornant també la ruta i el nom base del fitxer de propietats.

```
messages_ca.properties
1 # COMUNS
2 guardar=Guarda
3 editar=Editar
4 total=Total
5 limpieza=Neteja
6 eliminar=Eliminar
7 buscar=Cercar
8 si=Si
9 no=No
10 puntos=punts
11
```

```
messages_es.properties
1 # COMUNS
2 guardar=Guardar
3 editar=Editar
4 total=Total
5 limpieza=Limpiar
6 eliminar=Eliminar
7 buscar=Buscar
8 si=Si
9 no=No
10 puntos=puntos
11
```

Il·lustració 23: Exemple fitxers internacionalització

Al fitxer *application.properties* es pot configurar les variables necessàries per connectar a base de dades. L'aplicació accedeix a una base de dades postgreSQL on la configuració mes important es username i password de la base de dades, url de la ubicació, driver a utilitzar i altres configuracions, com per exemple, mostrar les sql que s'executen al log. A desenvolupament es interesant activar aquesta configuració ja que mostra totes les connexions i es comprova que executa les consultes necessàries per a obtenir la informació.

Al mateix fitxer també s'ha configurat el port on arrancarà l'aplicació web, en el meu cas la vaig configurar al port 8081, ja que el port 8080, que es el port per defecte, ja tenia altre aplicació arrancada.

Tot el projecte d'spring boot utilitza l'eina maven per a la gestió de llibreries i la construcció del projecte per a poder executar-ho. La base d'un projecte Spring Boot ja ve configurat de base per a poder utilitzar maven, i només amb la execució de la configuració maven podem aconseguir tant la compilació del projecte i crear un fitxer jar per a poder incloure dins projecte java o un war per a poder afegir-ho dins un servidor apache per mostrar la vista web.

Per la part maven i la configuració de Spring Boot les llibreries incloses ja duen incorporat un servidor apache per a la execució de l'aplicació només executant una línia de codi.

4.7. Posant en marxa l'aplicació

Per a executar l'aplicació tot el que hem de fer es executar la següent línia de execució de maven:

> mvn spring-boot:run

On en un inici cercarà totes les llibreries necessàries que encara no es tenen al sistema. La primera vegada pot tardar uns 5 minuts, ja que per a cada dependència afegida al pom.xml maven cerca dins un repositori extern per a baixar-nos al nostre sistema les llibreries corresponents, i aquestes, poden a la vegada dur mes dependències.

```
Símbolo del sistema - mvn spring-boot:run

C:\workspaceTFG\borsestreball>mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.jrosselloj:borsestreball >--------------------
[INFO] Building borsestreball 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot-maven-plugin:2.4.2:run (default-cli) > test-compile @ borsestreball >>>
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ borsestreball ---
```

Il·lustració 24: Comanda maven per arrancar l'aplicació

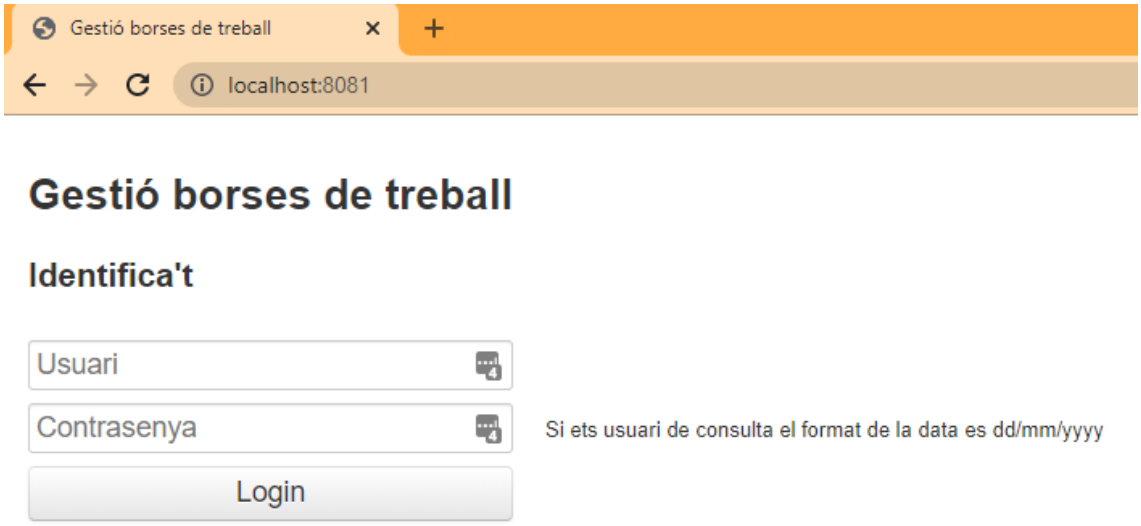
Una vegada davallades totes les dependències necessàries comença l'execució de l'aplicació, on ho podrem comprovar a partir de mostrar el logotip de Spring.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.2)

2021-05-30 18:49:55.837  INFO 16808 --- [           main] com.jrosselloj.BorsestreballApplication  : Starting BorsestreballApplication using Java 1.8.0
-PC with PID 16808 (C:\workspaceTFG\borsestreball\target\classes started by Jaume in C:\workspaceTFG\borsestreball)
2021-05-30 18:49:55.842  INFO 16808 --- [           main] com.jrosselloj.BorsestreballApplication  : No active profile set, falling back to default pro
t
2021-05-30 18:49:57.210  INFO 16808 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFA
2021-05-30 18:49:57.333  INFO 16808 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 114 ms
 repository interfaces.
2021-05-30 18:49:58.440  INFO 16808 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2021-05-30 18:49:58.453  INFO 16808 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-05-30 18:49:58.454  INFO 16808 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.41]
2021-05-30 18:49:58.856  INFO 16808 --- [           main] org.apache.jasper.servlet.TldScanner     : At least one JAR was scanned for TLDs yet containe
able debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning
startup time and JSP compilation time.
2021-05-30 18:49:58.885  INFO 16808 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-05-30 18:49:58.886  INFO 16808 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization complet
```

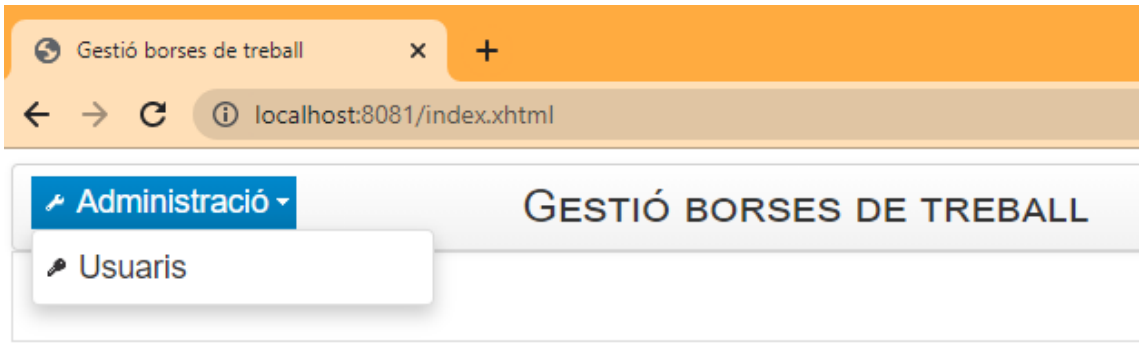
Il·lustració 25: Log d'inici de l'aplicació

Accedim a http://localhost:8081/ i mostra la pantalla de login:



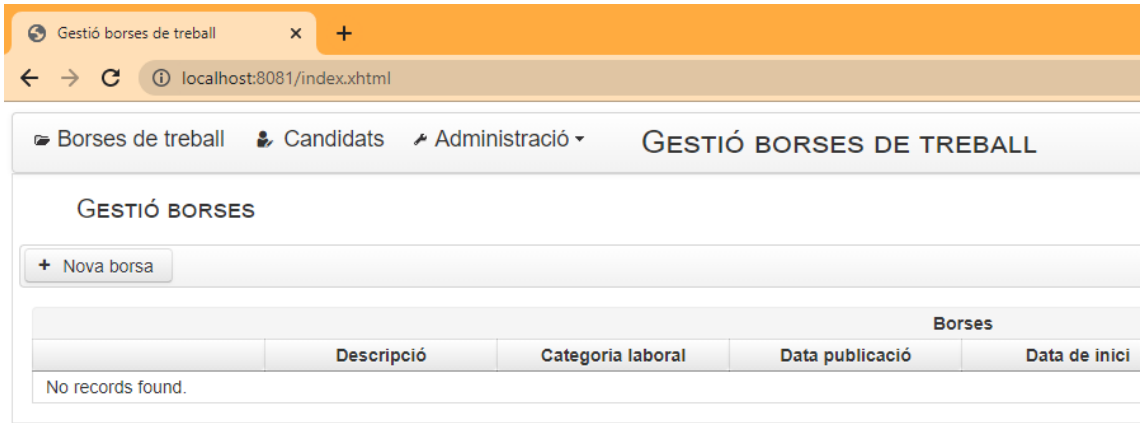
Il·lustració 26: Pantalla accés aplicació

El primer cop que arranca l'aplicació la base de dades es buida, però s'ha creat un usuari "administrador de sistema", per a poder crear un usuari administrador de l'aplicació per a poder començar. L'usuari es ADMIN_SIS amb la contrasenya 1234.



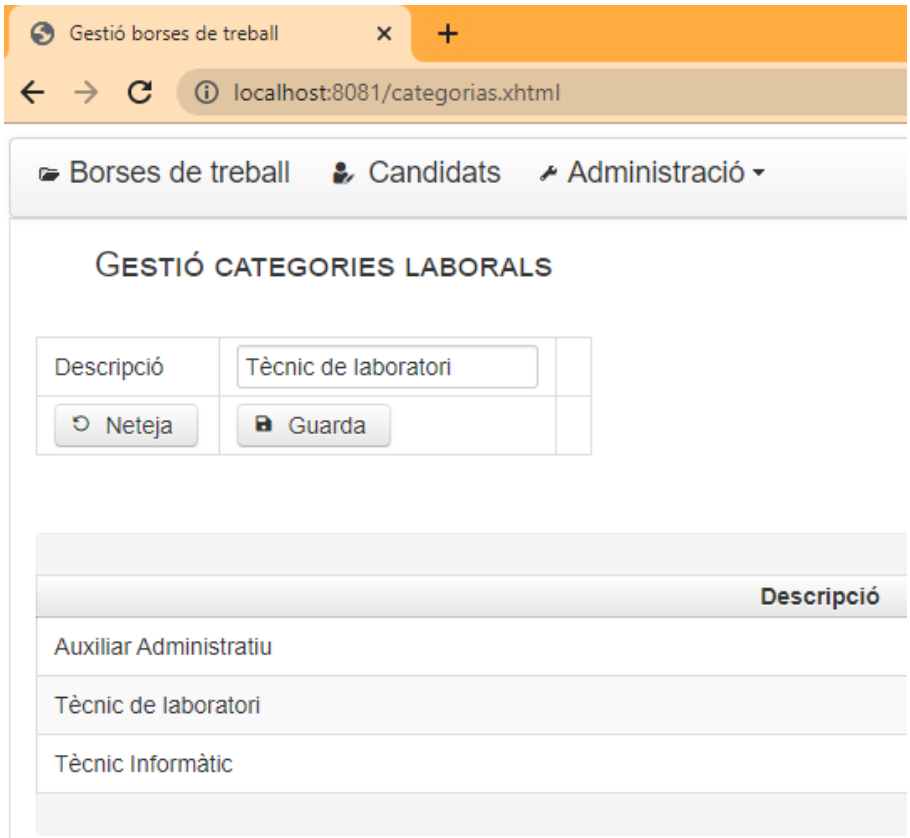
Il·lustració 27: Menú amb rol SIS_ADMIN

Es crea un usuari administrador i es torna a accedir a l'aplicació i ara si que ens mostra totes les opcions de l'usuari administrador.



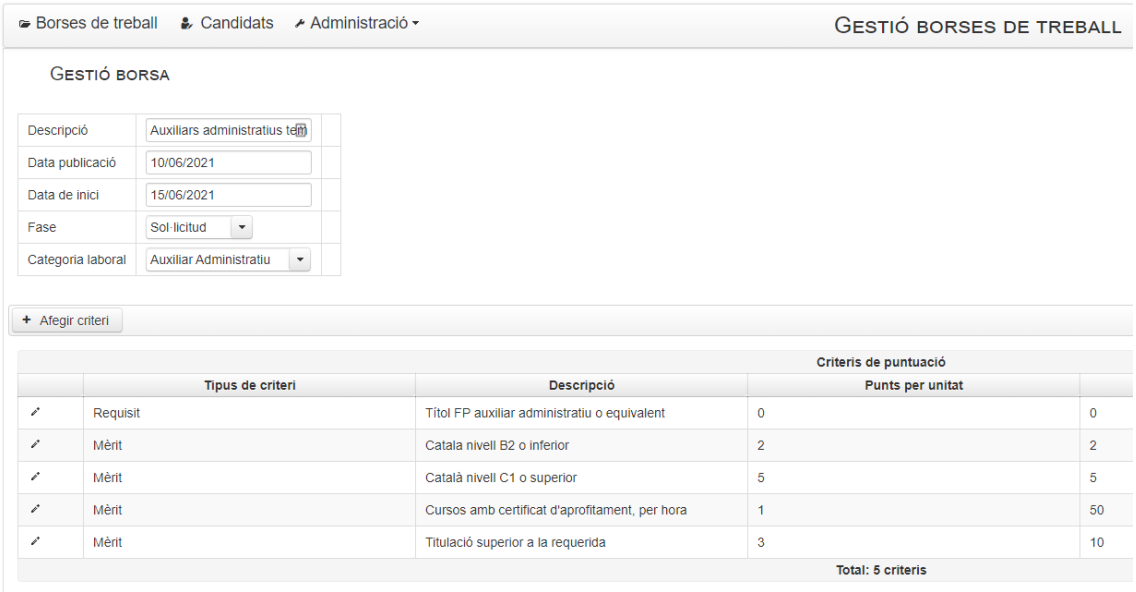
Il·lustració 28: Pantalla inicial borses de treball amb rol Admin

Per a crear una bosa de treball, el primer que hem de fer es crear les categories laborals, ja que son una dada obligatoria per a crear una nova borsa. Administració > Categories laborals.



Il·lustració 29: Pantalla gestió de categories laborals

I des de el menu principal, ara ja podem crear una nova borsa. Borses de treball > + Nova borsa



Il·lustració 30: Pantalla gestió de borsa

Una vegada generada la nova borsa, afegirem noves inscripcions de persones que han presentat una sol·licitud. Per això, el primer que hem de fer es crear aquestes persones a la BBDD, si aquestes no hi fossin, ja que una persona pot estar a varies bores. Accedirem des del menú a Candidats.

Per a provar, crearem 4 candidats amb distints documents

Candidats				
Document	Nom complet	Telèfon	Data de naixament	
doc1	Pericàs Alemany, Joan	666555666	01/01/1980	Català
doc2	Bosch Olmedo, Vanesa	666666666	02/02/1982	Castellà
doc3	Ibañez Anton, Roberto	555555555	03/03/1983	Català
doc4	Arenas Ruiz, Natalia	444444444	04/04/1984	Català
			Total: 4 candidats	

Il·lustració 31: Exemple pantalla candidats

I accedirem a la pantalla d'inscripcions de la borsa, per a afegir aquestes persones

Il·lustració 32: Pantalla gestió borses, botó accés inscripcions

Clicarem sobre +Afegir sol·licitud i escriurem els documents i els afegirem a la llista. I podem provar d'excloure a un d'ells.

Il·lustració 33: Pantalla gestió inscripcions a borsa

Les borses tenen distintes fases:

- Sol·licitud: Es poden afegir noves sol·licituds i modificar-les (incloure o excloure).
- Esmenes: Només es poden modificar les inscripcions (incloure o excloure).

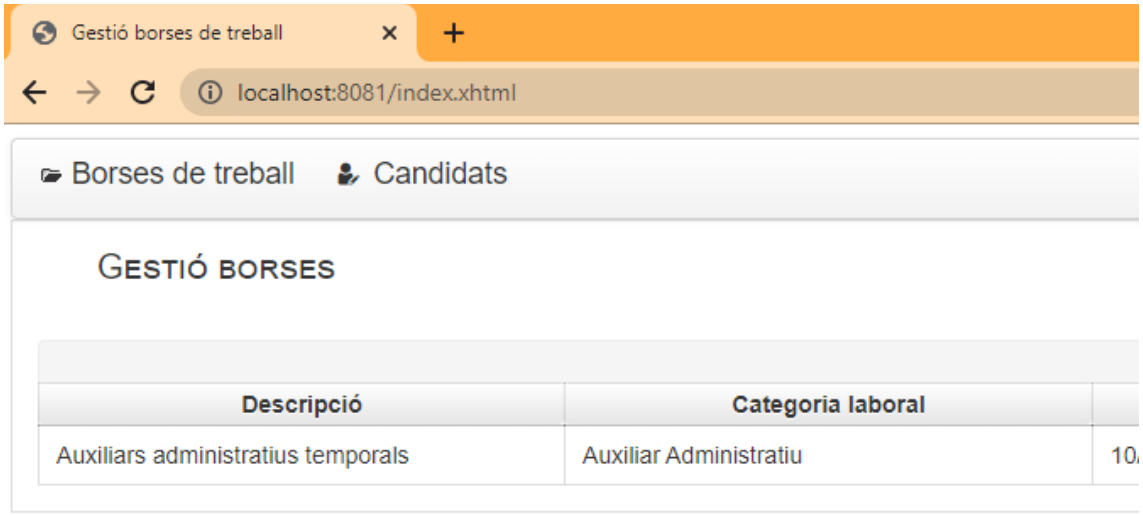
- Mèrits: Només es poden gestionar els mèrits dels candidats (només els inclosos a llista).
- Finalitzada: Borsa només de consulta per a poder comprovar la llista de candidats, la borsa segueix activa.
- Tancada: Ja no es poden cridar a mes candidats d'aquesta llista.

- L'usuari administrador pot crear usuaris, categories i borses.
- L'usuari editor pot gestionar borses i crear candidats
- L'usuari consultor es qualsevol candidat on el DNI es l'usuari i la data de naixement es la seva contrasenya.

Per a temes de seguretat a l'usuari editor nomes pot accedir a les pantalles de gestió de persones a la llista, però no pot modificar la situació de la borsa, ja que només tindria accés una persona encarregada.

Per a provar això, canviarem l'estat de la borsa a mèrits (Borses de treball > editar borsa > canviar valor de la llista de fase), per a afegir puntuacions als usuaris i crear un usuari editor (Administració > usuaris > crear usuari amb rol editor) i afegir els valors amb aquest usuari.

Una vegada accedit a l'aplicació, comprovem que ara nomes es mostren les opcions a les quals te accés:



Il·lustració 34: Pantalla gestió borses amb rol Editor

Accedim a la pantalla inscripcions i afegirem puntuació als candidats:

Mèrits

Tipus de criteri	Descripció	Punts per unitat	Puntuació màxima	Unitats	Total
Requisit	Títol FP auxiliar administratiu o equivalent	0	0	1	0
Mèrit	Catala nivell B2 o inferior	2	2		
Mèrit	Català nivell C1 o superior	5	5	1	5
Mèrit	Cursos amb certificat d'aprofitament, per hora	1	50	15	15
Mèrit	Titulació superior a la requerida	3	10	4	10

Guardar mèrits **Total: 30 punts**

Il·lustració 35: Exemple pantalla mèrits d'un candidat

I es comprova que la llista ha quedat ordenada per puntuació

Gestió borses de treball

localhost:8081/index.xhtml

Borses de treball Candidats GESTIÓ BORSES DE TREBALL

GESTIÓ INSCRIPCIONS | AUXILIARS ADMINISTRATIUS TEMPORALS | AUXILIAR ADMINISTRATIU

Borsa en fase de Mèrits > Només es poden gestionar els mèrits dels candidats (només els inclosos a llista).

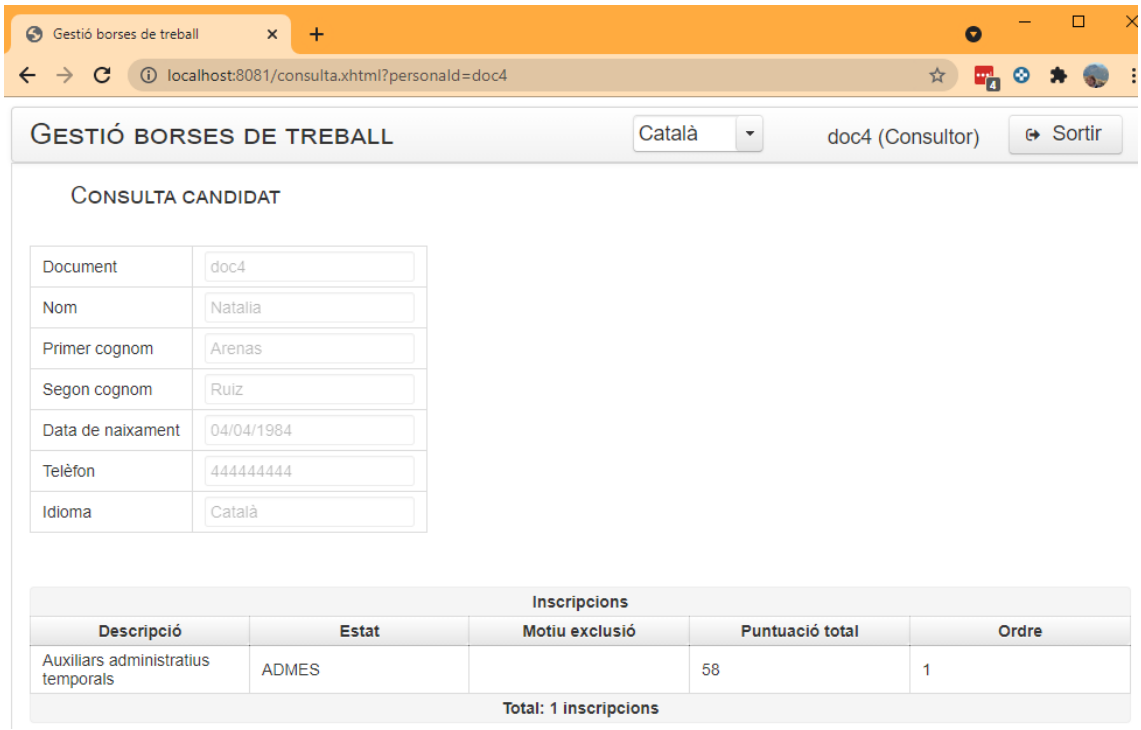
Inscripcions					
Ordre	Document	Nom complet	Telèfon	Puntuació total	
1	doc4	Arenas Ruiz, Natalia	444444444	58	Admès
2	doc1	Pericàs Alemany, Joan	666555666	30	Admès
3	doc2	Bosch Olmedo, Vanesa	666666666	15	Admès
	doc3	Ibañez Anton, Roberto	555555555	0	Exclòs
Total: 4 inscrits					

generar PDF

Il·lustració 36: Pantalla gestió inscripcions. Llista candidats

Una vegada acabat el procés, l'usuari administrador podria tancar el procés, i ja no es podrien modificar les puntuacions i la borsa quedaria definida.

Tots els candidats se lis ha generat un usuari i una contrasenya per a poder comprovar les seves inscripcions. L'usuari es el nombre de document i la contrasenya es la data de naixement amb format dd/mm/aaaa. I es consultaria la informació de la següent manera:

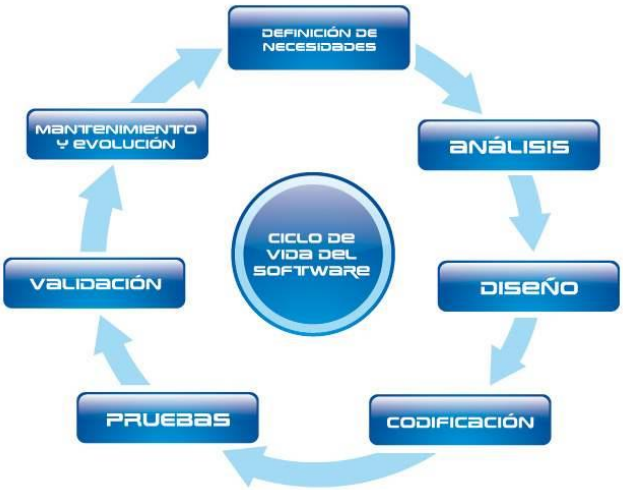


Il·lustració 37: Pantalla consulta candidats

4. Conclusions

Una de les lliçons apreses amb aquest treball de final de grau es la de fer la creació d'un projecte complet i en solitari amb els seus avantatges i desavantatges.

Una vegada començat amb el projecte i ja havent trobat més d'un problema a solucionar sense que això ens compliqui el temps planificat, comprova que el guió de cicle de vida del software es una base del que s'ha tingut que seguir perquè el projecte arribi a bon port.



Il·lustració 38: Esquema cicle vida del software

Veiem que la vida del software es un procés circular, on una vegada acabat un cicle, es pot tornar a començar amb un altra fase, tant podria ser per fer millores, com una ampliació de la fase anterior o una revisió de si s'adapta a l'estipulat per l'usuari (node de manteniment i evolució).

Però el treball de final de grau podríem dir que hem fet casi la volta d'un cicle, on el meu treball es pot dir que acaba a la fase de proves, encara que podríem dir que la part de codificació i proves van molt relacionades.

La part mes enriquidora, i a la vegada mes complicada ha sigut una vegada feta la definició de necessitats, generar l'anàlisi i valorar el temps necessari per a dur a terme la resta del cicle, com es el disseny, codificació i proves. Tot això acabant amb una aplicació funcional i basada en l'estipulació de les necessitats de l'usuari.

La fase de codificació a moments també s'ha tornat complicada per diversos motius, a vegades per mala valoració del temps, altres vegades per desconeixements tècnics i haver d'investigar i a vegades fer proves d'assaig-error per a trobar la solució més adient.

Però en general podria dir-se que s'ha assolit l'objectiu plantejat des de un inici, tant per temps com per la qualitat de l'aplicació. Com a reflexió podria dir que seguint en part un treball creatiu sempre queda la sensació que es podrien millorar algunes parts, tant com part visual, com per part funcional. Ja que a vegades es complicat canviar el rol de desenvolupador a usuari final i poder crear l'aplicació el mes funcional possible per a l'usuari, ja que es per ell per a qui està feta l'aplicació.

En tot moment s'ha intentat seguir la planificació, encara que, al començament quan està tot per començar costa enfocar l'objectiu de l'aplicació i els temps son mes relatius i s'allarguen un poc mes. Però per altra banda s'han pogut recuperar per altres

fites, on es podria fer una recapitulació i es podrien canviar els períodes d'algunes fases del projecte. En contra, hi ha punts que no s'han tingut en compte, com ha sigut algunes configuracions internes del projecte que han fet perdre algunes jornades, i tot això ha fet recapitular i no fer la fase de proves unitàries amb jUnit que estava planificada en un principi.

Aquest treball s'ha basat en una idea bàsica de gestió de borses de treball, però, per exemple a l'administració pública totes les borses s'han d'estipular a una determinada llei, tant sigui nacional com d'àmbit local, com per exemple podrien ser dates, terminis de lliurament, titulacions obligatòries, titulacions no valides, criteris ordenació llistes, percentatge d'homes i dones, etc.

Però tenint un projecte base, amb la gestió d'usuaris, candidats i borses, només es tindrien que fer adaptacions de las parts necessàries sense modificar altres parts de l'aplicació.

Altres idees que es podrien afegir seria:

- Gestió de cridades de telèfon, per a dur un registre de les persones del dia i hora que s'ha cridat, i quin ha sigut el resultat, com podria ser que acceptes la plaça, com que renuncia, o no l'hi interessa temporalment, etc.
- Gestió de places, per a poder assignar una plaça quan a algun candidat l'hagi acceptat, i poder dur un historial dels candidats que han acceptat places.
- Una part a millorar es el disseny de possibles publicacions imprimibles, com podria ser un pdf. En aquesta fase s'ha fet un exemple senzill però el disseny es una feina farragosa que a aquesta fase no donava temps.

5. Glossari

- **Borsa de treball**: llista de persones presentades a una oferta de concurs on normalment l'ordre d'accés a la plaça de feina es comptabilitza amb els punts obtinguts per les distintes experiències aportades per el candidat.
- **Candidat:** persona que es presenta a un concurs per un lloc de feina.
- **Baremació:** efecte d'establir una barem de avaluació per a gestionar una preferència.
- **Concurrència:** en el cas de l'aplicació es refereix a l'accés a un mateix fitxer a la vegada per distintes persones.
- **Plaça/lloc de feina:** l'àrea de desenvolupament d'una feina definida a la oferta per definir quines seran les funcions de l'empleat.

6. Bibliografia

- [1] https://spring.io/projects/spring-boot
  Spring Boot – Documentation

- [2] https://maven.apache.org/
  Pàgina projecte maven

- [3] https://www.primefaces.org/
  Pàgina inicial framework primefaces

- [4] https://www.oracle.com/es/java/technologies/javase/javase-jdk8-downloads.html
  Pàgina descarrega jdk

- [5] https://www.eclipse.org/
  Pàgina projecte eclipse IDE

- [6] https://www.postgresql.org/
  Pàgina inici projecte base de dades PostgreSQL

- [7] https://github.com/gitextensions
  Git extensions

- [8] https://start.spring.io/
  Pàgina creació projecte inicial de Spring Boot

- [9] https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
  Spring Data JPA – Reference Documentation

- [10] https://www.primefaces.org/showcase
  Pàgina portfoli d'exemples de primefaces

7. Annex – Guia d'instal·lació

7.1. Instal·lar JDK 8

Descarregar una versió de JDK 8, per exemple la 8.291

https://www.oracle.com/es/java/technologies/javase/javase-jdk8-downloads.html

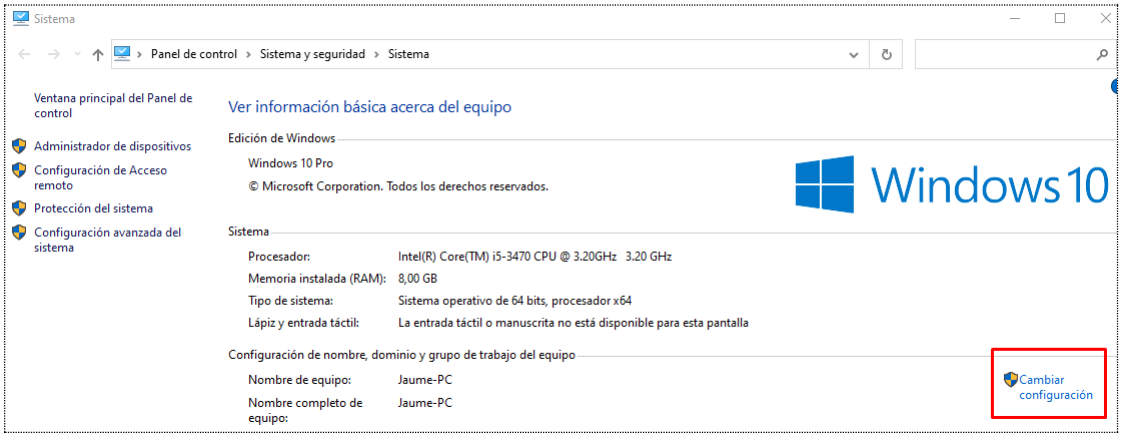
En el meu cas es per Windows x64

Windows x86	155.67 MB	jdk-8u291-windows-i586.exe
Windows x64	168.67 MB	jdk-8u291-windows-x64.exe

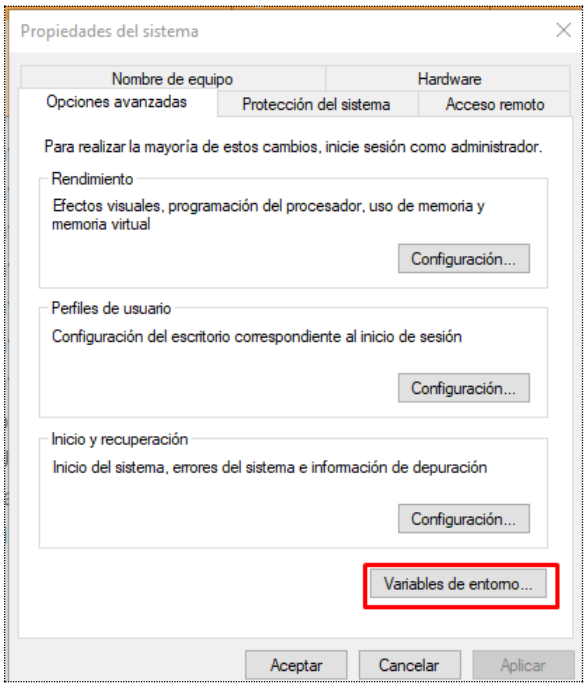
NOTA: per a poder descarregar de la pàgina d'oracle s'ha de crear un compte gratuït.

Instalar amb les opcions per defecte, i afegir les següents variables d'entorn:

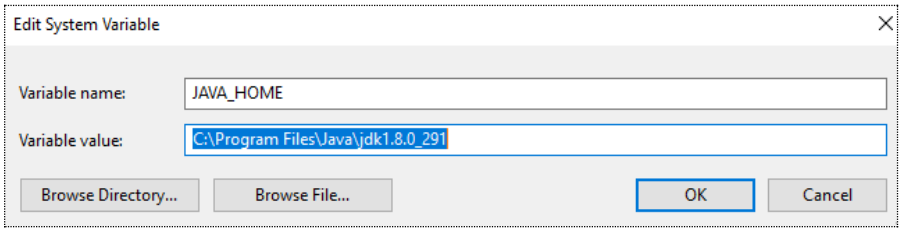
A windows 10: Sistema > Cambiar configuracion



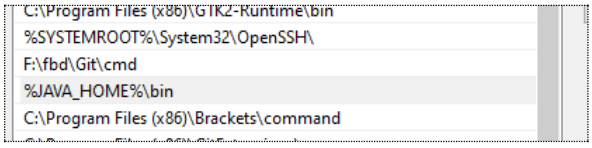
Anar a “variables de entorno”



Crear: JAVA_HOME=C:\Program Files\Java\jdk1.8.0_101 (modificar per la instal·lada)



Afegir a Path=%JAVA_HOME%\bin;



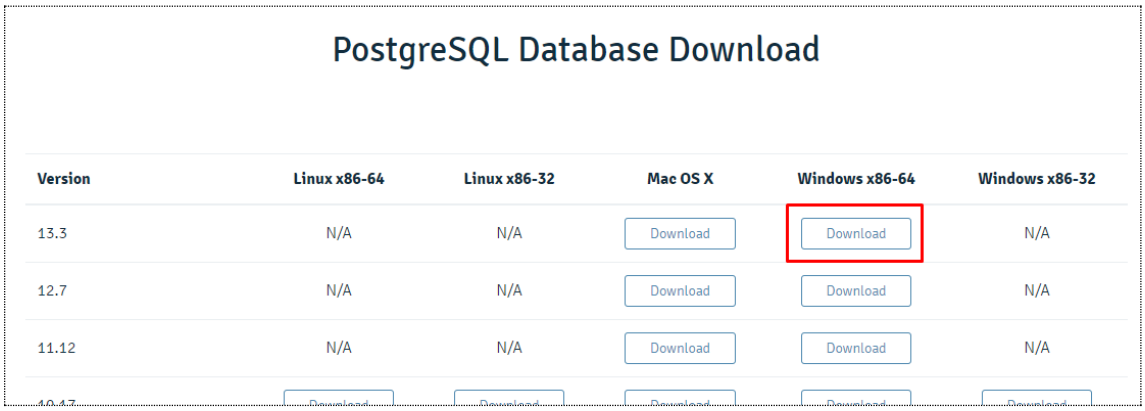
Comprovar si ha anat be executant des de qualsevol ruta

```
C:\javac -version
```

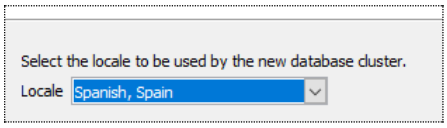
7.2. Instal·lar PostgreSQL

Descarregar i instal·lar la darrera versió de PostgreSQL

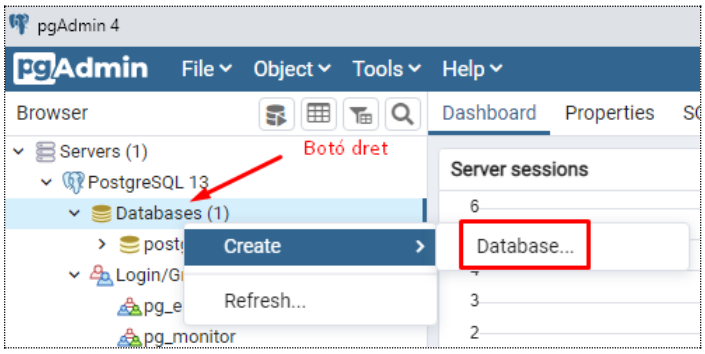
https://www.postgresql.org/download/
https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

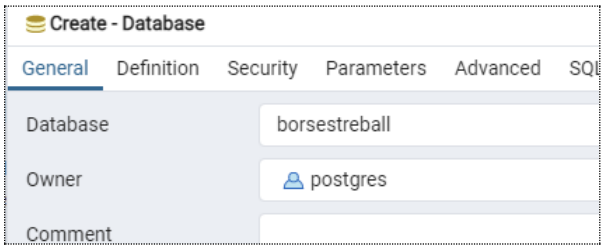


- Instal·lar amb les opcions per defecte.
- Quan demana la contrasenya si s'indica 1234, no es tendra que modificar la configuració de l'aplicació. Si no, es tindrà que modificar mes endavant per a poder connectar correctament.
- Com a port, deixar per el que ve per defecte: 5432.
- Canviar es Locale per Spanish, Spain



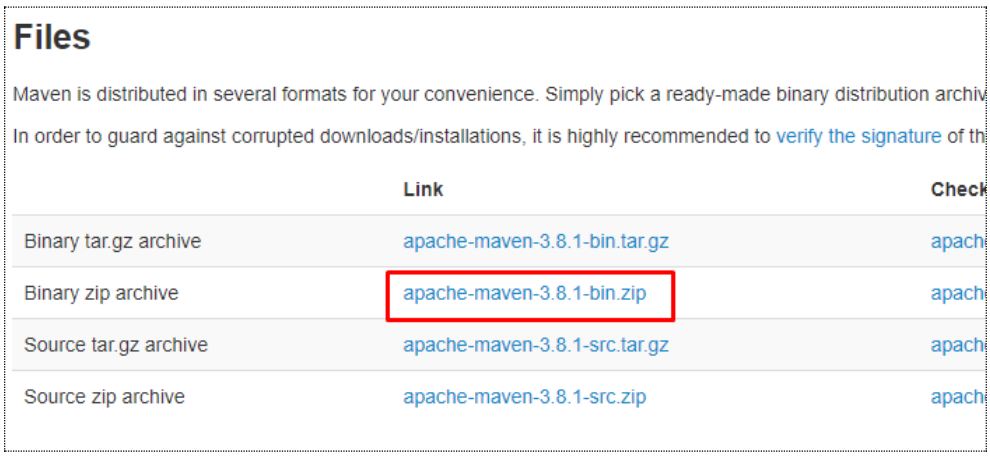
Una vegada instal·lat executar **pgAdmin4** i crear la base de dades **borsestreball** buida



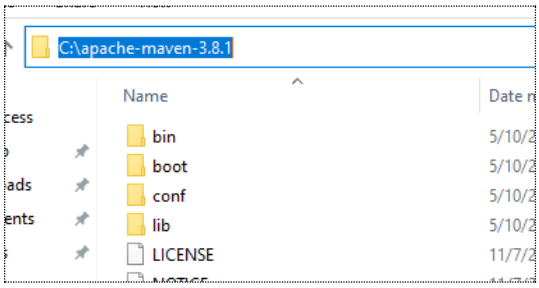


7.3. Instal·lar maven

https://maven.apache.org/
https://maven.apache.org/download.cgi

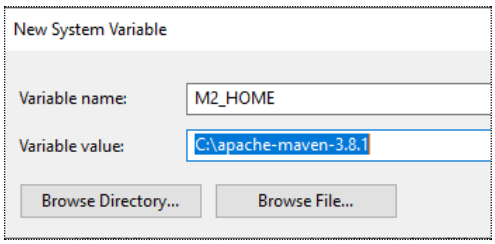


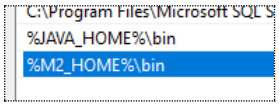
Descomprimir el contingut a una carpeta:



Afegir les següents variables d'entorn com s'ha fet a la instalació de Java

```
M2_HOME=C:\apache-maven-3.8.1
Afegir a path=%M2_HOME%\bin
```





Per a comprovar si s'ha instal·lat correctament podem executar **mvn -v**

```
C:\Users\User>mvn -v
Apache Maven 3.8.1 (05c21c65bdfed0f71a2f2ada8b84da59348c4c5d)
Maven home: C:\apache-maven-3.8.1\bin\..
Java version: 1.8.0_291, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_291\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

7.4. Configurar i executar la aplicació

L'aplicació està configurada per a arrancar al port 8081

Com a usuari d'acces a la base de dades, utilitzo l'usuari admin de postgres (que es "postgres")

I com a password vaig definir 1234 com a password de l'usuari admin

En el cas que es vulgui canviar el port o si s'ha definit un altra password per a l'usuari administrador, s'ha de modificar al fitxer **\src\main\resources\application.properties** i modificar les línies següents

```
server.port=8081
spring.datasource.username=postgres
spring.datasource.password=1234
```

Dins la carpeta del projecte hi ha el codi font dins la carpeta \src i també una llibreria all-themes.1.0.10.jar i el fitxer pom.xml que es fitxer de configuració de l'aplicació per maven.

La llibreria jar s'ha d'instal·lar dins el repositori de maven per a que el projecte la pugui utilitzar. S'ha de fer d'aquesta manera, ja que el repositori remot de primefaces redirecciona a un servidor ssl, i a vegades dona error d'accés.

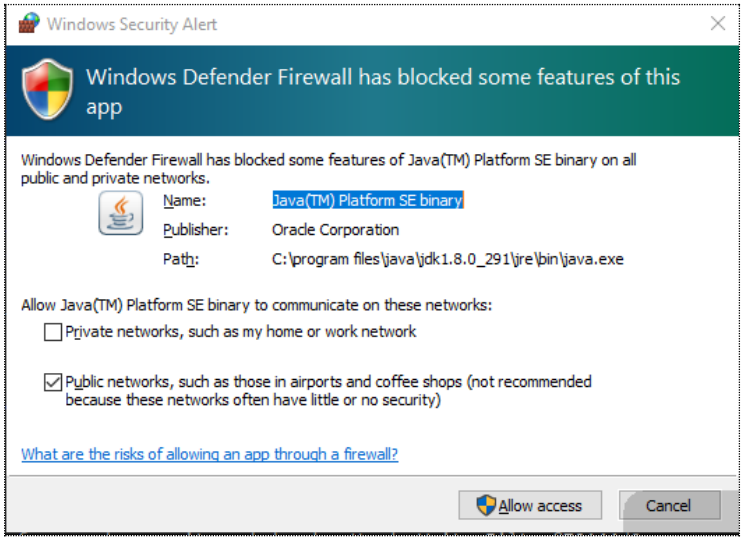
Executar la següent línia dins la carpeta del projecte per a poder instal·lar la llibreria del tema de primefaces:

```
> mvn install:install-file -Dfile=all-themes-1.0.10.jar -
DgroupId=org.primefaces.themes -DartifactId=all-themes -
Dversion=1.0.10 -Dpackaging=jar
```

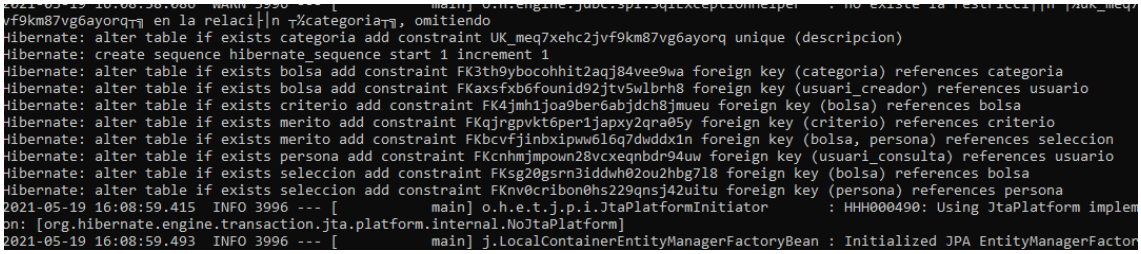
Una vegada configurat i instal·lada la llibreria, es pot arrancar l'aplicació, on, el primer pic que l'executem tardarà una estona degut a davallar i instal·lar la base de spring-boot per a poder executar l'aplicació

```
> mvn spring-boot:run
```

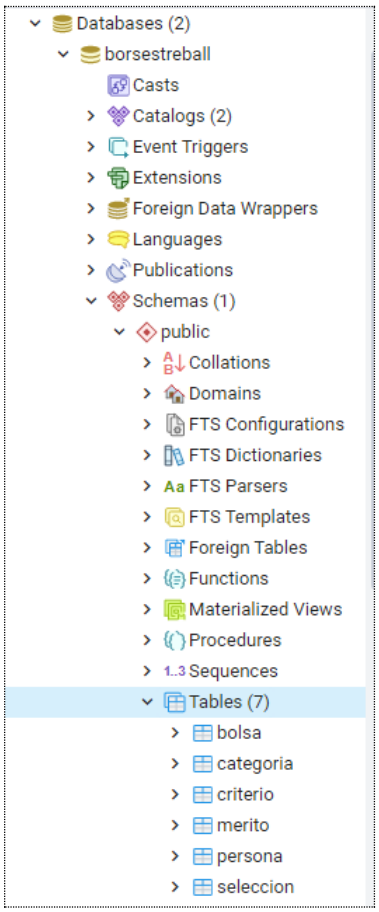
Si ens mostra un missatge del Firewall de Windows, donar permisos d'accés.



Una vegada instalades les llibrerias, es posa en marxa l'aplicació i veiem que ha accedit correctament a la BBDD creant les taules necessaries:

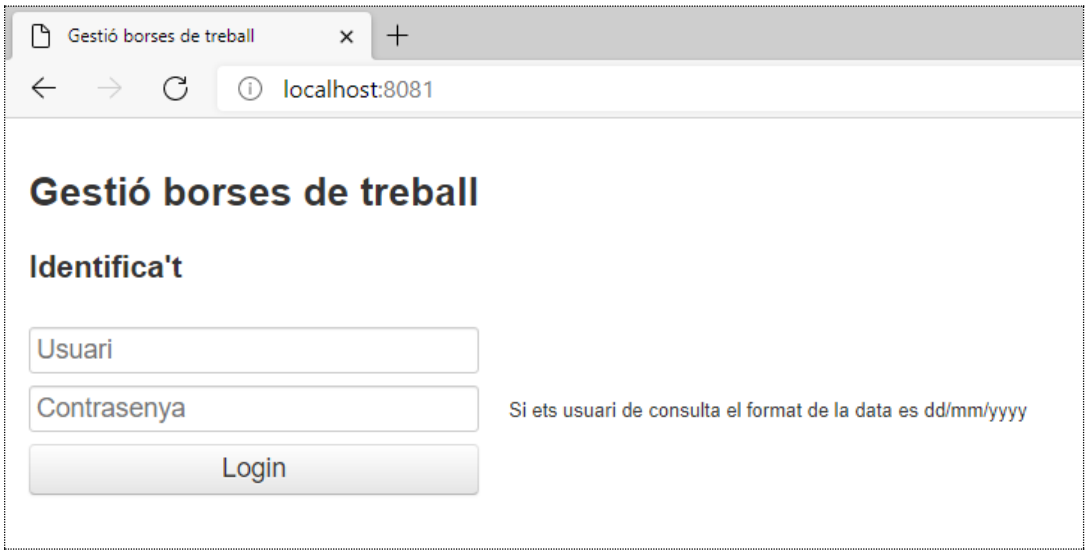


I si ho volem comprovar a postGres, des de pgAdmin podem veure les noves taules



Una vegada en marxa, podem accedir a l'aplicació

http://localhost:8081



Com a usuari inicial s'ha creat un administrador de sistema que només deixa accedir a la pantalla d'usuaris i així poder crear un usuari administrador de l'aplicació per a començar a crear borses.

Usuari: ADMIN_SIS
Contrasenya: 1234

Una vegada creat l'usuari administrador, s'ha de sortir de l'aplicació i tornar a entrar amb les noves credencials.

Per a poder crear la primera borsa de feina, primer s'ha de afegir una categoria laboral per a poder assignarla a una borsa.

Per a poder afegir a gent dins una borsa, primer s'han de crear candidats on després les cercarem per document per a poder afegir-los.

Les borses tenen distintes fases:

- **Sol·licitud:** Es poden afegir noves sol·licituds i modificar-les (incloure o excloure).
- **Esmenes:** Només es poden modificar les inscripcions (incloure o excloure).
- **Mèrits:** Només es poden gestionar els mèrits dels candidats (només els inclosos a llista).
- **Finalitzada:** Borsa només de consulta per a poder comprovar la llista de candidats, la borsa segueix activa.
- **Tancada:** Ja no es poden cridar a mes candidats d'aquesta llista.

- L'usuari **administrador** pot crear usuaris, categories i borses.
- L'usuari **editor** pot gestionar borses i crear candidats
- L'usuari **consultor** es qualsevol candidat on el DNI es l'usuari i la data de naixement es la seva contrasenya.